Broadview
www.broadview.com.cn

百炼成钢系列丛书

超值、大容量DVD
400分钟多媒体教学视频

— 内容精彩：
200个实例帮助读者领略Android开发的魅力

— 循序渐进：
实例由浅入深，随着读者的进步而逐步提高学习难度

— 实操性强：
按照书中讲解操作，即可轻松掌握相应的实例

— 面面俱到：
实例涵盖Android应用开发的多个方面，随查随用

— 举一反三：
把书中的例子改头换面，就可以完成手头的工作

Android 编程经典200例

◎楚无咎 编著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Broadview®
www.broadview.com.cn
博文视点 · IT出版旗舰品牌
技术凝聚实力 · 专业创新出版

本书读者对象

- 具备初步Android知识的用户
- Android开发入门学习者
- Android开发移动应用的编程爱好者
- 编程零基础初学者
- 业余应用制作的爱好者

上架建议：程序设计 >Android

新浪微博 weibo.com
@博文视点Broadview

策划编辑：胡辛征
责任编辑：徐津平
封面设计：李　玲

ISBN 978-7-121-20535-4

定价：79.00元（含DVD光盘1张）

百炼

Android
编程经典200例

◎楚无咎　编著

電子工業出版社
Publishing House of Electronics Industry

内 容 简 介

本书通过200个经典实例全面、系统地介绍了Android平台下的软件开发知识，重点突出、涉及面广、实用性强，在实例的讲解过程中还详细分析了开发思路及侧重点，使读者达到举一反三的效果。

全书分为14章，分别为：Android简介、Android简单控件的开发及应用、Android高级控件的开发及应用、手机用户界面、手机通信服务及手机控制、手机的自动服务功能、手机文件I/O与数据库的应用、手机网络应用、手机的Google服务功能、手机多媒体服务功能、Android手机的3D世界、手机特效开发、休闲游戏——Q版疯狂大炮、娱乐游戏——3D迷宫。

本书内容由浅入深，从Android平台下开发应用软件的基础知识到开发大型商务软件、3D游戏以及2D游戏，开发思路清晰明了、语言简明扼要，非常适合初学者和Android开发人员阅读参考。

图书在版编目（CIP）数据

Android编程经典200例 / 楚无咎编著. —北京：电子工业出版社，2013.8
（百炼成钢）
ISBN 978-7-121-20535-4

Ⅰ. ①A… Ⅱ. ①楚… Ⅲ. ①移动终端－应用程序－程序设计 Ⅳ. ①TN929.53

中国版本图书馆CIP数据核字（2013）第111691号

责任编辑：徐津平
特约编辑：顾慧芳
印　　刷：北京中新伟业印刷有限公司
装　　订：三河市皇庄路通装订厂
出版发行：电子工业出版社
　　北京市海淀区万寿路173信箱　邮编100036
开　　本：787×1092　1/16　印张：40.25　字数：1159.2千字
印　　次：2013年8月第1次印刷
印　　数：4000册　定价：79.00元（含DVD光盘1张）

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

随着 Google Android 新版本的发布，Android 除了应用在智能手机上以外，目前已成功地进入平板电脑市场，而 Android Market 的软件数量，也在以惊人的速度增长。此外，各手机及平板电脑厂商，更是纷纷投入以 Google Android 为平台的产品研发。根据以上现象，可以预见 Google Android 系统的产品在不久以后将成为市场的主流。

Google 向全世界推广 Android 的策略是持续的，而且推广的力度很大。随着越来越多硬件厂商的加入，精心设计的程序通过各种接口到达 Android 终端设备，如手机、平板电脑、手持游戏设备、数字相框、电子书阅读器、电子书和 Google TV 电视盒等。

本书结合作者多年的编程经验，介绍了 Android 平台下软件开发的基础知识，并且全书详解了 200 个实际实例，协助读者掌握应用重点，享受开发乐趣。不论你是 Android 初版的读者，还是刚加入 Android 开发领域的新人，都能在轻松愉快的氛围中迅速上手。

本书特点

1. 内容全

通过众多经典实例对 Android 常用的知识点进行了详细的介绍，同时剖析每个概念，让读者对 Android 应用开发有一个全面的认识。

2. 实例多

为了帮助读者快速掌握 Android，本书对每一个知识点都安排了相应的实例代码，让读者通过实例掌握关键知识。读者只需将代码输入计算机调试，即可轻松地掌握相关的知识。

3. 实用性强

本书采用 Android 应用程序常用的知识点，并结合实例讲解，让读者在实际应用中能够快速上手，同时也方便读者对程序进一步扩展。

4. 作为参考书

本书也是一本百科全书式的图书，知识全面，即查即用。可将本书作为 Android 应用程序开发的参考书。

本书内容及结构体系

第 1 章　Android 简介

本章介绍了 Android 的诞生、特点以及开发环境的搭建。

第 2 章　Android 简单控件的开发及应用

本章通过 25 个实例以分类的方式简单介绍了各个基本控件的应用。通过本章的学习，初学者可以做几个简单的实例。

第 3 章　Android 高级控件的开发及应用

本章通过 18 个实例以分类的方式详细介绍了 Android 平台下高级控件的应用。通过本章的

学习，读者可以做出稍微复杂点的实例。

第 4 章　手机用户界面的开发

本章通过 15 个实例介绍了 Android 平台下基于界面的开发，主要有界面响应事件、定时改变 Toast 消息以及使用 Bundle 实现 Activity 间的数据传送等。

第 5 章　手机通信服务及手机控制

本章通过 20 个实例介绍了手机通信服务以及手机桌面背景的切换，以及查看手机 SIM 卡的信息。

第 6 章　手机的自动服务功能

本章通过 15 个实例介绍了 Android 手机自动服务功能，主要有提醒用户收短信、查看手机电池剩余电量、更改手机模式等。

第 7 章　手机文件 I/O 与数据库的应用

本章通过 8 个实例介绍了有关 Android 手机的文本阅读、文件管理，以及 Android 手机端 Sqlite 数据库的应用。

第 8 章　手机网络应用

本章通过 16 个实例介绍了手机网络组件的应用，有自定义手机浏览器、网络音乐的播放，以及网络下载数据等。

第 9 章　手机的 Google 服务功能

本章通过 6 个实例介绍了 Google 服务功能的实现，有手机端的 Google 账号的登录、制作成绩柱状图，以及 Google 地图的实现等。

第 10 章　手机多媒体服务功能

本章通过 14 个实例为读者介绍了手机的多媒体服务功能，包括音频、视频的采集，以及 2D 游戏的开发。

第 11 章　Android 手机的 3D 世界

本章通过 8 个实例为读者介绍了 3D 的一部分基础技术，每一项技术通过深入剖析其原理，并通过实例对其进行了详细的讲解，相信读者一定会受益匪浅。

第 12 章　手机特效开发

本章通过 12 个实例向读者介绍了 Android 中手机特效的开发，在 Android 平台下开发，可能会使用到其中的一些特效，希望读者可以灵活地运用本章中的手机特效。

第 13 章　休闲游戏——Q 版疯狂大炮

本章通过 30 个实例介绍了 Android 2D 游戏《Q 版疯狂大炮》的开发，主要介绍了 2D 的贴图技术，以及碰撞检测的开发。

第 14 章　娱乐游戏——3D 迷宫

本章通过 13 个实例介绍了 Android 3D 游戏《3D 迷宫》的开发，主要介绍了大范围贴图技术和地图设计器的应用。

本书面向的读者

- ❑ 初学 Android 的人员
- ❑ 有一定的 Android 基础并且希望学习 Android 高级开发技术的读者
- ❑ 在职的 Android 开发人员
- ❑ 在校相关专业大学生

目　录

第 1 章　Android 简介

Android 一词的本义是指“机器人”，同时也是 Google 于 2007 年 11 月 5 日宣布的基于 Linux 平台的开源手机操作系统的名称，该平台由操作系统、中间件、用户界面和应用软件组成，号称是首个为移动终端打造的真正开放和完整的移动软件。

1.1　Android 的诞生

Android 最初是由 Andy Rubin 创造的，其最初的目标是把 Android 打造成一个可以对任何软件设计人员开放的移动终端平台。很快 Android 就获得了很多软件厂商的青睐，表示要买下它。几周后，Google 就抢先买下了 Android。

Google 收购 Android 的时候没有宣布任何计划，直到 2007 年 11 月 5 日，Google 终于揭开了谜底，宣布与其他 33 家手机制造商，包括摩托罗拉、华为、宏达电、三星、LG 等，手机芯片供货商、软硬件供货商、移动运营商联合组成开发手机联盟（Open Handset Alliance，OHA），并发布了名为 Android 的开放手机软件。

1.2　Android 的特点

Android 基于 Linux 技术，由操作系统、用户界面和应用程序组成，允许开发人员自由获取、修改源代码，也就是说，这是一套具有开源性质的手机终端解决方案。其特点如下所列：

- 开放性；
- 所有的应用程序都是平等的；
- 应用程序间无界限；
- 快速方便的应用程序开发。

下面来详细介绍一下上述四个特点。

（1）开放性

Android 是一个真正意义上的开放式移动开发平台，其同时包括底层操作系统以及上层的用户界面和应用程序（移动电话所需要的全部软件都囊括在内），而且不存在任何以往阻碍移动产业创新的专有权障碍。

（2）应用程序平等

所有的 Android 应用程序之间是完全平等的，Android 平台被设计成由一系列应用程序所组成的平台。所有的应用程序都运行在虚拟机上面，虚拟机提供了一系列用于应用程序和硬件资源之间通信的 API。

抛开虚拟机，Android 所有的其他应用，包括系统的核心应用和第三方应用都是完全平等的。因此，用户甚至可以将系统中默认的电话拨号软件替换成其他第三方的电话拨号软件。用户也可以改变主界面的内容，或者将手机中任意的应用程序替换成所需要的其他应用程序。

（3）应用程序之间无界限

Android 打破了应用程序之间的界限，开发人员可以把 Internet 上的数据与本地的联系人、日历、位置信息结合起来，创造全新的用户体验。一个应用程序不但可以通过 API 访问系统提供的功能，还可以申明自身的功能供其他应用程序调用。

（4）快速方便的应用程序开发

Android 为开发人员提供了大量的使用库和工具，使得开发人员可以快速地创建自己的应用程序。例如，在其他平台的手机上要开发基于地图的应用是十分困难的，而 Android 将著名的 Google Map 集成进来，开发人员通过简单的几行代码就可以快速地开发出基于地图的应用。

1.3 Android 开发环境的搭建

本节将详细介绍一下 Android 开发环境的搭建，其开发环境的搭建主要包括如下几个步骤：

（1）安装 JDK，配置“AVA_HOME”环境变量；

（2）安装下载的 Android 开发包；

（3）安装 Ant 工具；

（4）将 Android 安装目录下的 tools 目录及 Ant 安装目录下的 bin 目录添加到 PATH 环境变量中。

经过上述几个步骤后，可以在 Android 安装目录下的 tools 目录中运行 android list targets，若能看到如图 1-1 所示的列表，则表示安装成功。

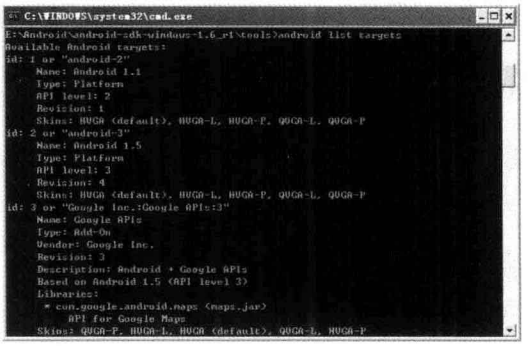


图 1-1　Android 安装列表

开发环境搭建成功后，还需要创建虚拟设备，用鼠标双击运行 tools 目录下的 Android.bat，在弹出的窗口界面中首先用鼠标单击右上侧的“New”按钮，如图 1-2 所示。

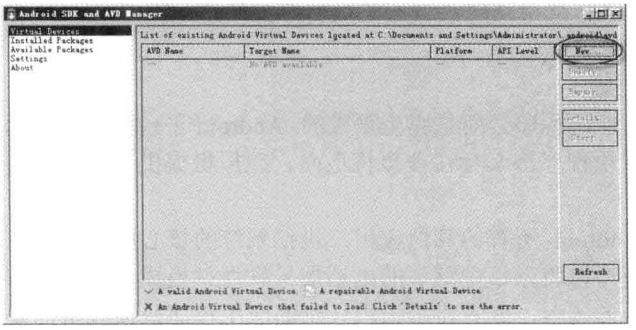


图 1-2　模拟器的创建

单击“New”按钮后，会弹出一个 AVD 设置窗口，如图 1-3 所示。在该窗口内对 AVD 进行相关的配置，最后单击“Create AVD”按钮。

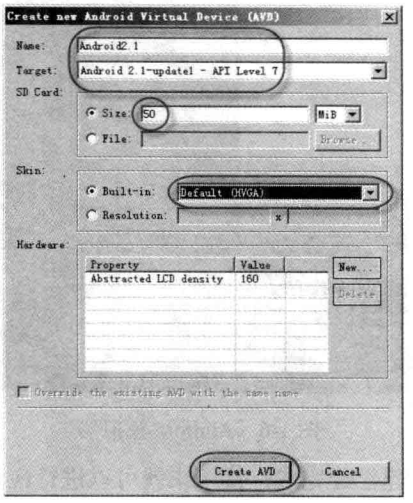


图 1-3　AVD 配置

最后单击弹出窗口中的“OK”按钮，即可完成虚拟机的设置，如图 1-4 所示。

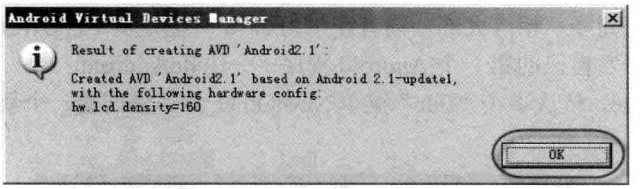


图 1-4　虚拟机的设置

成功创建虚拟机后，下一步要做的就是启动所创建的模拟器，首先选中所创建的模拟器，然后单击“Start”按钮，在弹出窗口中选择“Launch”按钮，就能够成功启动模拟器了，如图 1-5 所示。

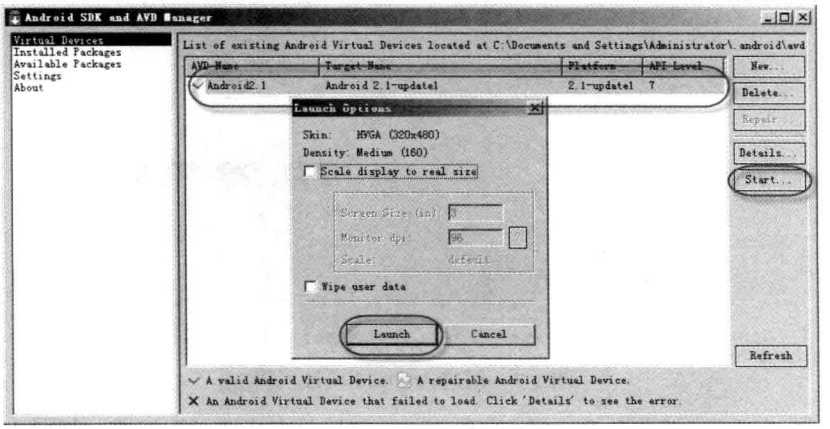


图 1-5　启动模拟器

下面要做的就是耐心等待模拟器的启动，模拟器成功启动后效果如图 1-6 所示。

图 1-6　Android 模拟器

到此，Android 环境的搭建已经完成，接下来就可以进行应用程序的开发了。

1.4　第一个 Android 程序——HelloAndroid

千里之行，始于足下。相信大部分读者在学习 Java 时也是从 HelloWorld 开始的。本节讲述如何在 Android 中开发自己的第一个 Android 程序——HelloAndroid。

首先打开 Eclipse，依次单击“File”菜单|New|Other 命令来创建一个新的 Android 项目，如图 1-7 所示。

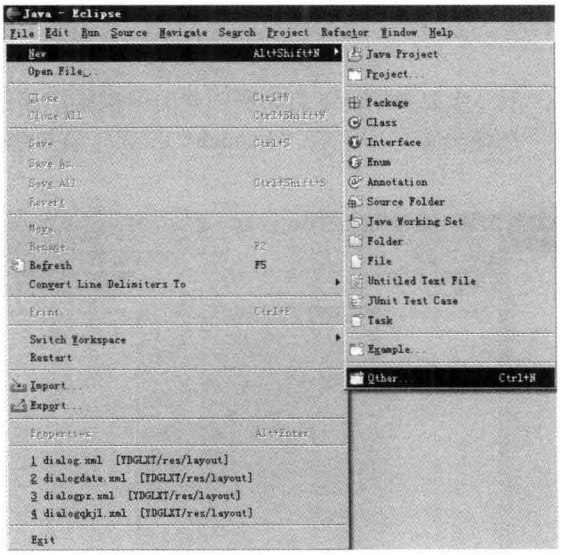


图 1-7　创建 Android 项目步骤 1

然后选择 Android 目录下的 Android Project，最后单击“Next”按钮，如图 1-8 所示。

随后为创建 HelloAndroid 程序的具体步骤，如下所示。

（1）在 Project name 中输入项目名称，这里输入的是 Sample1_1。

（2）在 Build Target 中选择项目的版本，本书中均使用 Android 2.1 版本。

（3）在 Application name 中输入程序的名称，这里输入的是 HelloAndroid。

（4）在 Package name 中输入程序的包名，这里输入的是 com.bn.chap1.ha。

选中 Create Activity 让系统帮我们创建一个 Acitvity，输入 Activity 名称，这里输入的是 Sample1_1_Activity，最后单击“Finish”按钮，如图 1-9 所示。

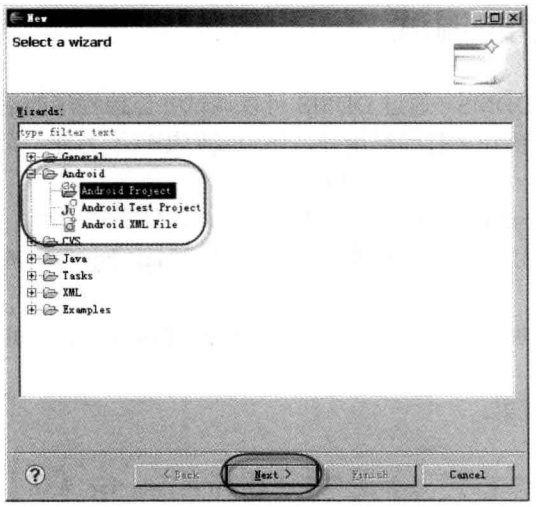


图 1-8 创建 Android 项目步骤 2

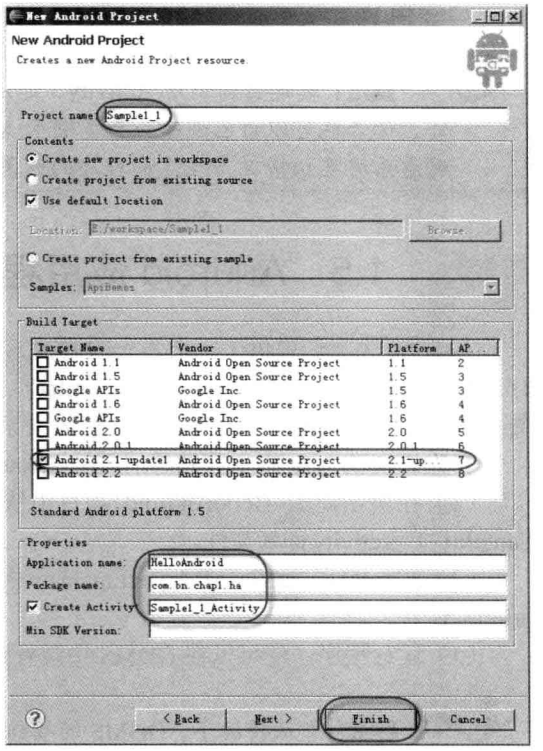


图 1-9 HelloAndroid 应用程序的创建

提示：Package name 中输入的包名，必须是两级以上的包名。

在 Project Explorer 中会自动增加一个项目包。项目包中每个文件夹及其文件的作用在下面给出一一分析。这里还是先继续 Sample1_1 程序。

（1）首先展开项目包，找到 values 文件夹，打开其中的 strings.xml 文件，如图 1-10 所示，添加字符串资源。

（2）在 strings.xml 左下角有两个视图，Resources 与 strings.xml，前者为所见即所得，后者是通过 xml 文件添加字符串资源，这里将介绍后者。

（3）将<string name="hello">Hello World,Sample1_1_MyActivity!</string>一行删除，该行是默认情况下 Acitvity 中显示的字符串信息。然后在</resources>标记前添加代码<string name="hello"> Hello AndroidWorld, My name is silver! </string>。

图 1-10 strings.xml 位置

（4）单击“运行”按钮，如图 1-11 所示，运行我们的第

一个 HelloAndroid 程序。

（5）运行效果如图 1-12 所示。

图 1-11　运行按钮

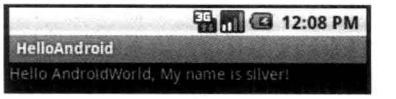


图 1-12　HelloAndroid 运行效果图

提示：选择自动创建 Acitvity，是让开发环境帮我们创建，省去了一些不必要的工作，读者也可以自己创建一个类继承 Activity。在项目生成后，读者可自己运行一下，观看系统默认情况下生成的程序的效果。

1.5　Android 应用程序的调试

Android 为我们提供了一个强大的调试工具 DDMS，通过 DDMS 可以调试并监控程序的运行。Android 除了可以用 System.out.println("…");语句来打印输出外，还为我们提供了另外一个类 android.util.Log 来调试程序。下面就为大家一一讲解两种方法的使用。

首先讲解在 Java 中十分熟悉的 System.out.println()方法，进行打印输出，具体步骤如下。

（1）在所创建的 HelloAndroid 项目中依次打开 src 文件夹下的 com.bn.chap1.ha 包中的 Sample1_1_Activity.java 文件。

（2）在 setContentView(R.layout.main)下面添加一句代码 System.out.println("first System.out.println");

（3）运行程序，便可以在 DDMS 中观看到效果。

提示：默认情况下，DDMS 的按钮是隐藏的，读者需要通过 Open Perstective 来添加 DDMS。在工具栏中，通常 Open Perstective 在开发环境的右上角，如图 1-13 所示。

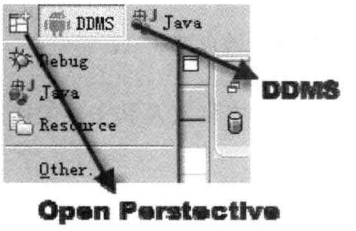


图 1-13　Open Perstective 和 DDMS 按钮

图 1-14　DEBUG 按钮

（4）打开 DDMS，找到 LogCat 面板，选择 DEBUG 按钮，如图 1-14 所示。选择 Log，可以找到刚刚打印的语句，如图 1-15 所示。

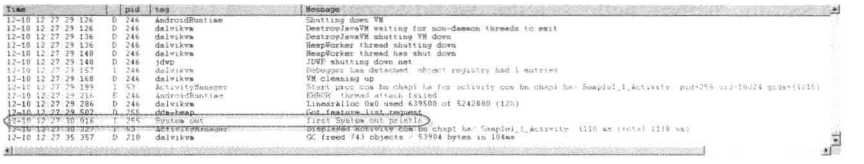


图 1-15　Log 面板打印输出

（5）如果读者觉得在 Log 中有太多信息，也可以在 LogCat 中添加一个专门输出 System.out 信息的面板。单击“DEBUG”按钮后面的加号，然后会弹出 Log Filter 面板，在 Filter Name 中输入过滤器名称，在 by Log Tag 中输入过滤的内容，如图 1-16 所示。

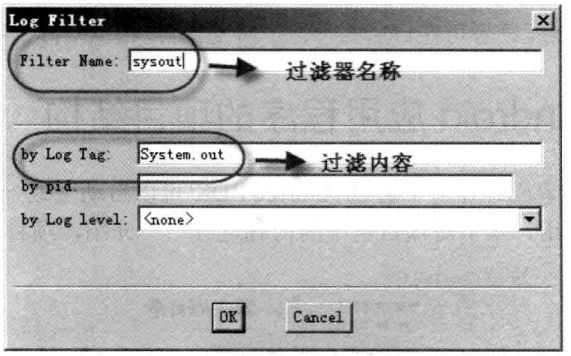


图 1-16　过滤器设置

提示：在过滤器内容一项中，程序是用 System.out.println()输出的，但有时可能用 System.out.print()输出，所以此处笔者设置为 System.out。

（6）此时可以再次运行，观看效果，如图 1-17 所示。

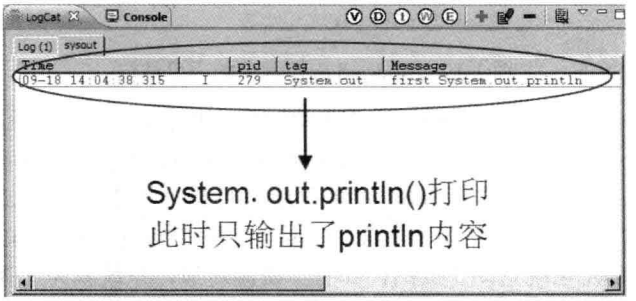


图 1-17　System.out 输出面板

当然除了熟悉的 System.out 外，Android 还提供了另外一个类 android.util.Log 进行打印输出。下面介绍 Log 类的使用，步骤如下。

（1）在刚才的项目中注释掉刚刚添加的 System.out.println("first System.out.println")，然后在后面添加代码 Log.d("TAG", "first Log")。

（2）运行程序，在 DDMS 中找到 LogCat 面板，选择 Log 页面，观看打印的内容，如图 1-18 所示。

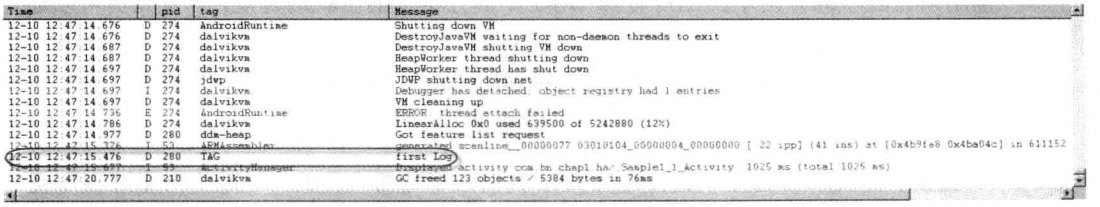


图 1-18　Log 输出内容

提示：public static int d (String tag, String msg)方法为打印输出，tag 为输出的标签，mag 为输出的信息内容。使用该方法还需要导入 android.util.Log 包。

两种输出各有优缺点，读者可在日后的学习中慢慢体会，根据实际情况选择相应的打印语句。

1.6 Android 应用程序的项目结构

从上面的项目中读者已经知道了通过 Android SDK 可以自动生成一个项目包，但是没有对项目包里的内容进行介绍，本节对项目包中的内容进行一一介绍，项目包如图 1-19 所示。

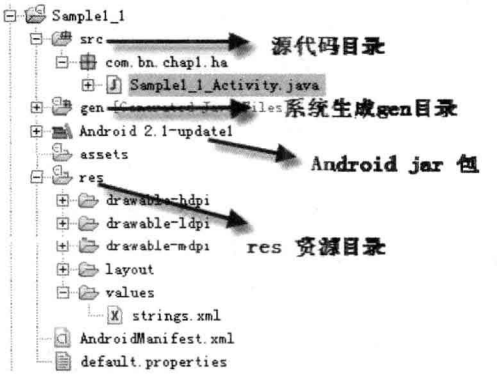


图 1-19　Android 应用程序的项目结构

src 源代码目录：该目录存放 Android 应用程序所有的源代码。该目录项有不同的包，包中对应开发的源程序，开发人员开发的主要精力都是在开发 src 目录下的内容。

gen 文件夹目录：该目录存放了 Eclipse 的 ADT 插件自动生成的 R.java 文件。实际上 R.java 定义了一个 R 类，它包含了应用中用户界面、图像、字符串等各种资源与之相对应的资源编号（id），这些资源编号都是系统自动生成的，即有一资源对象，系统就为此在 R 类中生成相应的资源编号，好比一本字典。

提示：Android 应用程序中通过 R 类别来实现对资源的应用。同时，编译器也会查看这个资源列表，没有使用到的资源就不会编译进去，为手机应用程序节省空间。

res 资源目录：该目录下定义了 drawable、layout 及 values 三个目录，下面对这三个目录一一进行介绍。

- drawable 目录：该目录下有 drawable-hdpi,drawable-ldpi,drawable-mdpi 三个文件夹，分别用来存放不同分辨率的图片资源，用于不同分辨率的手机的开发，开发人员可以通过 Resource.getDrawable(id)获得该资源。
- layout 目录：该目录下包含了所有使用 XML 格式的界面描述文件。主要用于表述应用程序的用户界面布局，也用于描述用户界面和接口组件。

提示：开发人员也可以直接通过 Java 代码来创建用户界面，不过使用 xml 描述文件则更简单，架构更清晰，维护也更容易。在这里需要强调的是，如果在程序里面要使用这些用户界面组件，必须通过前面提到的 R 类来调用。

- values 目录：该目录包含了使用 XML 格式的参数描述文件，读者可以在此添加一些额外的资源如字符串（string.xml）、颜色（color.xml）、样式（style.xml）和数组（arrays.xml）等。主要用于在代码中通过 R 类来调用它们，而不直接使用，这样就可将代码和资源分开管理，便于维护。

除了以上文件夹外，还有一个 AndroidManifest.xml 文件，该文件系统的控制文件，告诉系统如何处理创建的所有顶层组件，尤其是 Activity、IntentReceiver、Service 及 ContentProvider，凡是需要用到的组件都要在此注册，同时该文件也是所有 Android 应用程序都需要的文件，其描述了程序包的全局变量，包括公开的应用程序组件和为每个组件的实现类，什么样的数据可以操作、在什么地方可以运行等。

这个文件中最重要的一个内容就是 Intent 过滤器，这些过滤器描述了何时在何种情况下让 Activity 启动。当一个 Activity（或操作系统）想要执行一个动作，它将创建一个 Intent 对象。该对象包含了很多描述符，描述了想做的操作，如处理的数据、数据的类型，以及一些其他的信息。Android 将 Intent 对象中的信息与所有公开的 Intent 过滤器比较，找到一个最能恰当处理请求的数据和动作的 Activity。

提示：上述功能以外，AndroidManifest.xml 文件中也可以指定权限、安全控制及测试，这些功能在以后的开发中会经常用到。

1.7　Android 的系统架构

从软件分层的角度看，Android 平台由应用程序、应用程序框架、Android 运行时、系统库以及 Linux 内核构成，如图 1-20 所示。

图 1-20　Android 系统框架

通过图 1-20，读者能够大致了解 Android 的系统框架，下面就对其进行一一讲解。

（1）Android 应用程序框架

该框架是进行 Android 开发的基础，开发人员的大部分时间也是在这一层进行的，应用程

序框架包含了视图系统、活动管理器、通知管理器、内容提供者、窗口管理器、位置管理器、资源管理器、电话管理器和包管理器共 9 大部分，如图 1-21 所示。

图 1-21　应用程序框架

应用程序框架的各部分具体功能如下所列。

- 活动管理器（Activity Manager）：管理所有的应用程序生命周期，以及通常的导航返回栈。
- 窗口管理器（Window Manager）：管理所有的窗口程序。
- 内容提供者（Content Provider）：用来让应用程序之间互相存取/分享数据。例如，某个应用程序可以存取联系人应用程序内的联系人数据。
- 视图系统（View System）：用来构建应用程序的基本组件，包含列表、网格、文本框和按钮等。
- 包管理器（Package Manager）：用来进行 Android 系统内的程序管理。
- 电话管理器（Telephone Manager）：用来管理所有的移动设备的功能。
- 位置管理器（Location Manager）：用来提供位置服务。
- 资源管理器（Resource Manager）：提供各种资源让应用程序去使用，如本地化字符串、图片、布局文件。
- 通知管理器（Notification Manager）：是应用程序可以在状态栏应用程序中显示警告信息。状态栏通常在手机的顶部，如短信、语音邮件提示就会在这里出现。

提示：Android 平台的应用程序平等和应用程序无界限等特性就是由应用程序框架来保证的。

（2）Android 运行时

Android 运行时包括核心库和 Dalvik 虚拟机两部分，如图 1-22 所示，接下来将一一介绍这两部分。

图 1-22　Android 运行时

核心库：包括两大部分，一部分由 Java 所需调用的功能函数组成，另一部分由 Android 的核心库如 android.os、android.net、android.media 等组成。

提示：与传统的 Java 程序不同的是，每一个 Android 应用程序都有一个自有的进程，每一个 Android 应用程序都是由一个自有的 Dalvik 虚拟机来执行的。

Dalvik 虚拟机：是一种基于寄存器的 Java 虚拟机，其依靠转换工具 dx 将 Java 字节码转换为 dex 格式（称为.dex）。基于寄存器的虚拟机相对于基于栈的虚拟机的优点是，其所需资源相对较少，而且硬件实现虚拟机也会比较容易。

（3）系统库

应用程序框架是贴近于应用程序的软件组件服务，而更底层则是 Android 的函数库，这一部分是应用程序框架的支撑，其架构如图 1-23 所示。架构中各部分的具体功能如下所列。

图 1-23 系统库

- 媒体函数库：是以 Packet Video 公司的 OpenCORE 为基础所发展而成的，使用这个函数库在播放、录制多种常见的影音格式时，非常方便。
- SurfaceManager：负责合成 2D 与 3D 绘图之间软件的合成。
- WebKit：这是一套网页浏览器的软件引擎。WebKit 可以为 Android 内部自带的浏览器所调用，WebKit 是一个开源项目，许多浏览器也都是用 WebKit 引擎开发的，如 Apple 的 Safari、Nokia s60 手机的浏览器等。
- SGL：提供 Android 在 2D 绘图方面的绘图引擎。
- OpenGLES：Android 是依据 OpenGL ES1.0API 标准来实现的 3D 绘图函数库，该函数库可以用软件方式执行也可以用硬件加速方式执行，其中 3D 软件在光栅处理方面进行了高度优化。
- FreeType：该库提供位图、向量字的绘图显示。
- 媒体框架：提供了对各种音频、视频的支持。Android 支持多种音频、视频、静态图像格式。
- SQLite：这是一套轻量级的数据库引擎，支持关系数据库的事务等。
- Libc：提供了一套 C 库。

（4）Linux 内核

采用 Linux 内核（如图 1-24 所示）是 Android 平台开放性的基础。Android 平台中的操作系统采用了 Linux 2.6 版的内核，它包括了显示驱动、摄像头驱动、Flash 内存驱动、Binder（ipc）驱动、键盘驱动、Wifi 驱动、Audio 驱动及电源管理驱动。

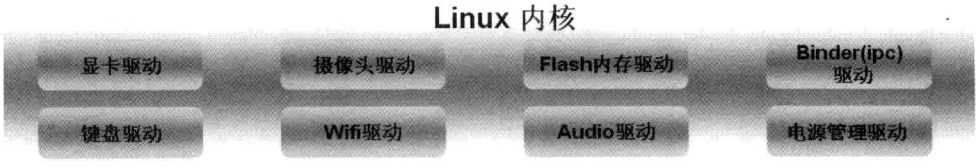


图 1-24 Linux 内核

Linux 内核层是软件与硬件之间的桥梁，软件开发者不必关心内核的底层实现，而只需将

精力全部投入到上层软件的开发中。而底层的工作都由 Google 和手机开发商来完成，如驱动的更新、新硬件驱动的编写等。

1.8 小结

本章主要介绍了 Android 的诞生及其特点，相信读者对 Android 已有所了解。本章中还介绍了 Android 开发环境的搭建以及在 Android 世界的第一个应用程序——HelloAndroid，通过该程序读者应该对 Android 应用程序的开发步骤有所了解。

接着为读者介绍的是 Android 应用程序的调试、项目结构和系统架构，这些同样能够帮助读者进一步了解 Android。对于理论性的知识，读者可能暂时不能彻底理解，这些知识读者只要先暂时有些概念，在以后的学习中一定会遇到，当结合实际例子之后，才能更进一步理解。

第 2 章　Android 简单控件的开发及应用

本章中，将主要向读者介绍 Android 平台下简单的控件以及 5 种布局的开发与应用，控件主要包括 Button、TextView、ImageView、ImageButton、EditText 简单控件。布局包括线性布局、相对布局、帧布局、表格布局和绝对布局。

实例 1　按钮的使用技巧

在本小节的程序中，主要向读者介绍 Button 按钮的应用。

【实例描述】

Button 按钮是最常见的控件，本程序主要是添加一个 Button 按钮，单击按钮即会弹出 Toast 提示。本实例的运行效果图，如图 2-1 所示。

图 2-1　Button 按钮

提示：该应用开始运行时首先进入的界面如图 2-1 所示，单击按钮，即会弹出 Toast 提示。

【实现过程】

本程序添加一个 Button 按钮，单击按钮可以弹出 Toast 提示。

【代码解析】

首先为读者介绍本程序主界面的 main.xml 的开发，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_1/res/layout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>          <!--版本号和编码方式-->
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="vertical"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent"
6      >                                            <!--LinearLayout 布局 -->
7      <Button
8           android:text="@string/button1"
9           android:id="@+id/Button01"
10          android:layout_width="wrap_content"
11          android:layout_height="wrap_content">   <!--自适应大小-->
12     </Button>                                    <!--Button 按钮的 id -->
13  </LinearLayout>
```

提示：上面的 xml 文件为本项目的主界面的文件。

上面已经介绍了本程序的主界面 main.xml 的开发，接下来将为读者介绍本程序具体功能的实现，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_1/src/com/bn/es2a 目录下的 Sample2_1_Activity。

```
1   package com.bn.es2a;                                    //声明包
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码
3   import android.widget.*;                                //导入相关类
4   public class Sample2_1_Activity extends Activity{       //创建继承 Activity 的类
5      @Override
6      public void onCreate(Bundle savedInstanceState){ //重写的方法
7         super.onCreate(savedInstanceState);              //调用父类
8         setContentView(R.layout.main);                   //跳转到主界面
9         final Button button = (Button) findViewById(R.id.Button01);
                                                           //获得 Button 引用
10        button.setOnClickListener(                       //OnClickListener 事件
11          new OnClickListener(){
12                public void onClick(View v){             //重写的 onClick 方法
13                  Toast.makeText(Sample2_1_MyActivity.this,    //弹出 Toast
14                                 "按钮单击",
15                                 Toast.LENGTH_SHORT).show();
16} } ); } }
```

其中：

- 第 8 行表示的是跳转到主界面。
- 第 9～16 行表示的是对按钮的监听，单击按钮弹出 Toast 提示。

实例 2　最常用的线性布局

在本小节中，通过 LinearLayout 线性布局的应用，构建了一个小型的计算器界面。该计算器实现了整数间的加减乘除四则运算，并通过本计算器的实现，向读者介绍 LinearLayout 线性布局的具体应用。

【实例描述】

在本小节中实现了小型计算器软件的开发。该计算器通过 0～9 这 10 个数字按钮，“加”、“减”、“乘”、“除”和“等于”5 个运算按钮，以及清空按钮来实现对本程序的操控，并通过一个文本框来显示计算的结果。应用本程序可以进行整数间简单的加、减、乘、除四则运算。

本实例的运行效果图，如图 2-2 所示。

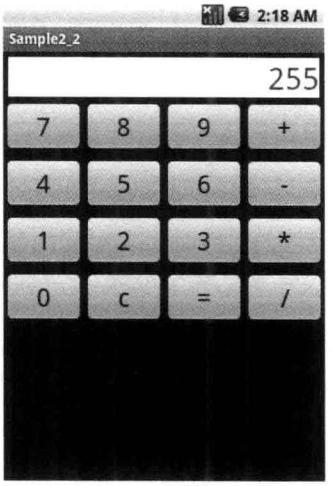


图 2-2　小型计算器

提示：使用者可以通过单击数字输入第一个计算值，然后单击所需计算按钮确定计算方式，再通过单击数字输入第二个计算值，之后当再次单击“等于”按钮时，软件会自动计算出结果数值，如图 2-2 所示。

【实现过程】

小型计算器的开发，主要运用了 LinearLayout 线性布局的相关知识。线性布局是最简单的布局之一，其提供了控件水平或者垂直排列的模型。同时，使用此布局时可以通过设置控件的 weight 参数控制各个控件在容器中的相对大小。

提示：线性布局垂直时占用一列，水平时占用一行。特别要注意的是，水平时若超过一行则不会像 SE 中的 FlowLayout 一样自动转行。

线性布局中可以使用 gravity 属性设置控件的对齐方式，详情如表 2.1 所列。

表 2.1　线性布局 gravity 属性表

属性名称	属性说明
Top	不改变控件大小，对齐到容器顶部
Bottom	不改变控件大小，对齐到容器底部
Left	不改变控件大小，对齐到容器左侧
Right	不改变控件大小，对齐到容器右侧
center_vertical	不改变控件大小，对齐到容器纵向中央位置

续表

属性名称	属性说明
center_horizontal	不改变控件大小，对齐到容器横向中央位置
Center	不改变控件大小，对齐到容器中央位置
fill_vertical	若有可能，纵向拉伸以填满容器
fill_horizontal	若有可能，横向拉伸以填满容器
Fill	若有可能，纵向横向同时拉伸以填满容器

提示：当需要使用多个属性时，用“|”分隔开即可。

【代码解析】

首先为读者介绍的是小型计算器的主界面 mian.xml 的开发，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_2/ res/layout 目录下的 main.xml。

```
<?xml version="1.0" encoding="utf-8"?>                    <!--版本号和编码方式-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical" android:layout_width="fill_parent"
   android:layout_height="fill_parent" android:paddingTop="5dip">
   <TextView
    android:id="@+id/tv" android:layout_width="fill_parent"
    android:layout_height="40dip" android:layout_marginRight="5dip"
    android:layout_marginLeft="5dip" android:background="#FFFFFF"
    android:gravity="center_vertical|right" android:textSize="30dip"
    android:textColor="#ff0000">                          <!--设置颜色-->
   </TextView>
   <LinearLayout
    android:orientation="horizontal" android:layout_width="fill_parent"
    android:layout_height="wrap_content" android:paddingTop="5dip">
                                                          <!--距上面的距离-->
      <Button
       android:text="7"  android:id="@+id/Button07"
       android:layout_width="80dip" android:layout_height="wrap_content"/>
      <Button
       android:text="8"  android:id="@+id/Button08"
       android:layout_width="80dip" android:layout_height="wrap_content"/>
      <Button
       android:text="9"  android:id="@+id/Button09"
       android:layout_width="80dip" android:layout_height="wrap_content"/>
      <Button
       android:text="+"  android:id="@+id/ButtonJia"
       android:layout_width="80dip" android:layout_height="wrap_content"/>
   </LinearLayout>
   <LinearLayout
    android:orientation="horizontal" android:layout_width="fill_parent"
    android:layout_height="wrap_content" android:paddingTop="5dip">
      <Button
       android:text="4"  android:id="@+id/Button04"
       android:layout_width="80dip" android:layout_height="wrap_content"/>
      <Button
       android:text="5"  android:id="@+id/Button05"
       android:layout_width="80dip" android:layout_height="wrap_content"/>
      <Button
       android:text="6"  android:id="@+id/Button06"
       android:layout_width="80dip" android:layout_height="wrap_content"/>
      <Button
       android:text="-"  android:id="@+id/ButtonJian"
```

```
        android:layout_width="80dip" android:layout_height="wrap_content"/>
    </LinearLayout>
                                                    <!--LinearLayout 布局-->
    <LinearLayout
     android:orientation="horizontal" android:layout_width="fill_parent"
     android:layout_height="wrap_content" android:paddingTop="5dip">
        <Button
         android:text="1"  android:id="@+id/Button01"
         android:layout_width="80dip" android:layout_height="wrap_content"/>
       <Button
         android:text="2"   android:id="@+id/Button02"  android:layout_width
="80dip"
         android:layout_height="wrap_content"/>
        <Button
         android:text="3"  android:id="@+id/Button03"
         android:layout_width="80dip" android:layout_height="wrap_content"/>
        <Button
         android:text="*"  android:id="@+id/ButtonCheng"
         android:layout_width="80dip" android:layout_height="wrap_content"/>
    </LinearLayout>                                    <!--LinearLayout 布局-->
    <LinearLayout
     android:orientation="horizontal" android:layout_width="fill_parent"
     android:layout_height="wrap_content" android:paddingTop="5dip">
        <Button
         android:text="0"  android:id="@+id/Button00"
         android:layout_width="80dip" android:layout_height="wrap_content"/>
       <Button
         android:text="c"  android:id="@+id/ButtonC"
         android:layout_width="80dip" android:layout_height="wrap_content"/>
        <Button
         android:text="="  android:id="@+id/ButtonDengyu"
         android:layout_width="80dip" android:layout_height="wrap_content"/>
        <Button
         android:text="/"  android:id="@+id/ButtonChu"
         android:layout_width="80dip" android:layout_height="wrap_content"/>
    </LinearLayout>                                    <!--LinearLayout 布局-->
</LinearLayout>
```

提示：在该 xml 文件中实现了布局的设置。总的 LinearLayout 为垂直排列模式，其中垂直排列了四个子 LinearLayout，并且每个子 LinearLayout 为水平排列模式，在每个子 LinearLayout 中水平排列了 4 个 Button 按钮控件。通过对各个控件的属性设置，实现了小型计算器的界面效果。

上面已经介绍了本程序主界面 main.xml 的开发，接下来将要为读者介绍该小型计算器的功能的实现，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_2/src/com/bn/es2b 目录下的 Sample2_2_Activity.class。

```
package com.bn.es2b;                                    //声明包
......//该处省略了部分类的导入，读者可自行查看随书光盘中源代码
import android.view.View.OnClickListener;               //导入相关类
public class Sample2_2_Activity extends Activity{       //创建继承 Activity 的类
    ......//该处省略了成员变量的代码，读者可自行查看随书光盘中源代码
    public void onCreate(Bundle savedInstanceState){    //继承必须重写的方法
      super.onCreate(savedInstanceState);               //调用父类
      setContentView(R.layout.main);                    //跳转到 main 界面
      initButton();                                     //初始化控件
```

```
        buttonC.setOnClickListener(                          //清空按钮的单击事件监听器
            new OnClickListener(){
                public void onClick(View v){                 //重写的 onClick 方法
                    str1="";str2="";                         //设定 str1 与 str2 的值
                    tv.setText(str1);                        //清空记录
                    flag=0;                                  //标志为 0
          } } );
        for(int i=0;i<buttons.length;i++){                   //for 循环
          temp=(Button)findViewById(buttons[i]);             //初始化按钮
          temp.setOnClickListener(                           //为 Button 添加监听器
              new OnClickListener(){                         //内部类
                  @Override
                  public void onClick(View v){               //重写的方法
                      str1=tv.getText().toString().trim();
                                                             //得到 TextView 中的字符串

str1=str1+String.valueOf(((Button)v).getText());//获得新输入的值
                          tv.setText(str1);                  //设置 TextView
          } } ); }
         buttonListener(buttonJia,1);                        //加号的监听器
         buttonListener(buttonJian,2);                       //减号的监听器
         buttonListener(buttonCheng,3);                      //乘号的监听器
         buttonListener(buttonChu,4);                        //除法的监听器
        buttonDengyu.setOnClickListener(                     //等号的监听器
          new OnClickListener(){                             //内部类
              @Override
              public void onClick(View v){                   //重写的方法
                 result1=Integer.parseInt(str1);             //字符串转换为 int 类型
                  if(flag==1){                               //判断 flag 是否为 1
                  result=result0+result1;                    //实现加法
                  System.out.println(result0+":"+result1);
                }else if(flag==2){                           //判断 flag 是否为 2
                  result=result0-result1;                    //实现减法
                }else if(flag==3){                           //判断 flag 是否为 3
                  result=result0*result1;                    //实现乘法
                }else if(flag==4){                           //判断 flag 是否为 4
                  result=(int)(result0/result1);             //实现除法
                }
                  String str=(result+"").trim();             //运算结果转换为字符串
                  tv.setText(str);                           //设置 TextView 的显示值
  } } ); }
    public void initButton(){                                //初始化控件资源
     ......//初始化控件的方法，读者可自行查看随书光盘中源代码
    }
    public void buttonListener(Button button,final int id){
                                                             //对按钮实现监听的方法
     button.setOnClickListener(                              //设置监听
         new OnClickListener(){
             public void onClick(View v){                    //重写 onClick 方法
                 String str=tv.getText().toString().trim();
                                                             //得到 TextView 的字符串
                 result0=Integer.parseInt(str);              //转换为整型
                 tv.setText("");                             //清空
                 flag=id;                                    //设定 flag 的值
  } } ); } }
```

其中：

- 第 10～16 行表示的是对计算器清空按钮的监听，并且实现清空功能。
- 第 17～26 行表示的是对计算器九个值的按键进行监听。
- 第 31～48 行表示的是对计算器等号的监听，并且实现基本的四则运算。
- 第 52～60 行表示的是对计算器运算符按钮的监听。

实例 3　相对性布局的方法

在本小节中，通过 RelativeLayout 相对布局的应用，构建了一个为图片添加说明的小程序的界面。本程序可以在图片的上、下、左、右为图片添加说明信息。通过本程序的实现，读者可以了解到 RelativeLayout 相对布局的具体应用。

【实例描述】

在本小节中实现了对图片不同位置添加说明信息的小程序，通过单击位于主界面上、下、左、右的四个按钮，可以在图片对应的位置添加文本编辑框，进而为图片输入说明信息。本实例的运行效果图如图 2-3 所示。

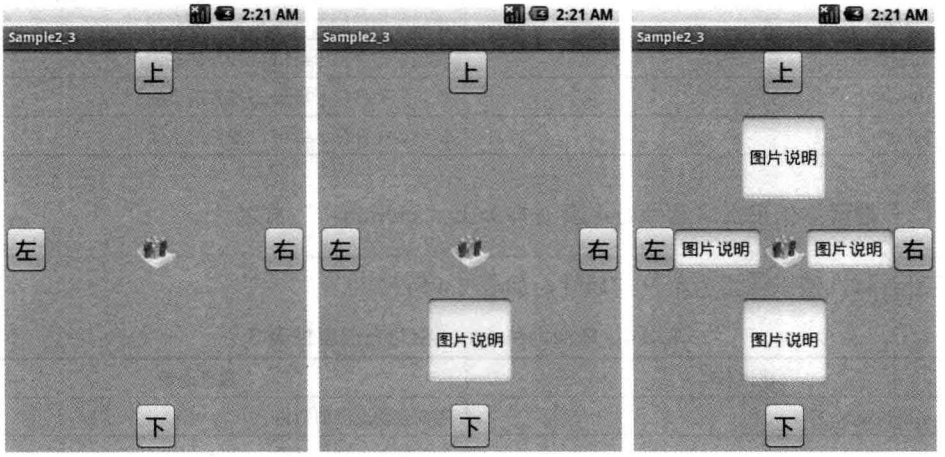


图 2-3　为图片添加说明小软件截图

提示：本程序的功能是为图片添加图片说明，可以在图片的上侧、下侧、左侧、右侧添加说明信息。如单击“下”按钮，则会在图片下方弹出文本编辑框，用于添加图片说明信息，如图 2-3 所示。

【实现过程】

该小程序的开发主要运用了 RelativeLayout 相对布局的相关知识。相对布局的容器中控件会根据所设置的参照控件和参数进行相对布局。参照控件可以是上层容器，也可以是本层的兄弟控件。但要注意的是，被参照的控件必须要在参照其的控件前定义。

进行相对布局时可能用到的属性有很多，但都不复杂，首先看属性值为 true 或 false 的属性，如表 2.2 所列。

表 2.2　RelativeLayout 相对布局属性表 1

属性名称	属性说明
layout_centerHorizontal	当前控件位于父控件的横向中间位置
layout_centerVertical	当前控件位于父控件的纵向中间位置
layout_centerInParent	当前控件位于父控件的中央位置
layout_alignParentBotton	当前控件低端与父控件低端对齐
layout_alignParentLeft	当前控件左侧与父控件左侧对齐
layout_alignParentRight	当前控件右侧与父控件右侧对齐
layout_alignParentTop	当前控件顶端与父控件顶端对齐
layout_alignWithParentIfMissing	参照控件不存在或不可见时参照父控件

提示：使用上述属性时全部都要加上“android：”前缀。

接着再看属性值为其他控件 id 的属性，如表 2.3 所列。

表 2.3　RelativeLayout 相对布局属性表 2

属性名称	属性说明
layout_toRightOf	使当前控件位于给出 id 控件的右侧
layout_toLeftOf	使当前控件位于给出 id 控件的左侧
layout_above	使当前控件位于给出 id 控件的上面
layout_below	使当前控件位于给出 id 控件的下面

提示：使用上述属性时也要全部加上“android：”前缀。

最后看属性值以像素为单位的属性，如表 2.4 所列。

表 2.4　RelativeLayout 相对布局属性表 3

属性名称	属性说明
layout_marginLeft	当前控件左侧的留白
layout_marginRight	当前控件右侧的留白
layout_marginTop	当前控件上方的留白
layout_marginBottom	当前控件下方的留白

提示：使用上述属性时也要全部加上“android：”前缀。

【代码解析】

首先为读者介绍本程序主界面 main.xml 的开发，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_3/ res/layout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>                <!--版本号和编码方式-->
2   <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3       android:id="@+id/rl"
4       android:layout_width="fill_parent"
5       android:layout_height="fill_parent"
6       android:background="#edab4a"
```

```
    >                                                    <!--RelativeLayout 布局-->
     <Button
         android:text="上"
         android:id="@+id/Shang"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_alignParentTop="true"
         android:layout_centerHorizontal="true"
    >                                                    <!--Button 控件-->
    </Button>
    <Button
         android:text="下"
         android:id="@+id/Xia"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_alignParentBottom="true"
         android:layout_centerHorizontal="true"
    >                                                    <!--Button 控件-->
    </Button>
    <Button
         android:text="左"
         android:id="@+id/Zuo"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_alignParentLeft="true"
         android:layout_centerVertical="true"
    >
    </Button>                                            <!--Button 控件-->
    <Button
         android:text="右"
         android:id="@+id/You"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_alignParentRight="true"
         android:layout_centerVertical="true"
    >
    </Button>                                            <!--Button 控件-->
    <ImageView
         android:src="@drawable/fengjing"
         android:id="@+id/Start"
         android:layout_width="60dip"
         android:layout_height="100dip"
         android:layout_centerInParent="true"
    >
    </ImageView>                                         <!--ImageView-->
</RelativeLayout>
```

提示：在该 xml 文件中实现了该程序的布局。总的布局为 RelativeLayout 布局，其中排列了四个 Button 按钮和一个 ImageView。通过对各个控件的属性设置，实现了本计算器软件的界面效果。

上面已经介绍了本程序主界面 main.xml 的开发，接下来为读者介绍本程序具体功能的实现，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_3/src/com/bn/ex2c 目录下的 Sample2_3_Activity.class。

```
package com.bn. ex2c;                                    //包名
import android.app.Activity;                             //导入相关类
```

```
3   public class Sample2_3_Activity extends Activity {
4       RelativeLayout rl;                                    //相对布局对象
5       Button shang;                                         //上面的按钮
6       Button xia;                                           //下面的按钮
7       Button zuo;                                           //左侧按钮
8       Button you;                                           //右侧按钮
9       ImageView currButton;                                 //记录当前 ImageView
10      ImageView start;                                      //ImageView 开启
11     public void onCreate(Bundle savedInstanceState) {
12        super.onCreate(savedInstanceState);                 //调用父类
13        setContentView(R.layout.main);                      //界面跳转到主界面
14        rl=(RelativeLayout)this.findViewById(R.id.rl);//获取布局对象
15        shang=(Button)this.findViewById(R.id.Shang);  //获取当前控件
16        xia=(Button)this.findViewById(R.id.Xia);      //下面的按钮
17        zuo=(Button)this.findViewById(R.id.Zuo);      //左侧的按钮
18        you=(Button)this.findViewById(R.id.You);      //右侧的按钮
19        start=(ImageView)this.findViewById(R.id.Start);    //中间的图片
20        currButton=start;                                   //规定当前 ImageView
21        shang.setOnClickListener(                           //监听
22          new OnClickListener(){
23                  public void onClick(View v) {
24                  EditText temp=new EditText(Sample2_3_Activity.this);
                                                              //添加新的 EditText
25                  temp.setText("图片说明");                   //设置信息
26                  RelativeLayout.LayoutParams lp1 =  //设置控件位置
27              new   RelativeLayout.LayoutParams(ViewGroup.LayoutParams.WRAP_
CONTENT, 95);
28                    lp1.addRule(RelativeLayout.ABOVE,currButton.getId());
29                   lp1.addRule(RelativeLayout.CENTER_HORIZONTAL, currButton.
getId());
30                    rl.addView(temp, lp1);                    //将控件添加到布局中
31                  }});
32        xia.setOnClickListener(                             //设置监听
33          new OnClickListener(){
34                  public void onClick(View v) {       //重写 onClick 方法
35                      EditText temp=new EditText(Sample2_3_Activity.this);
                                                              //添加新的 EditText
36                      temp.setText("图片说明");
37                      RelativeLayout.LayoutParams lp1 =       //设置控件位置
38          new RelativeLayout.LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT,
95);
39                    lp1.addRule(RelativeLayout.BELOW,currButton.getId());
40                    lp1.addRule(RelativeLayout.CENTER_HORIZONTAL,
currButton.getId());
41                    rl.addView(temp, lp1);                    //将控件添加到布局中
42                  }});
43        zuo.setOnClickListener(                             //设置监听
44          new OnClickListener(){
45                  public void onClick(View v) {       //重写 onClick 方法
46                      EditText  temp=new  EditText(Sample2_3_Activity.this);
    //添加新的 EditText
47                      temp.setText("图片说明");          //设置为本框的文字
48                      RelativeLayout.LayoutParams lp1 =  //设置控件位置
49          new     RelativeLayout.LayoutParams(95,ViewGroup.LayoutParams.WRAP_
CONTENT);
50                    lp1.addRule(RelativeLayout.LEFT_OF,currButton.getId());
51                    lp1.addRule(RelativeLayout.CENTER_VERTICAL,  currButton.
getId());
```

```
52                      rl.addView(temp, lp1);   //将控件添加到布局中
53                    }});
54          you.setOnClickListener(       //单击“右”键时，在 currButton 下面添加新控件
55            new OnClickListener(){
56                    public void onClick(View v) {
57                        EditText temp=new EditText(Sample2_3_Activity.this);
                                  //添加新的 EditText
58                        temp.setText("图片说明");
59                        RelativeLayout.LayoutParams lp1 =  //设置控件位置
60                new      RelativeLayout.LayoutParams(95,ViewGroup.LayoutParams.
WRAP_CONTENT);
61                      lp1.addRule(RelativeLayout.RIGHT_OF,currButton.getId());
62                      lp1.addRule(RelativeLayout.CENTER_VERTICAL,
currButton.getId());
63                      rl.addView(temp, lp1);        //将控件添加到布局中
64                    }});
65      }}
```

其中：

- 第 4～10 行，为声明该类的成员变量。包括界面的控件对象和布局对象。
- 第 14～20 行，为获取界面控件和布局资源。
- 第 21～65 行，为上、下、左、右四个按钮添加监听器，当单击按钮时，在布局中的相应位置添加新的 EditText 控件，并设置该控件的属性。

实例 4　帧布局结构的学习

在本小节中，通过 FrameLayout 帧布局的应用，构建了一个饮食介绍的界面。并通过本程序的实现，来向读者介绍 FrameLayout 帧布局的具体应用。

【实例描述】

该小节构建了一个饮食介绍的小程序，在本程序中可以显示需要的信息。在网页的右下角有一个“退出”按钮，单击“退出”按钮，即可退出网页。本实例的运行效果图，如图 2-4 所示。

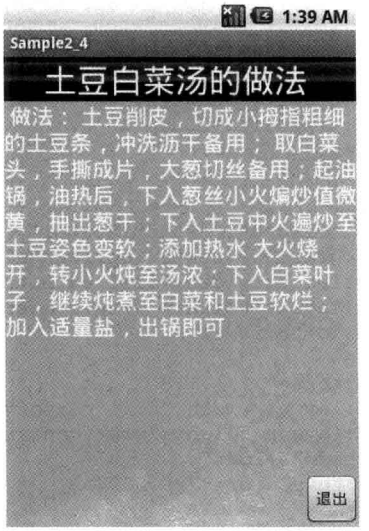


图 2-4　饮食介绍小网页

提示：本程序的功能是显示信息到主界面，并且可以单击“退出”按钮退出本程序，如图 2-4 所示。

【实现过程】

本程序的开发主要运用了 FrameLayout 帧布局的相关知识。帧布局是最简单的布局之一，采用帧布局的容器中无论放入多少个控件，默认情况下控件都对齐到容器的左上角。

如果控件一样大，同一时刻只能见到最上面的一个控件。

【代码解析】

首先为读者介绍本程序的主界面 main.xml 的开发，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_4/ res/layout 目录下的 main.xml。

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#edab4a"
    android:id="@+id/fl">
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:id="@+id/cc">
        <LinearLayout
            android:orientation="horizontal"
            android:layout_width="fill_parent"
            android:layout_height="40dip"
            android:id="@+id/tt"
            android:background="@drawable/title">
            <TextView
            android:text=
            "土豆白菜汤的做法 … …"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:id="@+id/t"
            android:textColor="#FF0000"
            android:gravity="center"
            android:textSize="30dip">
            </TextView>                          <!--TextView 控件-->
        </LinearLayout>                          <!--LinearLayout-->
        <LinearLayout
            android:orientation="horizontal"
            android:gravity="center_horizontal"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:id="@+id/tt">
            <TextView
            android:text="... ..."
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:id="@+id/t"
            android:textColor="#FF0000">
            </TextView>                          <!--TextView 控件-->
        </LinearLayout>                          <!--LinearLayout-->
    </LinearLayout>                              <!--LinearLayout-->
```

```
44        <LinearLayout
45            android:layout_width="fill_parent"
46            android:layout_height="fill_parent"
47            android:id="@+id/llllll"
48            android:gravity="bottom|right">
49            <Button
50                android:text="退出"
51                android:id="@+id/b"
52                android:layout_width="wrap_content"
53                android:layout_height="wrap_content">
54            </Button>                                    <!--Button 控件-->
55        </LinearLayout>                                  <!--LinearLayout-->
56    </FrameLayout>
```

提示：在该 xml 文件中实现了布局的设置。总的布局为 FrameLayout 布局，其中包括一个垂直模式的 LinearLayout 线性布局和一个 gravity 设置为 bottom 和 right 的 LinearLayout 线性布局。通过对各个控件的属性设置，实现了该小软件的界面效果。

上面已经介绍了本程序主界面 main.xml 的开发，接下来为读者介绍本程序具体功能的实现，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_4/src/com/bn/ex2d 目录下的 Sample2_4_Activity.class。

```
1    package com.bn. ex2d;                              //包名
2    import android.app.Activity;                       //包引用声明，该处有所省略，读者
可自行查看随书光盘源代码
3    public class Sample2_4_Activity extends Activity {  //该类继承自 Activity 类
4       public void onCreate(Bundle savedInstanceState) {     //onCreate 方法
5          super.onCreate(savedInstanceState);             //继承父类的 onCreate 方法
6          setContentView(R.layout.main);                  //界面跳转到 main.xml 界面
7          Button b=(Button)this.findViewById(R.id.b);     //获取 Button 按钮
8          b.setOnClickListener(                           //为 Button 按钮添加监听器
9             new OnClickListener(){
10                   public void onClick(View v) {
11                       System.exit(0);                  //当单击 Button 按钮时，退出程序
12                }}
13         );
14   }}
```

提示：在 Activity 类中，获取了按钮对象，并对其添加了按钮监听器，实现了单击按钮后退出程序的功能。

实例 5　结构紧凑的表格布局

在本小节中，通过 TableLayout 表格布局的应用，构建了一个显示学生信息表的界面，并通过对本程序的具体功能的实现，来向读者介绍 TableLayout 表格布局的具体应用。

【实例描述】

本小节构建了一个显示学生信息表小程序，在本程序中可以显示不同的同学的一些基本信息。包括学生的学号、姓名和籍贯等信息。本实例的运行效果图，如图 2-5 所示。

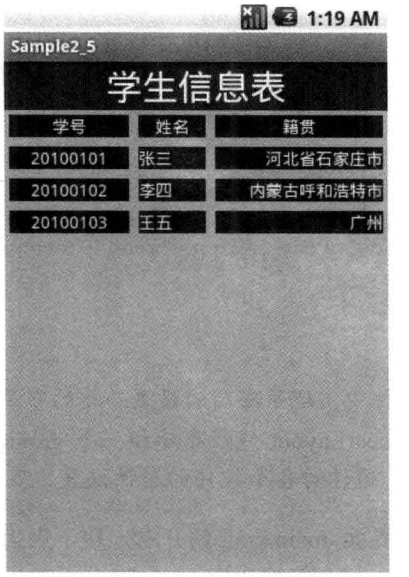


图 2-5　学生信息表截图

提示：在图 2-5 中，首先是“学生信息表”表头，在表头下面是字段名称部分，在字段名称下面是表中的一些基本信息。。

【实现过程】

本程序的开发主要运用了 TableLayout 表格布局的相关知识。表格布局以行和列的形式管理控件，每行为一个 TableRow 对象，也可以是一个 View 对象。TableRow 可以添加子控件，每次添加一个子控件即为一列。

【代码解析】

首先为读者介绍本程序的主界面 main.xml 的开发，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_5/ res/layout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
3       android:layout_width="fill_parent"   android:layout_height="fill_parent"
4       android:background="#edab4a"
5       android:stretchColumns="0,1,2"   android:shrinkColumns="1,2" >
6       <TextView
7       android:text="学生信息表"
8       android:gravity="center" android:id="@+id/tv1"
9       android:textColor="#FF0000" android:textSize="30dip"
10      android:background="#184124"/>
11      <TableRow>                                  <!--TableRow-->
12          <TextView
13              android:layout_column="0" android:text="学号"
14              android:gravity="center" android:textColor="#FF0000"
15              android:background="#182241" android:layout_margin="4dip"/>
16          <TextView
17              android:layout_column="1" android:text="姓名"
18              android:gravity="center" android:textColor="#FF0000"
19              android:background="#182241" android:layout_margin="4dip"/>
```

```
        <TextView
            android:text="籍贯"          android:gravity="center"
            android:textColor="#FF0000" android:background="#182241"
            android:layout_margin="4dip"/>
    </TableRow>                                    <!--TableRow-->
    <TableRow>
        <TextView
            android:text=" 20100101 "          android:gravity="center"
            android:textColor="#FF0000" android:background="#182241"
            android:layout_margin="4dip"/>     <!--TextView-->
        <TextView
            android:text="张三"          android:gravity="left"
            android:textColor="#FF0000" android:background="#182241"
            android:layout_margin="4dip"/>
        <TextView
            android:text="河北省石家庄市"      android:gravity="right"
            android:textColor="#FF0000" android:background="#182241"
            android:layout_margin="4dip"/>     <!--TextView 控件-->
    </TableRow>                                    <!--TableRow-->
    <TableRow>
        <TextView
            android:text=" 20100102 "          android:gravity="center"
            android:textColor="#FF0000" android:background="#182241"
            android:layout_margin="4dip"/>
        <TextView
            android:text="李四"          android:gravity="left"
            android:textColor="#FF0000" android:background="#182241"
            android:layout_margin="4dip"/>
        <TextView
            android:text="内蒙古呼和浩特市"      android:gravity="right"
            android:textColor="#FF0000" android:background="#182241"
            android:layout_margin="4dip"/>     <!--TextView 控件-->
    </TableRow>                                    <!--TableRow-->
    <TableRow>
        <TextView
            android:text="20100103"          android:gravity="center"
            android:textColor="#FF0000" android:background="#182241"
            android:layout_margin="4dip"/>
        <TextView
            android:text="王五"          android:gravity="left"
            android:textColor="#FF0000" android:background="#182241"
            android:layout_margin="4dip"/>
        <TextView
            android:text="广州"          android:gravity="right"
            android:textColor="#FF0000" android:background="#182241"
            android:layout_margin="4dip"/>
    </TableRow>                                    <!--TableRow-->
</TableLayout>
```

提示：在该 xml 文件中实现了布局的设置。总的布局为 TableLayout 布局。在总布局中，应用了 4 个 TableRow 来创建 4 行表数据。并通过对各个控件的属性设置，实现该小软件的界面效果。

上面已经介绍了本程序主界面 main.xml 的开发，接下来为读者介绍本程序的具体功能的实现，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_5/src/com/bn/ex2e 目录下的 Sample2_5_Activity.class。

```
1   package com.bn. ex2e;                                    //包名
2   import android.app.Activity;                             //包引用声明
3   import android.os.Bundle;
4   public class Sample2_5_Activity extends Activity {  //该类继承了 Activity 类
5      public void onCreate(Bundle savedInstanceState) {     //实现 onCreate()方法
6         super.onCreate(savedInstanceState);               //继承父类的 onCreate()方法
7         setContentView(R.layout.main);                    //将界面跳转到 main.xml 界面
8      }}
```

提示：该类主要实现 main.xml 界面的显示工作。

实例 6　用坐标精确布局

在本小节中，通过 AbsoluteLayout 绝对布局的应用，构建了一个软件登录界面，并通过本程序的实现，来向读者介绍 AbsoluteLayout 绝对布局的具体应用。

【实例描述】

本节中，构造了一个软件登录界面，在该界面中，可以输入姓名和密码，单击“确定”按钮，可以将所输入的姓名和密码显示到填写界面下方的文本编辑框中。单击“取消”按钮，则会清空所“填写”的姓名和密码。

本实例的运行效果图，如图 2-6 所示。

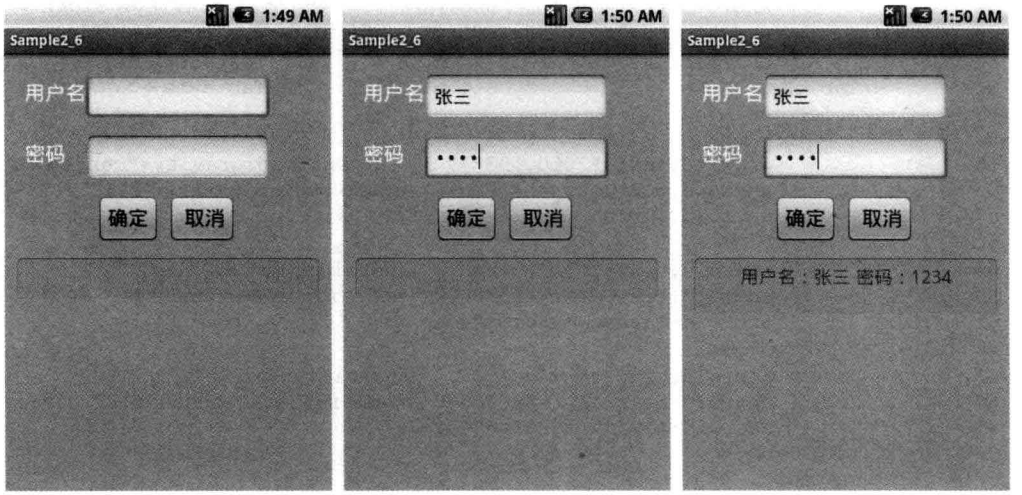


图 2-6　软件登录界面截图

提示：在图 2-6 中，实现了简单地输入数据并存储数据的功能。

【实现过程】

本程序的开发主要运用了 AbsoluteLayout 绝对布局的相关知识。所谓绝对布局是指由开发人员通过给出控件的坐标来在容器摆放控件，容器就不再负责这个工作了。

在该界面开发中，用到了 TextView 文本框控件，单行 EditText 控件、Button 按钮控件、

ScrollView 滚动控件和多行 EditText 控件。

【代码解析】

首先为读者介绍本程序主界面 main.xml 的开发，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_6/ res/layout 目录下的 main.xml。

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
   android:id="@+id/AbsoluteLayout01" android:layout_width="fill_parent"
   android:layout_height="fill_parent" android:background="#edab4a"
   xmlns:android="http://schemas.android.com/apk/res/android">
   <TextView
       android:layout_x="20dip" android:layout_y="20dip"
       android:layout_height="wrap_content"android:layout_width="wrap_content"
       android:id="@+id/TextView01" android:textColor="#FF0000"
       android:text="@string/uid">
    </TextView>                                    <!--TextView 控件-->
   <TextView
       android:layout_x="20dip" android:layout_y="80dip"
       android:layout_height="wrap_content"android:layout_width="wrap_content"
       android:id="@+id/TextView02" android:textColor="#FF0000"
       android:text="@string/pwd">
   </TextView>                                     <!--TextView 控件-->
   <EditText
      android:layout_x="80dip" android:layout_y="20dip"
      android:layout_height="wrap_content" android:layout_width="180dip"
      android:id="@+id/EditText01">
   </EditText>                                     <!--EditText 控件-->
   <EditText
      android:layout_x="80dip" android:layout_y="80dip"
      android:layout_height="wrap_content" android:layout_width="180dip"
      android:id="@+id/EditText02"android:password="true"
   >                                 <!-- android:password 设置是否为密码框 -->
   </EditText>                                     <!--EditText 控件-->
    <Button
       android:layout_x="155dip" android:layout_y="140dip"
       android:layout_height="wrap_content" android:id="@+id/Button01"
       android:layout_width="wrap_content" android:text="@string/ok">
    </Button>                                      <!--Button 控件-->
    <Button
       android:layout_x="210dip" android:layout_y="140dip"
       android:layout_height="wrap_content" android:id="@+id/Button02"
       android:layout_width="wrap_content" android:text="@string/cancel">
    </Button>                                      <!--Button 控件-->
    <ScrollView                                    <
      android:layout_x="10dip" android:layout_y="200dip"
      android:layout_height="220dip" android:layout_width="300dip"
      android:id="@+id/ScrollView01">               <!--ScrollView 控件-->
       <EditText
          android:layout_width="fill_parent"   android:layout_height="wrap_
content"
          android:id="@+id/EditText03" android:singleLine="false"
          android:gravity="top|center" android:enabled="false">
                                                   <!--EditText 控件-->
       </EditText>                                 <!--EditText 控件-->
    </ScrollView>                                  <!--ScrollView 控件-->
</AbsoluteLayout>
```

提示：在该 xml 文件中实现了布局的设置。总的布局为 AbsoluteLayout 布局。在总布局中，应用了 TextView 文本框控件、单行 EditText 控件、Button 按钮控件、ScrollView 滚动控件和多行 EditText 控件，并通过对各个控件的属性设置，实现了该小软件的界面效果。

上面已经介绍了本程序主界面 main.xml 的开发，接下来为读者介绍本程序具体功能的实现，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_6/src/com/bn/ex2f 目录下的 Sample2_6_Activity.class。

```
1   package com.bn. ex2f;                                          //包名
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中的源代码
3   import android.app.Activity;                                   //导入相关类
4   public class Sample2_6_Activity extends Activity {  //继承自 Activity 类
5      public void onCreate(Bundle savedInstanceState) {//实现 onCreate()方法
6         super.onCreate(savedInstanceState);                     //继承自父类的 onCreate()
7         setContentView(R.layout.main);                          //显示 main.xml 界面
8         final Button OkButton = (Button) findViewById(R.id.Button01);
                                                                  //获取界面中的“确定”按钮对象
9         final Button cancelButton = (Button) findViewById(R.id.Button02);
                                                                  //获取界面中的“取消”按钮对象
10        final EditText uid=(EditText)findViewById(R.id.EditText01);
11        final EditText pwd=(EditText)findViewById(R.id.EditText02);
12        final EditText log=(EditText)findViewById(R.id.EditText03);
13     OkButton.setOnClickListener(                               //负责监听鼠标单击事件
14           new View.OnClickListener()    {                     //为“确定”按钮添加监听器
15            public void onClick(View v) {
16               String uidStr=uid.getText().toString(); //得到 uid 的字符串
17               String pwdStr=pwd.getText().toString(); //得到 pwd 的字符串
18               log.append("用户名: "+uidStr+" 密码: "+pwdStr+"\n");
                                                                  //实现信息的存储
19            }});
20     cancelButton.setOnClickListener(                           //负责监听鼠标单击事件
21               new View.OnClickListener(){                     //为取“消按”钮添加监听器
22                public void onClick(View v) {
23                   uid.setText("");                             //清空输入文本
24                   pwd.setText("");                             //清空输入文本
25                }
26           });
27    }}
```

提示：在 Activity 类中，获取了按钮对象，并对其添加了按钮监听器，实现了单击“确定”按钮存储输入信息和单击“取消”按钮清空输入信息的功能。

实例 7　文字显示的技巧

【实例描述】

在本小节中实现了一个 TextView 文字显示的小程序，其可以在屏幕上显示文字。本实例的

运行效果图，如图 2-7 所示。

图 2-7　TextView 文本显示

提示：在该小应用开始运行时，在界面中显示的是“欢迎学习 android 实例 300 编”的提示文本信息，如图 2-7 所示。

【实现过程】

本小节通过对 TextView 文字显示的应用，实现了屏幕上显示文字的功能。

【代码解析】

接下来为读者介绍本程序中 main.xml 的开发，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_7/ res/layout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>          <!--版本号和编码方式-->
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="vertical"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent">
6      <TextView
7           android:id="@+id/TextView01"
8           android:layout_width="fill_parent"
9           android:layout_height="wrap_content"
10          android:text="@string/str">
11     </TextView>                                  <!--TextView 控件-->
12  </LinearLayout>                                 <!--LinearLayout-->
```

上面已经介绍了本程序主界面 main.xml 的开发，接下来向读者介绍本程序具体功能的实现，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_6/src/com/bn/ex2g 目录下的 Sample2_7_Activity.class。

```
1   package com.bn. ex2g;                           //声明包
2   import android.app.Activity;                    //导入相关类
3   import android.os.Bundle;                       //导入相关类
```

```
4   import android.widget.TextView;                                    //导入相关类
5   public class Sample2_7_Activity extends Activity {                 //继承自 Activity 类
6      /** Called when the activity is first created. */
7      @Override
8      public void onCreate(Bundle savedInstanceState) {               //重写的方法
9         super.onCreate(savedInstanceState);                          //调用父类
10        setContentView(R.layout.main);                               //跳转到主界面
11        TextView tv=(TextView)this.findViewById(R.id.TextView01);  //得到引用
12        String str_1="欢迎学习/nandroid 实例 300 编"; //其中/n 是换行符，具体看显示效果
13        tv.setText(str_1);                                           //设置文本
14     }}
```

提示：上面的主要内容是将一段文本添加到一个 TextView 中。

实例 8　文字颜色的设置

【实例描述】

在本小节中实现了一个 TextView 文字显示的小程序，其可以在屏幕上显示文字，并且可以为其设置不同的背景色和字体颜色。本实例的运行效果图，如图 2-8 所示。

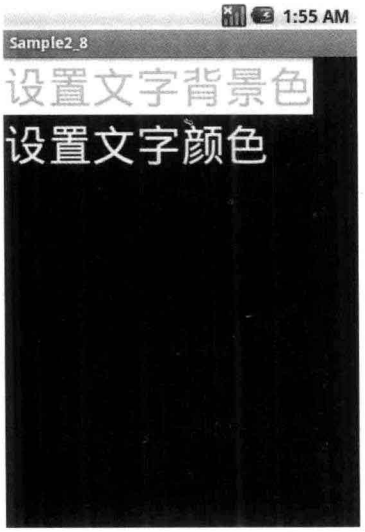


图 2-8　文字颜色设置

提示：在本程序开始运行时，在界面中显示如图 2-8 所示的提示文本信息。

【实现过程】

本程序通过对 TextView 颜色设置的应用，实现了更改 TextView 背景色和字体颜色的功能。

【代码解析】

首先为读者介绍本程序主界面 main.xml 的开发，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_8/ res/layout 目录下的 main.xml。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent">                    <!--LinearLayout-->
    <TextView
        android:text="@+id/TextView01"
        android:id="@+id/TextView01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
    </TextView>                                            <!--TextView 控件-->
    <TextView
        android:text="@+id/TextView02"
        android:id="@+id/TextView02"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
    </TextView>                                            <!--TextView 控件-->
</LinearLayout>                                            <!--LinearLayout-->
```

上面已经介绍了本程序的主界面 main.xml 的开发，接下来向读者介绍本程序具体功能的实现，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_6/src/com/bn/ex2h 目录下的 Sample2_8_Activity.class。

```
package com.bn. ex2h;                                      //声明包
……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码
import android.widget.TextView;                            //导入相关类
public class Sample2_8_Activity extends Activity {         //继承自 Activity 类
   @Override
   public void onCreate(Bundle savedInstanceState) {       //重写的方法
      super.onCreate(savedInstanceState);                  //调用父类
      setContentView(R.layout.main);                       //跳转到主界面
      TextView tv01=(TextView)this.findViewById(R.id.TextView01);
      tv01.setText("设置文字背景色");                       //设置 TextView 的文本
      Resources resources=getBaseContext().getResources();  //得到资源的引用
      Drawable Hdrawable=resources.getDrawable(R.color.white);
                                                           //得到图片的引用
      tv01.setBackgroundDrawable(Hdrawable);               //设置图片为背景
      TextView tv02=(TextView)this.findViewById(R.id.TextView02);//得到引用
      tv02.setText("设置文字颜色");                         //设置 TextView 的文本
      tv02.setTextColor(Color.RED);                        //设置颜色
   }}
```

提示：上面主要介绍了 TextView 在实际中的应用，可以在屏幕上显示不同背景和不同颜色的文字。

实例 9　使你的文字显得更独特

【实例描述】

在本小节中实现了一个使用 Style 样式化 TextView 的小程序，通过 Style 样式化的应用，避免了每次都设置 TextView 文字大小、TextView 颜色等。本实例的运行效果图，如图 2-9 所示。

图 2-9　样式化文本效果图

提示：在本程序开始运行时，在界面中显示如图 2-9 所示的提示文本信息。

【实现过程】

本程序通过使用 Style 样式化 TextView，可以实现 TextView 设置的批量处理。

【代码解析】

首先为读者介绍本程序的主界面 main.xml 的开发，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_9/ res/layout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="vertical"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent">              <!--LinearLayout-->
6       <TextView
7          style="@style/style01"
8           android:text=" android实例"
9           android:id="@+id/TextView01"
10          android:layout_width="wrap_content"
11          android:layout_height="wrap_content">
12      </TextView>                                      <!--TextView 控件-->
13      <TextView
14         style="@style/style02"
15          android:text=" android实例"
16          android:id="@+id/TextView02"
17          android:layout_width="wrap_content"
18          android:layout_height="wrap_content">
19      </TextView>                                  <!--TextView 控件-->
20  </LinearLayout>                                  <!--LinearLayout-->
```

上面已经介绍了本程序的主界面 main.xml 的开发，接下来介绍本程序关键的 style.xml 文件。

代码位置：见随书光盘中源代码/第 2 章/Sample2_9/ res/valuest 目录下的 sytle.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>          <!--版本号和编码方式-->
```

```
<resources>
    <style name="style01">
        <item name="android:textSize">16sp</item>        <!--设置大小-->
        <item name="android:textColor">#FFFFFF</item>    <!--设置颜色-->
    </style>
    <style name="style02">
        <item name="android:textSize">20sp</item>        <!--设置大小-->
        <item name="android:textColor">#fd8d8d</item>    <!--设置颜色-->
        <item name="android:fromAlpha">0.0</item>   <!--设置起始的透明度-->
        <item name="android:toAlpha">0.0</item>     <!--设置最终的透明度-->
    </style>
</resources>
```

上面已经介绍了本程序的 main.xml 与 style.xml 的开发，接下来为读者介绍本程序具体功能的实现，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_6/src/com/bn/ex2i 目录下的 Sample2_9_Activity.class。

```
package com.bn. ex2i;                                    //声明包
import android.app.Activity;                             //导入相关类
import android.os.Bundle;                                //导入相关类
public class Sample2_9_Activity extends Activity {  //继承自 Activity 类
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {//重写的方法
        super.onCreate(savedInstanceState);              //调用父类
        setContentView(R.layout.main);                   //跳转到主界面
    }}
```

提示：上面的代码主要是跳转到主界面，所有显示的工作全部都在 main.xml 中完成。

实例 10　简单的本地验证

在本小节中通过对 EdiText 编辑框的应用，构建了一个简单的登录界面——EditText 编辑框。并通过本程序的实现，来为读者介绍 EditText 编辑框的具体应用。

【实例描述】

在本小节中介绍简单的登录界面——EditText 编辑框的应用，在该界面中，若输入正确的用户名和密码，单击“确定”按钮，将出现一个 Toast 提示“恭喜您登录成功!”，否则将提示“请输入正确的用户名或密码!”。单击“清空”按钮，则会清空所填写的姓名和密码内容。

本实例的运行效果图，如图 2-10 所示。

提示：在该界面中，实现了对用户名和密码的验证功能，如图 2-10 所示。

【实现过程】

本程序的开发主要运用了 EditText 编辑框的相关知识。EditText 编辑框中可以输入内容，当用户名和密码正确的时候，单击“确定”按钮就会弹出 Toast 提示“恭喜您登录成功!”，否则将提示“请输入正确的用户名或密码!”。单击“清空”按钮，则会清空所填写的姓名和密码

内容。

在该界面开发中，还用到了 TextView 文本框控件、单行 EditText 控件与 Button 按钮控件。

图 2-10 EditText 编辑框

【代码解析】

首先为读者介绍本程序的主界面 main.xml 的开发，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_10/ res/layout 目录下的 main.xml。

```
<?xml version="1.0" encoding="utf-8"?>                    <!--版本号和编码方式-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >                                                     <!--LinearLayout-->
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="#ffcc66"
        android:paddingLeft="5dip"
        android:paddingRight="5dip"
        android:paddingTop="5dip">
        <LinearLayout
            android:id="@+id/LinearLayout01"
            android:orientation="horizontal"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent">          <!--LinearLayout-->
            <TextView
                android:text="用户名："
                android:id="@+id/TextView02"
                android:textColor="#222222"
                android:layout_width="wrap_content"
                android:layout_height="40dip"
                android:layout_marginLeft="5dip"
                android:textSize="18dip"
                android:gravity="center_vertical">
            </TextView>                                   <!--TextView 控件-->
              <EditText
                  android:id="@+id/EditTextuid"
                  android:singleLine="true"
```

```
33                  android:layout_width="fill_parent"
34                  android:layout_height="wrap_content"
35                  android:layout_marginLeft="0dip"
36                  android:text="e1001">                    <!--设置文本-->
37              </EditText>
38          </LinearLayout>                                  <!--LinearLayout-->
39          <LinearLayout
40              android:id="@+id/LinearLayout02"
41              android:orientation="horizontal"
42              android:layout_width="fill_parent"
43              android:layout_height="wrap_content">        <!--纵向自适应大小-->
44              <TextView
45                  android:text="密　码："
46                  android:id="@+id/TextView03"
47                  android:textColor="#222222"
48                  android:layout_width="wrap_content"
49                  android:layout_height="40dip"
50                  android:layout_marginLeft="5dip"
51                  android:textSize="18dip"
52                  android:gravity="center_vertical"><!--设置位置-->
53              </TextView>                                  <!--TextView 控件-->
54              <EditText
55                  android:id="@+id/EditTextPwd"
56                  android:singleLine="true"
57                  android:layout_width="fill_parent"
58                  android:layout_height="wrap_content"
59                  android:text="123456">                   <!--设置文本-->
60              </EditText>
61          </LinearLayout>
62          <LinearLayout
63              android:id="@+id/LinearLayout03"
64              android:orientation="horizontal"
65              android:layout_width="wrap_content"
66              android:layout_height="wrap_content">        <!--纵向自适应大小-->
67              <Button
68                  android:text=" 登录 "
69                  android:id="@+id/loginLog"
70                  android:layout_width="75dip"
71                  android:layout_height="40dip"
72                  android:textSize="18dip"
73                  android:gravity="center">                <!--设置位置-->
74              </Button>                                    <!--Button 控件-->
75              <Button
76                  android:text=" 清空 "
77                  android:id="@+id/loginClear"
78                  android:layout_width="75dip"
79                  android:layout_height="40dip"
80                  android:textSize="18dip"
81                  android:gravity="center">                <!--设置位置-->
82              </Button>                                    <!--Button 控件-->
83          </LinearLayout>                                  <!--LinearLayout-->
84      </LinearLayout>                                      <!--LinearLayout-->
85  </LinearLayout>                                          <!--LinearLayout-->
```

上面已经介绍了本程序的主界面 main.xml 的开发，接下来将要为读者介绍本程序的具体功能的实现，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_10/src/com/bn/ex2j 目录下的 Sample2_10_Activity.class。

```
1   package com.bn.ex2j;                                    //声明包
```

```
2    ……//该处省略了部分类的导入，读者可自行查看随书光盘中的源代码
3    import android.widget.Toast;                                    //导入相关类
4    public class Sample2_10_Activity extends Activity {             //继承自 Activity 类
5       @Override
6       public void onCreate(Bundle savedInstanceState) {            //重写的方法
7          super.onCreate(savedInstanceState);                       //调用父类
8          setContentView(R.layout.main);                            //跳转到主界面
9          Button bLogin =(Button)this.findViewById(R.id.loginLog);//“登录”按钮
10      Button bClear=(Button)this.findViewById(R.id.loginClear); //“清空”按钮
11      final EditText eUid=(EditText)this.findViewById(R.id.EditTextuid);
                                                                    //用户名
12      final EditText eMima=(EditText)this.findViewById(R.id.EditTextPwd);//密码
13      bLogin.setOnClickListener(                                  //添加“登录”按钮监听器
14           new OnClickListener(){                                 //匿名内部类
15              @Override
16              public void onClick(View v){                        //重写的方法
17                  String strUid=eUid.getText().toString().trim();
                                                                    //得到 eUid 的字符串
18                  String strPwd=eMima.getText().toString().trim();
                                                                    //得到 eMima 的字符串
19                  if(strUid.equals("e1001")&&strPwd.equals("123456")){
                                                                    //判断是否符合条件
20                      Toast.makeText(Sample2_10_Activity.this, //弹出 Toast
21                          "恭喜您登录成功！",
22                          Toast.LENGTH_SHORT).show();
23                  }else{
24                      Toast.makeText(Sample2_10_Activity.this, //弹出 Toast
25                          "请输入正确的用户名或密码！",
26                          Toast.LENGTH_SHORT).show();
27      }}});
28      bClear.setOnClickListener(                                  //添加“清空”按钮监听器
29              new OnClickListener(){                              //匿名内部类
30                  @Override
31                  public void onClick(View v) {                   //重写的方法
32                      eUid.setText("");                           //设置用户名为空
33                      eMima.setText("");                          //设置密码为空
34   }});}}
```

其中：

- 第 14～27 行表示单击“登录”按钮之后，判断其用户名和密码是否符合条件。
- 第 28～34 行表示单击“清空”按钮之后，将显示的用户名和密码全部清空。

实例 11　性别的选择

在本小节中，通过一个性别选择的实例，来向读者介绍 RadikButton 组的具体应用。

【实例描述】

在本小节中，通过对 RadioButton 组的应用，构建了一个性别选择的小程序。并通过本程序的实现，来为读者介绍 RadioButton 组的具体应用。本实例的运行效果图，如图 2-11 所示。

提示：该小应用的主要作用是，在只有一个选择的情况下为用户提供更方便的选择，如图 2-11 所示。

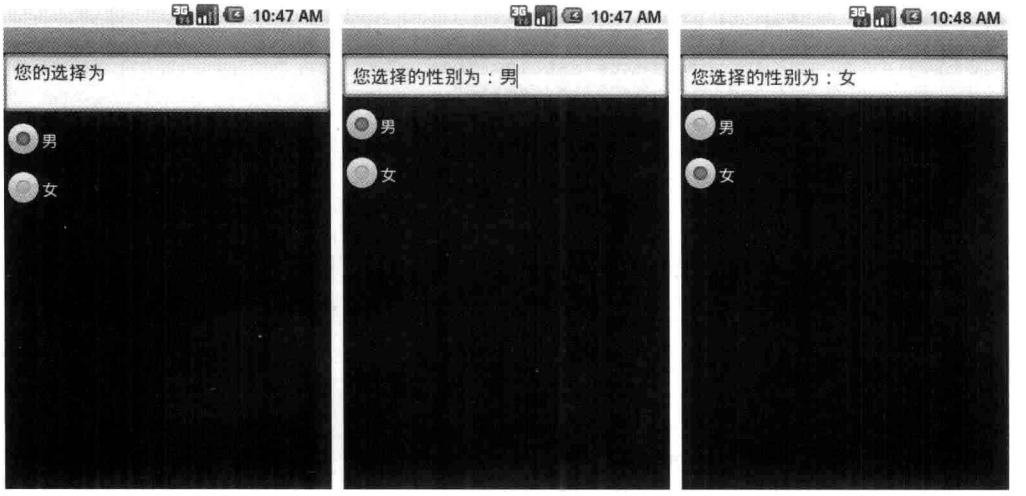


图 2-11　RadioButton 应用效果图

【实现过程】

本程序的开发主要运用了 RadioButton 单选按钮与 EditText 编辑框的相关知识。RadioButton 在选择时不可以同时选择两个，当选择某一项单选按钮时，可以在 EditText 编辑框中输出自己的选择。

在界面开发中，用到了 TextView 文本框控件、单行 EditText 控件与 Button 按钮控件。

【代码解析】

首先为读者介绍本程序的主界面 main.xml 的开发，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_11/ res/layout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>          <!--xml 文件版本号和编码方式-->
2   <LinearLayout
3       android:id="@+id/LinearLayout01"
4       android:layout_width="fill_parent"
5       android:layout_height="fill_parent"
6       android:orientation="vertical"
7       xmlns:android="http://schemas.android.com/apk/res/android">
8       <EditText
9           android:id="@+id/EditText01"
10          android:layout_width="fill_parent"
11          android:layout_height="wrap_content"
12          android:singleLine="false"
13          android:text="@string/label">
14      </EditText>                                     <!--EditText 控件-->
15      <RadioGroup
16        android:id="@+id/RadioGroup01"
17        android:orientation="vertical"
18        android:layout_width="wrap_content"
19        android:layout_height="wrap_content">         <!--纵向自适应大小-->
20         <RadioButton
21             android:text="@string/male"
22             android:id="@+id/male"
23             android:checked="true"
24             android:layout_width="wrap_content"
25             android:layout_height="wrap_content">    <!--纵向自适应大小-->
```

```
        </RadioButton>
         <RadioButton
             android:text="@string/female"
             android:id="@+id/female"
             android:layout_width="wrap_content"              >
             android:layout_height="wrap_content">       <!--纵向自适应大小-->
         </RadioButton>
      </RadioGroup>
</LinearLayout>
```

上面已经介绍了本程序的主界面 main.xml 的开发，接下来介绍本程序的 string.xml 文件。

代码位置：见随书光盘中源代码/第 2 章/Sample2_11/ res/values 目录下的 string.xml。

```
<?xml version="1.0" encoding="utf-8"?>          <!--xml 文件版本号和编码方式-->
<resources>
    <string name="hello">Hello World, Sample2_11!</string>
    <string name="app_name"></string>           <!--本应用程序的名称-->
    <string name="label">您的选择为\n</string>    <!--label 对应的字符串-->
    <string name="male">男</string>             <!--male 对应的字符串-->
    <string name="female">女</string>           <!--female 对应的字符串-->
</resources>
```

上面已经介绍了本程序的 main.xml 与 string.xml 的开发，接下来为读者介绍本程序具体功能的实现，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_11/src/com/bn/ex2k 目录下的 Sample2_11_RadioButton.class。

```
package com.bn.ex2k;                                        //声明包
……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码
import android.widget.RadioButton;                          //导入相关类
public class Sample2_11_RadioButton extends Activity{       //继承自 Activity 类
    @Override
    public void onCreate(Bundle savedInstanceState){        //重写的方法
        super.onCreate(savedInstanceState);                 //调用父类
        setContentView(R.layout.main);                      //跳转到主界面
        RadioButton rb=(RadioButton)findViewById(R.id.male);
                                                            //得到 RadioButton 的引用
        rb.setOnClickListener(                              //设置监听事件
           new OnClickListener(){                           //匿名内部类
                  @Override
                  public void onClick(View v){              //重写的方法
                      String result="您选择的性别为：男";     //设置字符串
                  EditTextet=(EditText)                     //得到引用

Sample2_11_RadioButton.this.findViewById(R.id.EditText01);
                  et.setText("");                           //设置为空
                  et.append(result);                      //添加 result 代表的字符串
    }});
        final RadioButton rbf=(RadioButton)findViewById(R.id.female);//得到引用
        rbf.setOnClickListener(                             //设置监听事件
              new OnClickListener(){                        //匿名内部类
                  @Override
                  public void onClick(View v){              //重写的方法
                      String result="您选择的性别为：女";     //设置字符串
                      EditText et=(EditText)                //得到引用

Sample2_11_RadioButton.this.findViewById(R.id.EditText01);
                      et.setText("");                       //设置为空
```

```
29                          et.append(result);              //添加 result 代表的字符串
30   }});}}
```

其中：

- 第 10～19 行表示选择的单选按钮为“男”，并且将选择的该项显示到 EditText 中。
- 第 20～30 行表示选择的单选按钮为“女”，并且将选择的该项显示到 EditText 中。

实例 12　选择喜欢的玩家

【实例描述】

在本小节中，通过 RadioButton ID 的应用，构建了一个用户选择喜欢的玩家的小程序。并通过本程序的实现，来为读者介绍 RadioButton ID 的具体应用。本实例的运行效果图，如图 2-12 所示。

图 2-12　RadioButton ID 的应用效果图

提示：该小应用的主要作用是，在只有一个选择的情况下为用户提供更方便的选择，如图 2-12 所示。

【实现过程】

本程序的开发主要运用了 RadioButton 单选按钮与 TextView 的相关知识。RadioButton 在选择时不可以同时选择两个，当选择某一项单选按钮时，可以在 TextView 中显示自己的选择。

【代码解析】

首先为读者介绍本程序的主界面 main.xml 的开发，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_12/ res/layout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>          <!--xml 文件版本号和编码方式-->
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3       android:orientation="vertical"
4       android:layout_width="fill_parent"
```

```
    android:layout_height="fill_parent">                    <!--LinearLayout-->
    <TextView
       android:text="请选择:"
         android:id="@+id/TextView01"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content">
     </TextView>                                            <!--TextView 控件-->
    <RadioGroup
       android:id="@+id/RadioGroup01"
       android:orientation="vertical"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content">
        <RadioButton
            android:text="蛛丝马迹"
            android:id="@+id/RadioButton01"
            android:checked="false"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content">
        </RadioButton>                                      <!--RadioButton-->
         <RadioButton
             android:text="burning"
             android:id="@+id/RadioButton02"
             android:checked="false"
             android:layout_width="wrap_content"
             android:layout_height="wrap_content">
         </RadioButton>                                     <!--RadioButton-->
        <RadioButton
            android:text="2009"
            android:checked="false"
            android:id="@+id/RadioButton03"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content">
        </RadioButton>                                      <!--RadioButton-->
    </RadioGroup>
</LinearLayout>
```

上面已经介绍了本程序的主界面 main.xml 的开发，接下来为读者介绍本程序具体功能的实现，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_12/src/com/bn/ex2l 目录下的 Sample2_12_Activity.class。

```
package com.bn.ex2l;                                        //声明包
……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码
import android.widget.TextView;                             //导入相关类
public class Sample2_12_Activity extends Activity{          //继承自 Activity 类
    TextView tv;                                            //TextView 的成员变量
    RadioButton rb1,rb2,rb3;                                //RadioButton 的成员变量
    RadioGroup rg;                                          //RadioGroup 的成员变量
   @Override
   public void onCreate(Bundle savedInstanceState){         //重写的方法
      super.onCreate(savedInstanceState);                   //调用父类
      setContentView(R.layout.main);                        //跳转到主界面
      tv=(TextView)this.findViewById(R.id.TextView01);  //得到 TextView 的引用
      rb1=(RadioButton)this.findViewById(R.id.RadioButton01);
                                                    //得到 RadioButton 01 的引用
      rb2=(RadioButton)this.findViewById(R.id.RadioButton02);
                                                    //得到 RadioButton 02 的引用
      rb3=(RadioButton)this.findViewById(R.id.RadioButton03);
                                                    //得到 RadioButton 03 的引用
      rg=(RadioGroup)this.findViewById(R.id.RadioGroup01);
```

```
                                                        //得到 RadioGroup 的引用
17          rg.setOnCheckedChangeListener(                  //设置监听事件
18            new RadioGroup.OnCheckedChangeListener(){     //匿名内部类
19                @Override
20                public void onCheckedChanged(RadioGroup group, int checkedId){
//重写的方法
21                    if(checkedId==rb1.getId()){    //判断是否选中第一个单选按钮
22                        tv.setText("您喜欢的玩家:"+rb1.getText());    //设置文本
23                    }else if(checkedId==rb2.getId()){//判断是否选中第二个单选按钮
24                        tv.setText("您喜欢的玩家:"+rb2.getText());    //设置文本
25                    }else if(checkedId==rb3.getId()){//判断是否选中第三个单选按钮
26                        tv.setText("您喜欢的玩家:"+rb3.getText());    //设置文本
27                        }
28    }});}}
```

其中：

- 第 12～16 行表示得到 TextView、RadioButton 与 RadioGroup 的引用。
- 第 17～28 行表示对 RadioGroup 监听，判断被选中的单选按钮，并在 TextView 中显示。

实例 13　确认提交

【实例描述】

在本小节中，通过对 CheckBox 的 isChecked 属性的应用，构建了 CheckBox 的 isChecked 属性控制 Button 按钮的 Enable 的小程序。并通过本程序的实现，来为读者介绍 CheckBox 多选项的具体应用。本实例的运行效果图，如图 2-13 所示。

图 2-13　CheckBox 应用效果图

提示：在多选按钮选中的情况下，单击“确定”按钮，在多选按钮上面显示“已经选择”，如图 2-13 所示。

【实现过程】

本程序主要用到的是 CheckBox 的 isChecked 属性，在选中的时候单击“确定”按钮可以为

TextView 设置信息。

【代码解析】

首先为读者介绍本程序的主界面 main.xml 的开发，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_13/ res/layout 目录下的 main.xml。

```
<?xml version="1.0" encoding="utf-8"?>          <!--xml 文件版本号和编码方式-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent">         <!--LinearLayout-->
    <TextView
        android:text="@string/TextView01"
        android:id="@+id/TextView01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
    </TextView>                                 <!--TextView 控件-->
    <CheckBox
        android:text="ischeck"
        android:id="@+id/CheckBox01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
    </CheckBox>                                 <!-- CheckBox -->
    <Button
        android:text="确定"
        android:id="@+id/Button01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">   <!--纵向自适应大小-->
    </Button>                                   <!--Button 控件-->
</LinearLayout>
```

上面已经介绍了本程序的主界面 main.xml 的开发，接下来介绍本程序的 string.xml 文件。

代码位置：见随书光盘中源代码/第 2 章/Sample2_13/ res/values 目录下的 string.xml。

```
<?xml version="1.0" encoding="utf-8"?>          <!--xml 文件版本号和编码方式-->
<resources>
   <string name="hello">Hello World, Sample2_13_Activity!</string>
   <string name="app_name">Sample2_13</string>  <!--本应用程序的名称-->
   <string name="TextView01">初始化</string>    <!--TextView01 对应的文字-->
</resources>
```

上面已经介绍了本程序的 main.xml 与 string.xml 的开发，接下来为读者介绍本程序具体功能的实现，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_11/src/com/bn/ex2m 目录下的 Sample2_13_Acitvity.class。

```
package com.bn.ex2m;                                        //声明包
……//该处省略了部分类的导入，读者可自行查看随书光盘中的源代码
import android.widget.TextView;                             //导入相关类
public class Sample2_13_Activity extends Activity{          //继承自 Activity 类
   @Override
   public void onCreate(Bundle savedInstanceState){         //重写的方法
      super.onCreate(savedInstanceState);                   //调用父类
      setContentView(R.layout.main);                        //跳转到主界面
      final TextView tv01=(TextView)this.findViewById(R.id.TextView01);
                                                            //得到 TextView 的引用
      final CheckBox cb=(CheckBox)this.findViewById(R.id.CheckBox01);
                                                            //得到 CheckBox 的引用
```

```
11        final Button  but=(Button)this.findViewById(R.id.Button01);
                                                          //得到 Button 按钮的引用
12        cb.setChecked(false);                           //设定一开始为选中
13        but.setEnabled(false);                          //Button 按钮不可单击
14        cb.setOnClickListener(                          //设置监听事件
15        new OnClickListener(){                          //匿名内部类
16              @Override
17              public void onClick(View v){              //重写的方法
18                   if(cb.isChecked()){                  //CheckBox 被选中
19                        tv01.setText("");               //设置文本的长度为“”
20                        but.setEnabled(true);           //Button 按钮可以被单击
21                   }else{
22                        but.setEnabled(false);          //Button 按钮不可以被单击
23                        tv01.setText("请勾选我");       //设置文本
24  }}});
25        but.setOnClickListener(                         //Button 按钮设置监听
26           new OnClickListener(){                       //匿名内部类
27                @Override
28                public void onClick(View v){            //重写的方法
29                   if(cb.isChecked()){                  //CheckBox 被选中
30                        tv01.setText("已经选择");       //设置文本
31  }}});}}
```

其中：

- 第 14～24 行表示对 CheckBox 多选按钮进行监听，判断其是否被选中。
- 第 25～31 行表示对 Button 按钮的监听，判断其是否被单击。

实例 14　个人爱好选择

【实例描述】

在本小节中，通过对 CheckBox 多选项的应用，构建了一个个人爱好选择的小程序。并通过本程序的实现，来为读者介绍 CheckBox 多选项的具体应用。本实例的运行效果图，如图 2-14 所示。

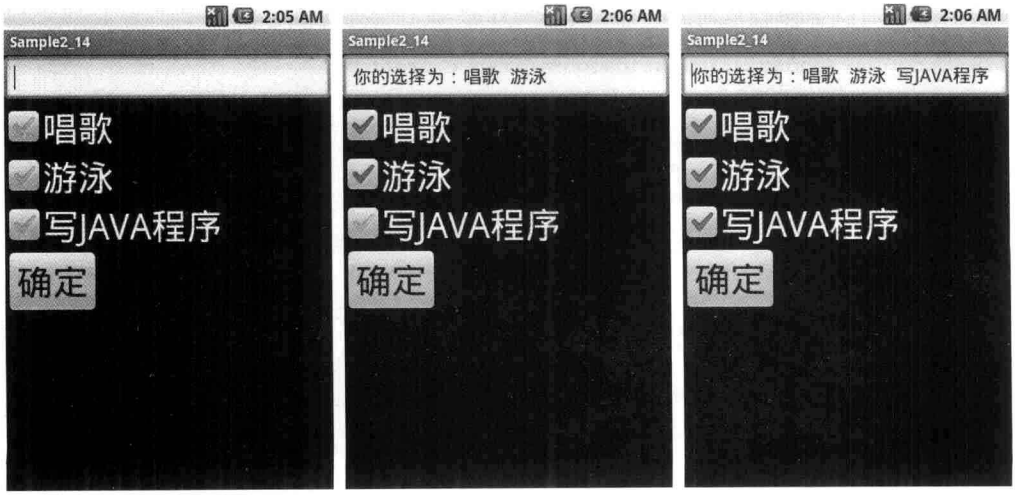


图 2-14　CheckBox 多选项应用效果图

提示：在图 2-14 中可以选择多个多选按钮，单击“确定”按钮，可以在文本框中显示自己所选的内容。

【实现过程】

本程序通过对 CheckBox 的应用，实现了多选的功能。

【代码解析】

首先为读者介绍本程序主界面 main.xml 的开发，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_14/ res/layout 目录下的 main.xml。

```
<?xml version="1.0" encoding="utf-8"?>          <!--xml 文件版本号和编码方式-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent">         <!--LinearLayout-->
    <EditText
        android:id="@+id/EditText01"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:singleLine="false">    <!--EditText 单行显示设置为 false-->
    </EditText>                                 <!--TextView 控件-->
    <CheckBox
        android:text="唱歌"
        android:id="@+id/CheckBox01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
    </CheckBox>                                 <!-- CheckBox -->
    <CheckBox
        android:text="游泳"
        android:id="@+id/CheckBox02"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
    </CheckBox>                                 <!-- CheckBox -->
    <CheckBox
        android:text="写 JAVA 程序"
        android:id="@+id/CheckBox03"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
    </CheckBox>                                 <!-- CheckBox -->
    <Button
        android:text="确定"
        android:id="@+id/Button01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">   <!--纵向自适应大小-->
    </Button>                                   <!--Button 控件-->
</LinearLayout>                                 <!--LinearLayout-->
```

上面已经介绍了本程序的主界面 main.xml 的开发，接下来为读者介绍本程序具体功能的实现，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_11/src/com/bn/ex2n 目录下的 Sample2_14_Acitvity.class。

```
package com.bn.ex2n;                                    //声明包
……//该处省略了部分类的导入，读者可自行查看随书光盘中的源代码
```

```
3   import android.widget.EditText;                                    //导入相关类
4   public class Sample2_14_Activity extends Activity{                 //继承自 Activity 类
5      String result;                                                  //声明成员变量
6      @Override
7      public void onCreate(Bundle savedInstanceState){ //重写的方法
8         super.onCreate(savedInstanceState);                          //调用父类
9         setContentView(R.layout.main);                               //跳转到主界面
10        final CheckBox cb1=(CheckBox)this.findViewById(R.id.CheckBox01);
                                                                       //多选按钮的引用
11        final CheckBox cb2=(CheckBox)this.findViewById(R.id.CheckBox02);
                                                                       //多选按钮的引用
12        final CheckBox cb3=(CheckBox)this.findViewById(R.id.CheckBox03);
                                                                       //多选按钮的引用
13        Button but=(Button)this.findViewById(R.id.Button01);   //Button 的引用
14        final EditText et=(EditText)this.findViewById(R.id.EditText01);
                                                                       //输入文本框的引用
15        but.setOnClickListener(                                      //设置监听事件
16              new OnClickListener(){                                 //匿名内部类
17                    @Override
18                    public void onClick(View v){   //重写的方法
19                        result="你的选择为: ";                       //声明 result 的字符串
20                        et.setText("");                              //文本框里的内容长度为 1
21                        if(cb1.isChecked()){                         //多选按钮 1 被选中
22                            result+="唱歌  ";                        //为 result 赋值
23                        }if(cb2.isChecked()){                        //多选按钮 2 被选中
24                            result+="游泳  ";                        //为 result 赋值
25                        }if(cb3.isChecked()){                        //多选按钮 3 被选中
26                            result+="写 JAVA 程序\n";    //为 result 赋值
27                        }
28                        et.setText(result.toString().trim());//设置文本框内容
29  }});}}
```

其中：

- 第 19～12 行表示对 3 个复选框的引用的声明。
- 第 15～29 行表示对按钮的监听事件，判断哪个多选按钮被选中，并在文本框中显示。

实例 15　灯泡开关

在本小节中，通过对 ToggleButton 控件的应用，并综合 RadioButton 控件、CheckBox 控件的应用，构建了一个开关灯程序。并通过本程序的实现，来为读者介绍 ToggleButton 的具体应用。

【实例描述】

在本小节中实现了一个开关灯的小程序，通过 ToggleButton 控件、RadioButton 控件、CheckBox 控件的应用，可以控制灯泡的开关。本实例的运行效果图，如图 2-15 所示。

提示：本程序可以通过 ToggleButton 控件、RadioButton 控件、CheckBox 控件等，控制灯泡的开关，如图 2-15 所示。

【实现过程】

本程序通过 ToggleButton 控件、RadioButton 控件、CheckBox 控件等来控制灯泡的开关，

同时实现了 ToggleButton 控件、RadioButton 控件、CheckBox 控件的同步。

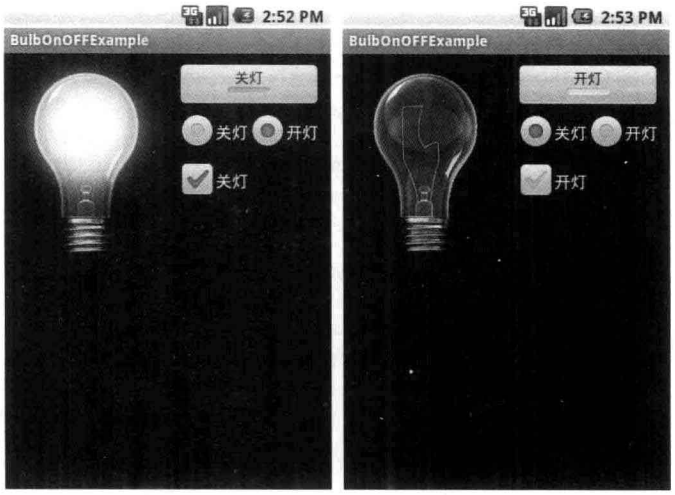


图 2-15　为开关灯软件截图

【代码解析】

首先为读者介绍本程序主界面 main.xml 的开发，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_15/ res/layout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>      <!--xml 文件版本号和编码方式-->
2   <LinearLayout
3      android:id="@+id/LinearLayout01"
4      android:orientation="horizontal"
5      android:layout_width="fill_parent"
6      android:layout_height="fill_parent"
7      xmlns:android="http://schemas.android.com/apk/res/android">
8      <ImageView
9          android:id="@+id/ImageView01"
10         android:src="@drawable/bulb_off"
11         android:layout_width="wrap_content"
12         android:layout_height="wrap_content">
13     </ImageView>                              <!--ImageView-->
14     <LinearLayout
15         android:id="@+id/LinearLayout02"
16         android:orientation="vertical"
17         android:paddingLeft="10dip"
18         android:paddingTop="10dip"
19         android:layout_width="wrap_content"
20         android:layout_height="wrap_content">
21         <ToggleButton
22             android:textOn="@string/off"
23             android:textOff="@string/on"
24             android:id="@+id/ToggleButton01"
25             android:layout_width="140dip"
26             android:layout_height="wrap_content">
27         </ToggleButton>                       <!--ToggleButton-->
28         <RadioGroup
29             android:id="@+id/RadioGroup01"
30             android:orientation="horizontal"
31             android:layout_width="wrap_content"
32             android:layout_height="wrap_content">
33             <RadioButton
```

```
                android:text="@string/off"
                android:id="@+id/off"
                android:checked="true"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content">    <!--纵向自适应大小-->
            </RadioButton>
            <RadioButton
                android:text="@string/on"
                android:id="@+id/on"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content">    <!--纵向自适应大小-->
            </RadioButton>
        </RadioGroup>
        <CheckBox
            android:text="@string/on"
            android:id="@+id/CheckBox01"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content">        <!--纵向自适应大小-->
        </CheckBox>
    </LinearLayout>
</LinearLayout>
```

上面已经介绍了本程序的主界面 main.xml 的开发，接下来介绍本程序的 string.xml 文件。

代码位置：见随书光盘中源代码/第 2 章/Sample2_15/ res/values 目录下的 string.xml。

```
<?xml version="1.0" encoding="utf-8"?>          <!--xml 文件版本号和编码方式-->
<resources>
    <string name="hello">Hello World, BulbOnOFF!</string>
    <string name="app_name">BulbOnOFFExample</string> <!--本应用程序的名称-->
    <string name="on">开灯</string>                    <!--设置名称-->
    <string name="off">关灯</string>                   <!--设置名称-->
</resources>
```

上面已经介绍了本程序的 main.xml 与 string.xml 的开发，接下来为读者介绍本软件的具体功能的实现，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_15/src/com/bn/ex2o 目录下的 Sample2_15_BulbOnOFF.class。

```
package com.bn.ex2o;                                    //声明包
……//该处省略了部分类的导入，读者可自行查看随书光盘中的源代码
import android.widget.CompoundButton.OnCheckedChangeListener;//导入相关类
public class Sample2_15_BulbOnOFF extends Activity{ //继承自 Activity 类
    @Override
    public void onCreate(Bundle savedInstanceState){ //重写的方法
        super.onCreate(savedInstanceState);          //调用父类
        setContentView(R.layout.main);               //跳转到主界面
        ToggleButton tb=(ToggleButton)this.findViewById(R.id.ToggleButton01);
        tb.setOnCheckedChangeListener(               //设置监听事件
              new OnCheckedChangeListener(){         //匿名内部类
              @Override
              public void onCheckedChanged(CompoundButton buttonView,
                                                     //重写的状态改变方法
                        boolean isChecked){
              setBulbState(isChecked);               //设置选中状态
        } } );
        CheckBox cb=(CheckBox)this.findViewById(R.id.CheckBox01);
                                                     //得到多选按钮的引用
        cb.setOnCheckedChangeListener(               //设置监听事件
          new OnCheckedChangeListener(){             //匿名内部类
```

```
20              @Override
21              public void onCheckedChanged(CompoundButton buttonView,
                                                          //重写的状态改变方法
22                      boolean isChecked){
23              setBulbState(isChecked);                  //设置选中状态
24          } } );
25          RadioButton rb=(RadioButton)findViewById(R.id.off);//得到单选按钮的引用
26          rb.setOnCheckedChangeListener(                //设置监听事件
27             new OnCheckedChangeListener(){             //匿名内部类
28              @Override
29              public void onCheckedChanged(CompoundButton buttonView,
                                                          //重写的状态改变方法
30                      boolean isChecked){
31              setBulbState(!isChecked);                 //设置选中状态
32       } } ); }
33      public void setBulbState(boolean state){          //设置状态的方法
34          ImageView iv=(ImageView)findViewById(R.id.ImageView01);
                                                          //得到 ImageView 引用
35
iv.setImageResource((state)?R.drawable.bulb_on:R.drawable.bulb_off);
                                                          //判断 state 的值
36          ToggleButton
tb=(ToggleButton)this.findViewById(R.id.ToggleButton01);  //得到引用
37          tb.setChecked(state);                         //设置其状态
38          CheckBox cb=(CheckBox)this.findViewById(R.id.CheckBox01);
                                                          //得到多选按钮的引用
39          cb.setText((state)?R.string.off:R.string.on); //设置文本
40          cb.setChecked(state);                         //设置对应的状态
41          RadioButton rb=(RadioButton)findViewById(R.id.off);
                                                          //得到 off 单选按钮引用
42          rb.setChecked(!state);                        //设置 off 单选按钮状态
43          rb=(RadioButton)findViewById(R.id.on);        //得到 on 单选按钮状态
44          rb.setChecked(state);                         //设置 on 单选按钮状态
45  } }
```

其中：

- 第 10～16 行表示对 ToggleButton 的监听，判断其状态是否改变。
- 第 18～24 行表示对多选按钮 CheckBox 的监听，判断其是否被选中。
- 第 26～32 行表示对单选按钮 RadioButton 的监听，判断其是否被选中。
- 第 33～45 行表示对本实例的按钮状态的设置。

实例 16　最亲和的提示

【实例描述】

在本小节中，通过对 Toast 消息提示的应用，构建了一个单击 Button 按钮出现 Toast 消息提示的小程序。并通过本程序的实现，来为读者介绍 Toast 的具体应用。本实例的运行效果图，如图 2-16 所示。

提示：在本程序运行后单击“Toast 提示”按钮，可以弹出 Toast 提示，如图 2-16 所示。

图 2-16　Toast 应用效果图

【实现过程】

本程序为 Toast 消息提示的应用，实现了单击 Button 按钮显示 Toast 提示的功能。

【代码解析】

首先为读者介绍本程序主界面 main.xml 的开发，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_16/ res/layout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>          <!--xml 文件版本号和编码方式-->
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="vertical"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent">         <!--LinearLayout-->
6       <Button
7            android:text="Toast 提示"
8            android:id="@+id/Button01"
9            android:layout_width="wrap_content"
10           android:layout_height="wrap_content"> <!--纵向自适应大小-->
11      </Button>                                   <!--Button 控件-->
12  </LinearLayout>                                 <!--LinearLayout-->
```

上面已经介绍了本程序的主界面 main.xml 的开发，接下来为读者介绍本程序的具体功能的实现，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_15/src/com/bn/ex2p 目录下的 Sample2_15_Activity.class。

```
1   package com.bn.ex2p;                                          //声明包
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中的源代码
3   import android.widget.Toast;                                  //导入相关类
4   public class Sample2_16_Activity extends Activity{            //继承自 Activity 类
5      @Override
6      public void onCreate(Bundle savedInstanceState){           //重写的方法
7         super.onCreate(savedInstanceState);                     //调用父类
8         setContentView(R.layout.main);                          //跳转到主界面
9         Button but=(Button)this.findViewById(R.id.Button01); //Button 按钮的引用
10        but.setOnClickListener(                                 //设置监听事件
```

```
11          new OnClickListener(){                          //匿名内部类
12                  @Override
13                  public void onClick(View v){            //重写的方法
14                      Toast.makeText(Sample2_16_Activity.this,//弹出 Toast 提示
15                                     "Toast 提示",
16                                     Toast.LENGTH_SHORT).show();
17  } } ); } }
```

提示：上面的主要功能是对按钮的监听，并且单击 Button 按钮弹出 Toast 提示。

实例 17　有背景图片的按钮

【实例描述】

在本小节中，通过对 ImageButton 的应用，构建了一个有背景图片的按钮的小程序。并通过本程序的实现，来向读者介绍 ImageButton 的具体应用。本实例的运行效果图，如图 2-17 所示。

图 2-17　ImageButton 应用效果图

提示：在运行本程序后单击“ImageButton”按钮可以弹出 Toast 提示，并且改变按钮的背景色，如图 2-17 所示。

【实现过程】

本程序主要应用了单击 ImageButton 按钮，更换按钮图片，即可达到改变背景色的效果。

【代码解析】

首先为读者介绍本程序主界面 main.xml 的开发，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_17/ res/layout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>          <!--xml 文件版本号和编码方式-->
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">          <!--LinearLayout-->
     <Button
         android:id="@+id/Button01"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:background="@drawable/myselectora">
                                                  <!--Button 按钮的背景图片-->
    </Button>                                     <!--Button 控件-->
</LinearLayout>
```

上面已经介绍了本程序的主界面 main.xml 的开发，接下来介绍本程序的 myselectora.xml。

代码位置：见随书光盘中源代码/第 2 章/Sample2_17/ res/drawable-mdpi 目录下的 myselectora.xml。

```
<?xml version="1.0" encoding="utf-8"?>          <!--xml 文件版本号和编码方式-->
<selector
  xmlns:android="http://schemas.android.com/apk/res/android">
  <item
   android:state_pressed="false"                <!--设置项目未被选中 -->
   android:drawable="@drawable/ok2"/>           <!--设置项目的背景 -->
  <item
   android:state_pressed="true"                 <!--设置项目被选中 -->
   android:drawable="@drawable/ok1"/>           <!--设置项目的背景 -->
 </selector>
```

上面已经介绍了本程序的 main.xml 与 myselectora.xml 的开发，接下来为读者介绍本程序具体功能的实现，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_17/src/com/bn/ex2q 目录下的 Sample2_17_Activity.class。

```
package com.bn.ex2q;                                    //声明包
……//该处省略了部分类的导入，读者可自行查看随书光盘中的源代码
import android.widget.Toast;                            //导入相关类
public class Sample2_17_Activity extends Activity{      //继承自 Activity 类
   @Override
   public void onCreate(Bundle savedInstanceState){     //重写的方法
        super.onCreate(savedInstanceState);             //调用父类
        setContentView(R.layout.main);                  //跳转到主界面
        Button but=(Button)this.findViewById(R.id.Button01);
                                                        //Button 按钮的引用
        but.setOnClickListener(                         //设置监听事件
             new OnClickListener(){                     //匿名内部类
                    @Override
                    public void onClick(View v){        //重写的方法
                       Toast.makeText(Sample2_17_Activity.this,
                                                        //弹出 Toast 提示
                                      "ImageButton 单击",
                                      Toast.LENGTH_SHORT).show();
                    }
} ); } }
```

提示：上面的主要作用是对按钮的监听，单击“ImageButton”按钮便改变按钮的背景颜色，并且弹出 Toast 提示。

实例 18　图片按钮的单击变换

【实例描述】

在本小节中，通过对 ImageButton 的应用，构建了一个图片按钮单击变换的小程序。并通过本程序的实现，来向读者介绍 ImageButton 的具体应用。本实例的运行效果图，如图 2-18 所示。

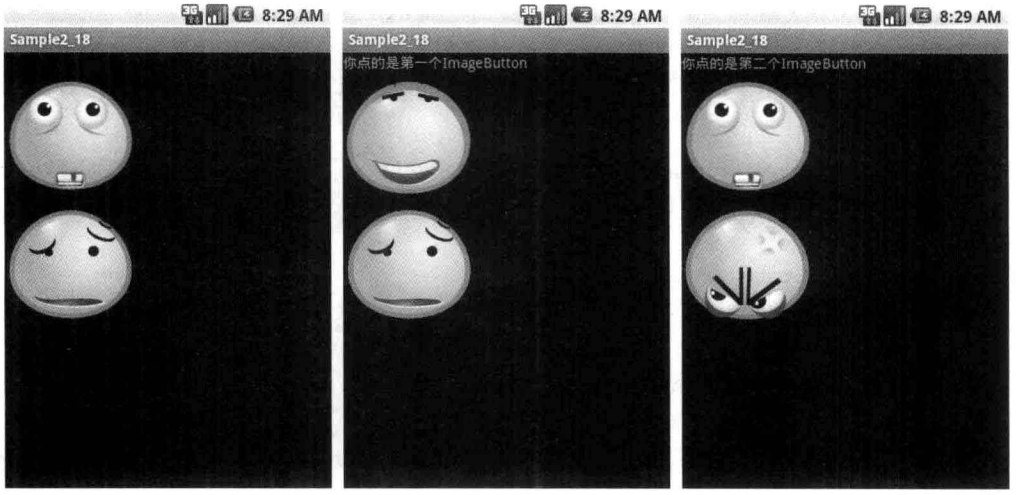


图 2-18　ImageButton 的选择特效效果图

提示：程序运行过程中，单击动画头像可以用另一幅头像替换该头像，如图 2-18 所示。

【实现过程】

本程序主要是对头像按钮设置监听，并且单击头像，更换另一幅图片。

【代码解析】

首先为读者介绍本程序主界面 main.xml 的开发，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_18/ res/layout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>          <!--xml 文件版本号和编码方式-->
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="vertical"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent">         <!--LinearLayout-->
6       <TextView
7           android:id="@+id/TextView01"
8           android:layout_width="wrap_content"
9           android:layout_height="wrap_content">  <!--纵向自适应大小-->
10      </TextView>                                 <!--TextView 控件-->
11      <ImageButton
12          android:id="@+id/ImageButton01"
13          android:layout_width="wrap_content"
14          android:layout_height="wrap_content"
```

```
15          android:background="@drawable/a"> <!--ImageButton 按钮的背景图片-->
16      </ImageButton>                        <!--ImageView-->
17      <ImageButton
18          android:id="@+id/ImageButton02"
19          android:layout_width="wrap_content"
20          android:layout_height="wrap_content"
21          android:background="@drawable/c"> <!--ImageButton 按钮的背景图片-->
22      </ImageButton>                        <!--ImageView-->
23  </LinearLayout>
```

上面已经介绍了本程序主界面 main.xml 的开发，接下来为读者介绍本程序的具体功能的实现，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_18/src/com/bn/ex2r 目录下的 Sample2_18_Activity.class。

```
1   package com.bn.ex2r;                                          //声明包
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中的源代码
3   import android.widget.TextView;                               //导入相关类
4   public class Sample_18_Activity extends Activity{             //继承自 Activity 类
5      @Override
6      public void onCreate(Bundle savedInstanceState){           //重写的方法
7         super.onCreate(savedInstanceState);                     //调用父类
8         setContentView(R.layout.main);                          //跳转到主界面
9         final TextView tv=(TextView)this.findViewById(R.id.TextView01);
                                                                  //得到 TextView 的引用
10        final ImageButton but1=(ImageButton)this.findViewById(R.id.ImageButton01);
                                                                  //ImageButton 引用
11        final ImageButton but2=(ImageButton)this.findViewById(R.id.ImageButton02);
                                                                  //ImageButton 引用
12         but1.setOnClickListener(                               //设置监听事件
13               new OnClickListener(){                           //匿名内部类
14                     @Override
15                     public void onClick(View v){               //重写的方法
16                        tv.setText("你点的是第一个 ImageButton");
                                                                  //设置 TextView 的显示
17                        but1.setImageDrawable                   //设置 ImageButton 图
18                           (getResources().getDrawable(R.drawable.b));
19                        but2.setImageDrawable                   //设置 ImageButton 图
20                           (getResources().getDrawable(R.drawable.c));
21           } } );
22        but2.setOnClickListener(                               //设置监听事件
23               new OnClickListener(){                           //匿名内部类
24                     @Override
25                     public void onClick(View v){               //重写的方法
26                        tv.setText("你点的是第二个 ImageButton");
                                                                  //设置 TextView 的显示
27                        but2.setImageDrawable                   //设置 ImageButton 图
28                           (getResources().getDrawable(R.drawable.d));
29                        but1.setImageDrawable                   //设置 ImageButton 图
30                           (getResources().getDrawable(R.drawable.a));
31           } } );
32  } }
```

其中：

- 第 12～21 行表示的是对第一个 ImageButton 的监听事件，其被单击则更换图片。
- 第 12～21 行表示的是对第二个 ImageButton 的监听事件，其被单击则更换图片。

实例 19　音乐播放的进度提示

【实例描述】

在本小节中，通过对 ProgressBar 的应用，构建了一个提示音乐播放进度的小程序。并通过本程序的实现，来向读者介绍 ProgressBar 的具体应用。本实例的运行效果图，如图 2-19 所示。

图 2-19　ProgressBar 运行效果图

提示：本程序的运行效果如图 2-19 所示，该图依次表示的是运行本程序进入主界面，单击“确定”按钮弹出模拟播放进度条，进度条不断地增加。

【实现过程】

该软件通过对 ProgressBar 的应用，实现了音乐播放进度的提示功能。

【代码解析】

首先为读者介绍本程序的主界面 main.xml 的开发，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_19/ res/layout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>          <!--xml 文件版本号和编码方式-->
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="vertical"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent">
6       <Button
7          android:text="@string/ok"
8          android:id="@+id/Button01"
9          android:layout_width="fill_parent"
10         android:layout_height="wrap_content">    <!--纵向自适应大小-->
11      </Button>                                   <!--Button 控件-->
12  </LinearLayout>
```

上面已经介绍了本程序的主界面 main.xml 的开发，接下来介绍本程序的 string.xml 文件。

代码位置：见随书光盘中源代码/第 2 章/Sample2_19/ res/lvalues 目录下的 string.xml。

```
<?xml version="1.0" encoding="utf-8"?>          <!--xml 文件版本号和编码方式-->
<resources>
    <string name="hello">Hello World, Sample2_19_Activity!</string>
    <string name="app_name">Sample2_19</string>
    <string name="ok">确定</string>              <!--设置文本-->
    <string name="title">音乐播放进度</string>    <!--设置文本-->
</resources>
```

上面已经介绍了本程序的 main.xml 与 string.xml 的开发，接下来为读者介绍的是本程序的具体功能的实现，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_19/src/com/bn/ex2s 目录下的 Sample2_19_Activity.class。

```
package com.bn.ex2s;                                          //声明包
……//该处省略了部分类的导入，读者可自行查看随书光盘中的源代码
import android.widget.Button;                                 //导入相关类
public class Sample2_19_Activity extends Activity{            //继承自 Activity 类
    final int PROGRESS_DIALOG=0;                              //弹出对话框
    final int INCREASE=0;                                     //设置 INCREASE 为 0
    ProgressDialog pd;                                        //PregressDialog 引用
    Handler hd;                                               //Handler 的引用
    @Override
    public void onCreate(Bundle savedInstanceState){          //重写的方法
        super.onCreate(savedInstanceState);                   //调用父类
        setContentView(R.layout.main);                        //跳转到主界面
        Button bok=(Button)this.findViewById(R.id.Button01);//得到 Button 的引用
        bok.setOnClickListener(                               //设置监听事件
          new OnClickListener(){                              //匿名内部类
                  @Override
                  public void onClick(View v){                //重写的方法
                       showDialog(PROGRESS_DIALOG);           //显示对话框
                  }
          } );
        hd=new Handler(){                                     //创建 Handler
          @Override
          public void handleMessage(Message msg){             //重写的方法
              super.handleMessage(msg);                       //调用父类
              switch(msg.what){                               //判断接收的消息
                case INCREASE:                                //接收消息为 INCREASE
                   pd.incrementProgressBy(1);                 //进度每次加 1
                        if(pd.getProgress()>=100){            //判断是否大于 100
                            pd.dismiss();
                        }
                break;                                        //退出
        }}};}
    @Override
    public Dialog onCreateDialog(int id){                     //创建对话框
     switch(id){                                              //判断 id
       case  PROGRESS_DIALOG:                                 //id 为 PROGRESS_DIALOG
          pd=new ProgressDialog(this);                        //创建 ProgressDialog
          pd.setMax(100);                                     //设置最大值
           pd.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
                                                              //设置该对话框样式
           pd.setTitle(R.string.title);                       //设置标题
           pd.setCancelable(false);                           //对话框不能用回退按钮关闭
```

```
42              Log.d("kkk", "kkk");
43            break;                                          //退出
44        }
45        return pd;                                          //返回 pd
46      }
47      @Override
48      public void onPrepareDialog(int id, Dialog dialog){//回调动态更新对话框内容
49        super.onPrepareDialog(id, dialog);                  //调用父类
50        System.out.println("=========onPrepareDialog(int id, Dialog dialog)");
                                                            //后台输出
51            switch(id){                                     //判断 id
52          case PROGRESS_DIALOG:
53            pd.incrementProgressBy(-pd.getProgress());      //对话框进度清零
54            new Thread(){                                   //建立新线程
55                 public void run(){                         //重写的方法
56                   while(true){                             //一直循环
57                     hd.sendEmptyMessage(INCREASE);         //发送消息
58                     if(pd.getProgress()>=100){             //进度条满的情况
59                         break;                             //跳出
60                     }
61                     try{
62                         Thread.sleep(40);                  //休息 40ms
63                     }
64                     catch(Exception e){                    //捕获异常
65                         e.printStackTrace();               //打印堆栈信息
66                     }}}
67            }.start();                                      //开启线程
68          break;
69  }}}
```

其中：

- 第 21～32 行表示创建 Handler 接收消息并判断，根据判断做出相应的措施。
- 第 33～46 行表示创建对话框的方法，在创建对话框时首先执行该方法。
- 第 48～69 行表示每次弹出对话框时，被回调以动态更新对话框内容的方法。

实例 20　音量大小的调节

【实例描述】

在本小节中，通过对 SeekBar 的应用，构建了一个音量大小调节的小程序。并通过本程序的实现，来向读者介绍 SeekBarr 的具体应用。本实例的运行效果图，如图 2-20 所示。

提示：在图 2-20 中，单击“确定”按钮可以显示音乐的进度条。

【实现过程】

该软件通过对 SeekBar 的应用，实现了模拟调节音量大小。

【代码解析】

首先为读者介绍本程序的主界面 main.xml 的开发，代码如下。

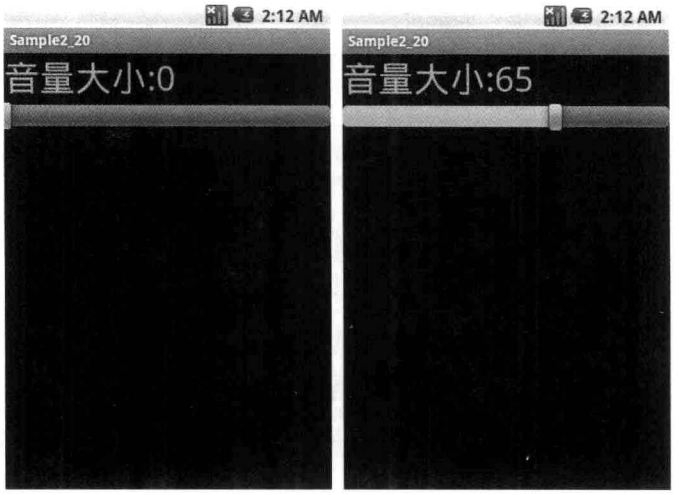


图 2-20　SeekBar 应用效果图

代码位置：见随书光盘中源代码/第 2 章/Sample2_20/ res/layout 目录下的 main.xml。

```
<?xml version="1.0" encoding="utf-8"?>          <!--xml 文件版本号和编码方式-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent">         <!--LinearLayout-->
    <TextView
        android:text="音量大小:0"
        android:id="@+id/TextView01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
    </TextView>                                 <!--TextView 控件-->
    <SeekBar
        android:id="@+id/SeekBar01"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">   <!--纵向自适应大小-->
    </SeekBar>
</LinearLayout>                                 <!--LinearLayout-->
```

上面已经介绍了本程序的主界面 main.xml 的开发，接下来向读者介绍本软件的具体功能的实现，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_20/src/com/bn/ex2t 目录下的 Sample2_20_Activity.class。

```
package com.bn.ex2t;                                    //声明包
……//该处省略了部分类的导入，读者可自行查看随书光盘中的源代码
import android.widget.TextView;                         //导入相关类
public class Sample2_20_Activity extends Activity{      //继承自 Activity 类
    final static double MAX=100;                        //SeekBar 的最大值
    SeekBar sb;                                         //SeekBar 引用
   TextView tv;                                         //TextView 引用
   @Override
   public void onCreate(Bundle savedInstanceState){     //重写的方法
      super.onCreate(savedInstanceState);               //调用父类
      setContentView(R.layout.main);                    //跳转到主界面
      sb=(SeekBar)this.findViewById(R.id.SeekBar01);    //创建 SeekBar 对象
      tv=(TextView)this.findViewById(R.id.TextView01);  //创建 TextView 对象
```

```
14          sb.setOnSeekBarChangeListener(                    //设置监听事件
15              new SeekBar.OnSeekBarChangeListener(){        //匿名内部类
16        @Override
17        public void onProgressChanged(SeekBar seekBar, int progress,  boolean
fromUser){
18            tv.setText("音量大小:"+(int)sb.getProgress());    //设置 TextView 的值
19        }
20        @Override
21        public void onStartTrackingTouch(SeekBar seekBar){ }     //重写的方法
22        @Override
23        public void onStopTrackingTouch(SeekBar seekBar) {}      //重写的方法
24    } ); } }
```

其中：

- 第 5～7 行表示对本程序的成员变量的声明。
- 第 14～24 行表示对本程序的拖拉条的监听，如果被拖动，则在 TextView 中显示信息。

实例 21　为你喜欢的作品打分

【实例描述】

在本小节中，通过对 RatingBar 的应用，构建了一个为作品打分的小程序。并通过本程序的实现，来向读者介绍 RatingBar 的具体应用。本实例的运行效果图，如图 2-21 所示。

图 2-21　RatingBar 运行效果图

提示：在图 2-21 中，程序通过对该打分拖拉条的拖动，显示在 TextView 中的信息。

【实现过程】

本程序通过对 RatingBar 的监听，可以在 TextView 显示用户所打的分数。

【代码解析】

首先为读者介绍本程序的主界面 main.xml 的开发，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_21/ res/layout 目录下的 main.xml。

```
<?xml version="1.0" encoding="utf-8"?>          <!--xml 文件版本号和编码方式-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent">         <!--LinearLayout-->
    <TextView
        android:text="请选择"
        android:id="@+id/TextView01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">  <!--纵向自适应大小-->
    </TextView>                                 <!--TextView 控件-->
    <RatingBar
        android:id="@+id/RatingBar01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">  <!--纵向自适应大小-->
    </RatingBar>
</LinearLayout>
```

上面已经介绍了本程序的主界面 main.xml 的开发，接下来为读者介绍本程序的具体功能的实现，代码如下。

代码位置：见随书光盘中源代码/第 2 章/Sample2_21/src/com/bn/ex2u 目录下的 Sample2_21_Activityp.class。

```
package com.bn.ex2u;                                        //声明包
……//该处省略了部分类的导入，读者可自行查看随书光盘中的源代码
import android.widget.TextView;                             //导入相关类
public class Sample2_21_Activity extends Activity{          //继承自 Activity 类
   RatingBar rb;                                            //声明成员变量
   TextView tv;
   @Override
   public void onCreate(Bundle savedInstanceState){ //重写的方法
      super.onCreate(savedInstanceState);                   //调用父类
      setContentView(R.layout.main);                        //跳转到主界面
      rb=(RatingBar)this.findViewById(R.id.RatingBar01);//创建 RatingBar 对象
      tv=(TextView)this.findViewById(R.id.TextView01);  //创建 TextView 对象
      rb.setOnRatingBarChangeListener(                      //设置监听事件
        new RatingBar.OnRatingBarChangeListener(){          //匿名内部类
        @Override
        public void onRatingChanged(RatingBar ratingBar,//重写的方法
                                float rating,boolean fromUser){
            tv.setText("您的打分为:"+(float)rb.getRating()+"分");
                                                            //设置 TextView 的显示
} } ); } }
```

其中：

- 第 5～6 行表示对本程序的成员变量的声明。
- 第 13～19 行表示对该打分拖拉条的监听，如果发生拖动事件，则在 TextView 中显示相应的输出，并且拖拉条也做出相应的变化，否则拖拉条和 TextView 不发生任何改变。

实例 22　自定义绘制画布

【实例描述】

本节将要介绍的是在 Android 中自定义绘制画布的技巧，通过在画布 Canvas 上绘制出各种各样的颜色来学习绘制画布的技巧，一般情况下都是通过刷新画布来进行图像的重绘。Canvas 画布绘图是基础中的基础，可以在画布上绘制我们想要的任何东西，当然，需要先设置一些关于画布的属性，如画布的颜色，尺寸等。

本实例的运行效果如图 2-22 所示，屏幕界面会根据刷新率不断地刷新屏幕颜色。

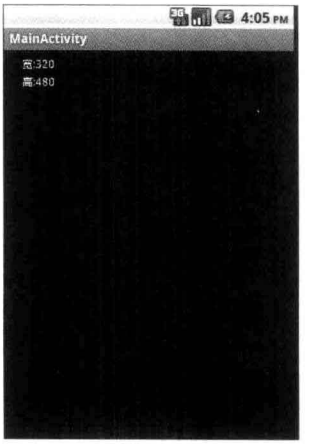


图 2-22　程序运行界面

【实现过程】

首先创建 MainActivity.java，其中 CanvasView 类继承自 SurfaceView 并且引用了 Callback 与 Runnable 接口来刷新线程，绘制的过程是锁定画布，然后调用 draw 方法在画布上进行绘制，接着解锁画布，刷新显示。

【代码解析】

本段为 MainActivity 程序代码，主要实现了整个程序的框架与刷新部分，代码如下。

代码位置：第 2 章\Sample2_22\src\com\example\sample2_22\MainActivity.java

```
1. package com.example.sample16_1;
2. ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3. public class MainActivity extends Activity{
4.   CanvasView canvasView=null;
5.   public void onCreate(Bundle savedInstanceState){
6.       super.onCreate(savedInstanceState);
7.       Display display=getWindowManager().getDefaultDisplay();
8.       canvasView=new CanvasView(this, display.getWidth(), display.getHeight());
9.       setContentView(canvasView);
10.       }
11.
12.  private class CanvasView extends SurfaceView implements Callback,Runnable {
13. ……//该处省略了部分代码，读者可自行查看随书光盘中源代码。
14.           //绘制画布的颜色根据时间变化
15.           public void draw(){
16.                   if (i<100) {
17.                       if (i%2==0) {
18.                       canvas.drawColor(Color.GREEN);
19.                       canvas.drawText("宽:"+mScreenWidth, 20, 20, paint);
20.                       canvas.drawText("高:"+mScreenHeight, 20, 40, paint);
21.                       }else if (i%3==0) {
22.                       canvas.drawColor(Color.GRAY);
23.                       canvas.drawText("宽:"+mScreenWidth, 20, 20, paint);
24.                       canvas.drawText("高:"+mScreenHeight, 20, 40, paint);
25.                       }else if (i%5==0) {
26.                       canvas.drawColor(Color.BLACK);
```

```
27.                         canvas.drawText("宽:"+mScreenWidth, 20, 20, paint);
28.                         canvas.drawText("高:"+mScreenHeight, 20, 40, paint);
29.                         }else if (i%7==0) {
30.                         canvas.drawColor(Color.CYAN);
31.                         canvas.drawText("宽:"+mScreenWidth, 20, 20, paint);
32.                         canvas.drawText("高:"+mScreenHeight, 20, 40, paint);
33.                         }else {
34.                         canvas.drawColor(Color.TRANSPARENT);
35.                         canvas.drawText("宽:"+mScreenWidth, 20, 20, paint);
36.                         canvas.drawText("高:"+mScreenHeight, 20, 40, paint);
37.                         }
38.                         i++;
39.                     }
40.                         if (i>=100) {
41.                             i=0;
42.                         }
43.                 }
44.             public void run(){
45.                 while (isRunning) {
46.                     synchronized (surfaceHolder) {
47.                         canvas=surfaceHolder.lockCanvas();//锁定画布
48.                         draw();
49.                         surfaceHolder.unlockCanvasAndPost(canvas);//解锁画布
50.                     }
51.                     try {
52.                         Thread.sleep(500);
53.                     } catch (InterruptedException e) {
54.                         e.printStackTrace();
55.                     }
56.                 }
57.             }
58. ……//该处省略了部分代码，读者可自行查看随书光盘中源代码。
59. }
```

其中：

- 第 15~43 行为实现画布绘制的 draw()方法的逻辑，绘制画布的颜色会根据时间变化。
- 第 44~57 行为实现 Runnable 接口方法的代码，主要是在其中先锁定画布，然后再利用前面已写的 draw()方法进行绘图，最后解锁画布刷新屏幕。

实例 23　自定义绘制字符串

【实例描述】

本例主要为如何在屏幕中显示字符串的技巧，实际上是通过在画布中绘制各种各样的字符串来进行显示，其中最能体现出其优越性的地方是可以轻松地进行多国语言的切换工作，开发者可以在 string.xml 中设置自己需要用到的字符串，并且进行相关开发，如图 2-23 所示。

【实现过程】

首先创建 MainActivity.java，并且在其中设置输出各种字符串显示要求，通过在画布中使用 draw 方法来绘制字符串进行直观显示，最后一行的字符串还需要从 string.xml 中进行数据调

用，所以也要对 string.xml 添加字符串。

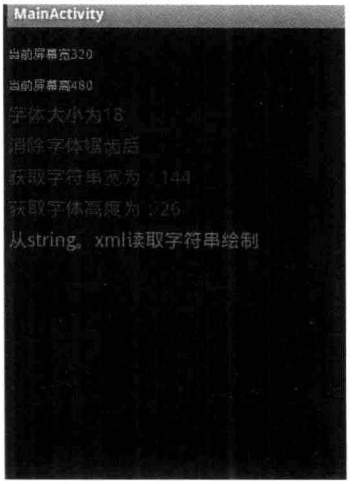


图 2-23　程序运行界面

【代码解析】

代码位置：第 2 章\Sample2_23\src\com\example\sample2_23\MainActivity.java

```
package com.example.sample2_23;
……//该处省略了部分代码，读者可自行查看随书光盘中源代码。
public class MainActivity extends Activity {
    public int mScreenWidth = 0;
    public int mScreenHeight = 0;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        setContentView(new FontView(this));
        // 获取屏幕宽高
        Display display = getWindowManager().getDefaultDisplay();
        mScreenWidth = display.getWidth();
        mScreenHeight = display.getHeight();
        super.onCreate(savedInstanceState);
}
class FontView extends View {
    public final static String STR_WIDTH = "获取字符串宽为：";
    public final static String STR_HEIGHT = "获取字体高度为：";
    Paint mPaint = null;
    Canvas mCanvas=null;
    public FontView(Context context) {
        super(context);
        mPaint = new Paint();
}
    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(mCanvas);
        canvas.drawColor(Color.BLACK);
        //设置字符串颜色
        mPaint.setColor(Color.GREEN);
        canvas.drawText("当前屏幕宽" + mScreenWidth, 0, 30, mPaint);
        canvas.drawText("当前屏幕高"+ mScreenHeight, 0, 60, mPaint);
        //设置字体大小
```

```
            mPaint.setColor(Color.RED);
            mPaint.setTextSize(18);
            canvas.drawText("字体大小为 18", 0, 90, mPaint);
            //消除字体锯齿
            mPaint.setFlags(Paint.ANTI_ALIAS_FLAG);
            canvas.drawText("消除字体锯齿后", 0, 120, mPaint);
            //获取字符串宽度
            canvas.drawText(STR_WIDTH + getStringWidth(STR_WIDTH), 0, 150,
mPaint);
            //获取字体高度
            canvas.drawText(STR_HEIGHT + getFontHeight(), 0, 180, mPaint);
            //从 string.xml 读取字符串绘制 ，可以进行多国语言切换
            mPaint.setColor(Color.GRAY);
            canvas.drawText(getResources().getString(R.string.string_font), 0,
210, mPaint);
    }
    /**
    * 获取字符串宽
    * @param str
    * @return
    */
        private int getStringWidth(String str) {
            return (int) mPaint.measureText(STR_WIDTH);
    }
    /*
    * 获取字体高度
    */
        private int getFontHeight() {
            FontMetrics fm = mPaint.getFontMetrics();
            return (int)Math.ceil(fm.descent - fm.top) + 2;
    }
    }
    }
```

其中：第 25~46 行重写了 onDraw()方法，在其中进行了画笔的各种设置，并且在设置后通过绘制字符串将效果展示出来。

代码位置：第 2 章\Sample2_23\res\values\strings.xml

```
1. <resources>
2.     <string name="app_name">Sample2_23</string>
3.     <string name="hello_world">Hello world!</string>
4.     <string name="menu_settings">Settings</string>
5.     <string name="title_activity_main">MainActivity</string>
6.    <string name="string_font">从 string.xml 读取字符串绘制</string>
7. </resources>
```

实例 24　自定义绘制几何图形

【实例描述】

本例将简单地介绍如何在自定义的画布上绘制简单的几何图形，其中包括绘制线段、绘制矩形、绘制圆形、绘制椭圆以及绘制无规则几何图形（绘制路径）所用到的方法，如图 2-24 所示。

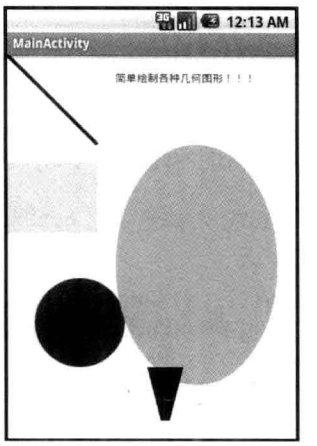


图 2-24　程序运行界面

【实现过程】

首先我们要创建一个 DrawView 类继承自 View 类，并且在 Activity 中通过 setContentView() 方法将其设置为屏幕显示，然后复写 onDraw()方法，通过 drawLine()、drawReft()、drawCircle()、drawOval()、drawPath()方法来绘制我们需要绘制的图形。

【代码解析】

代码位置：第 2 章\Sample2_24\src\com\example\sample2_24\MainActivity.java

```
1. package com.example.sample2_24;
2. ……//该处省略了部分代码，读者可自行查看随书光盘中源代码。
3. public class MainActivity extends Activity {
4.     public int mScreenWidth = 0;
5.     public int mScreenHeight = 0;
6.     @Override
7.     protected void onCreate(Bundle savedInstanceState) {
8.         setContentView(new DrawView(this));
9.         super.onCreate(savedInstanceState);
10.     }
11.
12.     private class DrawView extends View {
13.       Paint mPaint = null;
14.  public DrawView(Context context) {
15.     super(context);
16.     mPaint = new Paint();
17.     mPaint.setFlags(Paint.ANTI_ALIAS_FLAG);
18.  }
19.  @Override
20.  protected void onDraw(Canvas canvas) {
21.     super.onDraw(canvas);
22.     //设置画布颜色 也就是背景颜色
23.     canvas.drawColor(Color.WHITE);
24.     mPaint.setColor(Color.BLACK);
25.     canvas.drawText("简单绘制各种几何图形!!! ", 120, 30, mPaint);
26.     //绘制一条线
27.     mPaint.setColor(Color.BLACK);
```

```
28.     mPaint.setStrokeWidth(4);
29.     canvas.drawLine(0, 0, 100, 100, mPaint);
30.     //绘制一个矩形
31.     mPaint.setColor(Color.YELLOW);
32.     canvas.drawRect(0, 120, 100, 200, mPaint);
33.     //绘制一个圆形
34.     mPaint.setColor(Color.BLUE);
35.     canvas.drawCircle(80, 300, 50, mPaint);
36.     //绘制一个椭圆
37.     mPaint.setColor(Color.CYAN);
38.     canvas.drawOval(new RectF(300,370,120,100), mPaint);
39.     //绘制多边形
40.     mPaint.setColor(Color.BLACK);
41.     Path path = new Path();
42.     path.moveTo(150+5 , 400 - 50);
43.     path.lineTo(150+45, 400 - 50);
44.     path.lineTo(150+30, 460 - 50);
45.     path.lineTo(150+20, 460 - 50);
46.     path.close();
47.     canvas.drawPath(path, mPaint);
48. }
49.     }
50. }
```

其中：第 20~48 行重写了 onDraw()方法，在其中进行了画笔的各种设置，然后通过各种绘制方法绘制所需要绘制的图形。

实例 25　图片绘制的控制技巧

【实例描述】

本节介绍的是如何在屏幕中对图片进行一些基础操作的技巧，主要是通过多个按钮的事件监听来对图片的相关参数进行操作，如图 2-25 可以看到，当我们单击图中按钮的时候，图片可以按我们的意愿来进行“向左移动”、“向右移动”、“左旋转”、“右旋转”、“缩小”、“放大”等操作。

图 2-25　程序运行界面

【实现过程】

我们需要在布局文件中创建六个 Button 与一个 ImageView，然后在 MainActivity.java 中对这 6 个按钮进行监听，每个按钮对应一种方法来操纵 ImageView 的相关属性，通过这样的做法来达到我们需要的效果。

【代码解析】

代码位置：第 2 章\Sample2_25\src\com\example\sample2_25\MainActivity.java

```
1. package com.example.sample2_25;
2. ……//该处省略了部分代码，读者可自行查看随书光盘中源代码。
3. public class MainActivity extends Activity {
4.     ImageView imageView = null;
5.     @Override
6.     protected void onCreate(Bundle savedInstanceState) {
7.   imageView = new ImageView(this);
8.   setContentView(R.layout.activity_main);
9.   LinearLayout ll = (LinearLayout) findViewById(R.id.iamgeid);
10.  ll.addView(imageView);
11.  // 向左移动
12.  Button botton0 = (Button) findViewById(R.id.buttonLeft);
13.  botton0.setOnClickListener(new OnClickListener() {
14.     public void onClick(View arg0) {
15.       imageView.setPosLeft();
16.     }});
17.  // 向右移动
18.  Button botton1 = (Button) findViewById(R.id.buttonRight);
19.  botton1.setOnClickListener(new OnClickListener() {
20.     public void onClick(View arg0) {
21.       imageView.setPosRight();
22.     }});
23.  // 左旋转
24.  Button botton2 = (Button) findViewById(R.id.buttonRotationLeft);
25.  botton2.setOnClickListener(new OnClickListener() {
26.     public void onClick(View arg0) {
27.       imageView.setRotationLeft();
28.     }});
29.  // 右旋转
30.  Button botton3 = (Button) findViewById(R.id.buttonRotationRight);
31.  botton3.setOnClickListener(new OnClickListener() {
32.     public void onClick(View arg0) {
33.       imageView.setRotationRight();
34.     }});
35.  // 缩小
36.  Button botton4 = (Button) findViewById(R.id.buttonNarrow);
37.  botton4.setOnClickListener(new OnClickListener() {
38.     public void onClick(View arg0) {
39.       imageView.setNarrow();
40.     }});
41.  // 放大
42.  Button botton5 = (Button) findViewById(R.id.buttonEnlarge);
43.  botton5.setOnClickListener(new OnClickListener() {
44.     public void onClick(View arg0) {
```

```
45.       imageView.setEnlarge();
46.     }});
47.  super.onCreate(savedInstanceState);
48.    }
49.    class ImageView extends View {
50.  Paint mPaint = null;
51.  Bitmap bitMap = null;
52.  Bitmap bitMapDisplay = null;
53.  int m_posX = 120;
54.  int m_posY = 50;
55.  int m_bitMapWidth = 0;
56.  int m_bitMapHeight = 0;
57.  Matrix mMatrix = null;
58.  float mAngle = 0.0f;
59.  float mScale = 1f;//1 为原图的大小
60.
61.  public ImageView(Context context) {
62. ……//该处省略了部分代码，读者可自行查看随书光盘中源代码。
63.  }
64.  // 向左移动
65.  public void setPosLeft() {
66.     m_posX -= 10;
67.  }
68.  // 向右移动
69.  public void setPosRight() {
70.     m_posX += 10;
71.  }
72.  // 向左旋转
73.  public void setRotationLeft() {
74.     mAngle-=5;
75.     setAngle();
76.  }
77.  // 向右旋转
78.  public void setRotationRight() {
79.     mAngle+=5;
80.     setAngle();
81.  }
82.  // 缩小图片
83.  public void setNarrow() {
84.     if (mScale > 0.5) {
85.       mScale -= 0.1;
86.       setScale();
87.     }
88.  }
89.  // 放大图片
90.  public void setEnlarge() {
91.     if (mScale < 2) {
92.       mScale += 0.1;
93.       setScale();
94.     }
95.  }
96.  // 设置缩放比例
97.  public void setAngle() {
98.     mMatrix.reset();
99.     mMatrix.setRotate(mAngle);
```

```
100.          bitMapDisplay = Bitmap.createBitmap(bitMap, 0, 0, m_bitMapWidth,
101.              m_bitMapHeight, mMatrix, true);
102.      }
103.      // 设置旋转比例
104.      public void setScale() {
105.          mMatrix.reset();
106.          //float sx X轴缩放
107.          //float sy Y轴缩放
108.          mMatrix.postScale(mScale, mScale);
109.          bitMapDisplay = Bitmap.createBitmap(bitMap, 0, 0, m_bitMapWidth,
110.              m_bitMapHeight, mMatrix, true);
111.      }
112.      @Override
113.      protected void onDraw(Canvas canvas) {
114.          super.onDraw(canvas);
115.          canvas.drawBitmap(bitMapDisplay, m_posX, m_posY, mPaint);
116.          invalidate();
117.      }
118.     }
119. }
```

其中：

- 第 11~59 行是我们对按钮实例化，并且对其进行的监听，同时我们在监听后写入了需要触发的相关事件。
- 第 64~111 行是自定义了一些方法，对应我们想要对图片属性进行的一些操作。

第 3 章　Android 高级控件的开发及应用

本章将向读者介绍 Android 平台下的高级控件使用。其中控件主要包括 ListView、GridView、Spinner 自定义下拉菜单、ImageView、Gallery、Dialog 对话框、单选等控件。

实例 1　单击查看名人信息

本节将要介绍如何实现信息列表化以及对列表信息的选择，主要是对 ListView 的应用。

【实例描述】

在本程序中每个名片都由一张图片和一些简单文字说明组成，通过单击相应的名片，会显示出对应名片的详细信息。这是一个很好的保存信息的应用，方便快捷。也可以应用到自己的名片夹以保存信息。本实例的运行效果图，如图 3-1 所示。

图 3-1　程序的运行效果图

提示：本程序的运行效果如图 3-1 所示，在图 3-1 中依次表示的是运行本程序进入程序主界面，在该主界面单击列表中的某一项显示单击的信息。

【实现过程】

本程序应用了 ListView，BaseAdapter，ImageView 和 TextView。首先在 BaseAdapter 中添

加进一个 LinearLayout，再在这个 LinearLayout 中横向加入 ImageView 和 TextView。最后为 ListView 设置内容适配器。

【代码解析】

在本部分会详细介绍本程序的核心代码。首先要介绍的是本程序中 main.xml 文件的设置，其代码如下所示。

代码位置：见本书随书光盘中源代码/第 3 章/Sample3_1/res/layout 目录下的 main.xml。

```
<?xml version="1.0" encoding="utf-8"?>                    <!--版本号和编码方式-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   >                                                      <!--LinearLayout-->
   <TextView
       android:id="@+id/TextView01"
       android:layout_width="fill_parent"
       android:layout_height="wrap_content"
       android:textSize="24dip"
       android:textColor="@color/white"
       android:text="@string/hello"
   />                                                     <!--TextView 控件-->
    <ListView
        android:id="@+id/ListView01"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:choiceMode="singleChoice"
    >
    </ListView>                                           <!-- ListView 控件-->
</LinearLayout>                                           <!--LinearLayout-->
```

提示：上述代码表示的是主界面的搭建，首先控制的是 LinearLayout 的排列方式，然后依次摆放 Textview 与 ListView，并对各项进行设置。

在 xml 文件中搭建界面完成后，随后要做的就是在 Sample3_1_Activity 类中实现本程序单击 ListView 中的每一项，在 TextView 中显示自己选择的功能，代码如下。

代码位置：见本书随书光盘中源代码/第 3 章/Sample3_1/src/com/bn/ex3a 目录下的 Sample3_1_Activity。

```
package com.bn.chap3.lv;                                  //声明包
import android.app.Activity;                              //导入相关包
……//该处省略了部分类的导入代码，读者可自行查看随书光盘中源代码
public class Sample3_1_Activity extends Activity {        //创建继承 Activity 的类
   int[] drawableIds={                                    //所有资源图片 id 的数组
   R.drawable.andy,R.drawable.bill,                       //图片资源的 id
   R.drawable.edgar,R.drawable.torvalds,R.drawable.turing};
   int[] msgIds={                                         //所有字符串 id 的数组
   R.string.andy,R.string.bill,R.string.edgar,            //字符串的 id
   R.string.torvalds,R.string.turing};
   @Override
   public void onCreate(Bundle savedInstanceState) {//继承 Activity 重写的方法
   super.onCreate(savedInstanceState);                    //继承父类
   setContentView(R.layout.main);                         //设置界面
   ListView lv=(ListView)this.findViewById(R.id.ListView01);//初始化 ListView
   BaseAdapter ba=new BaseAdapter(){                      //初始化 BaseAdapter
```

```
17      @Override                                          //确定 ListView 中有几项
18      public int getCount() {return 5;}                  //重写的方法
19      ……该处省略了部分初始化代码，读者可自行查看随书光盘中源代码
20      @Override
21      public View getView(int arg0, View arg1, ViewGroup arg2) {//重写的方法
22         LinearLayout ll=new LinearLayout(Sample3_1_Activity.this);
23         ll.setOrientation(LinearLayout.HORIZONTAL);//设置朝向
24         ll.setPadding(5,5,5,5);                         //设置四周留白
25         ImageView  ii=new ImageView(Sample3_1_Activity.this);
                                                           //初始化 ImageView
26         ii.setImageDrawable(getResources()              //设置图片
27                                  .getDrawable(drawableIds[arg0]));
28         ii.setScaleType(ImageView.ScaleType.FIT_XY);    //设置 ImageView 大小
29         ii.setLayoutParams(new Gallery.LayoutParams(100,98));
30         ll.addView(ii);                                 //添加到 LinearLayout 中
31         TextView tv=new TextView(Sample3_1_Activity.this);//初始化 TextView
32         tv.setText(getResources().getText(msgIds[arg0]));  //设置内容
33         tv.setTextSize(24);                                //设置字体大小
34         tv.setTextColor(Sample3_1_Activity.this.getResources()
35         .getColor(R.color.white));                         //设置字体颜色
36         tv.setPadding(5,5,5,5);                            //设置四周留白
37         tv.setGravity(Gravity.LEFT);                    //tv 在 Gallery 中的位置
38         ll.addView(tv);                                 //添加到 LinearLayout 中
39         return ll;                                      //返回 LinearLayout
40      } };
41      lv.setAdapter(ba);                                 //为 lv 设置内容适配器
42      lv.setOnItemSelectedListener(                      //设置选项选中的监听器
43         new OnItemSelectedListener(){
44         @Override
45         public void onItemSelected(AdapterView<?> arg0,     //重写的方法
46                                    View arg1,int arg2, long arg3) {
47            TextView tv=(TextView)findViewById(R.id.TextView01);
                                                           //得到 TextView 引用
48            LinearLayout ll=(LinearLayout)arg1;
49            TextView tvn=(TextView)ll.getChildAt(1);     //获取其中的 TextView
50            StringBuilder sb=new StringBuilder();        //动态生成信息
51            sb.append(getResources().getText(R.string.ys)); //字符串连接
52            sb.append(":");                              //“。”添加到 sb 中
53            sb.append(tvn.getText());
54            String stemp=sb.toString();                  //sb 转换为字符串
55            tv.setText(stemp.split("\\n")[0]);           //信息设置进 TextView
56         }
57         @Override                                       //未选中选项方法
58         public void onNothingSelected(AdapterView<?> arg0) { }}
                                                           //重写的方法
59      );
60      lv.setOnItemClickListener(                         //设置单击的监听器
61         new OnItemClickListener(){                      //匿名内部类
62         @Override
63         public void onItemClick(AdapterView<?> arg0,    //重写的方法
64              View arg1, int arg2,long arg3) {
65            TextView tv=(TextView)findViewById(R.id.TextView01);
                                                           //主界面 TextView
66            LinearLayout ll=(LinearLayout)arg1;
67            TextView tvn=(TextView)ll.getChildAt(1);     //获取 TextView
68            StringBuilder sb=new StringBuilder();        //动态生成信息
69            sb.append(getResources().getText(R.string.ys));
```

```
                                                        //得到字符串添加到 sb 中
70                  sb.append(":");sb.append(tvn.getText());String stemp=sb.toString();
71                  tv.setText(stemp.split("\\n")[0]);      //信息设置进 tv}}
72          );
73  }}
```

其中：

- 第 5～10 行表示的是初始化图片资源与字符串资源，分别存放在对应的数据中。
- 第 20～40 行表示的是设置 View 的布局，并将图片与字符串以一定的布局方式添加到其中。
- 第 41 行表示的是为 ListView 使用适配器。
- 第 42～56 行表示的是设置获得鼠标焦点的监听器，如果有选项获得鼠标焦点，则在 TextView 中显示对应的信息。
- 第 60～73 行表示的是对单击事件的监听，如果有单击事件发生，则在 ListView 中输出对应的信息。

实例 2　动态图片排版

本节将要介绍如何实现信息表格化以及对表格信息的选择，主要是对 GridView 的应用。

【实例描述】

上一节中的名片看起来有点没有头绪，这一节为读者介绍一种将每张名片分在表格的空间中，这样名片看起来就清晰多了，更有助于管理。本实例的运行效果图，如图 3-2 所示。

图 3-2　程序的运行效果图

提示：本程序的运行效果如图 3-2 所示，在图 3-2 中依次表示的是运行本程序进入程序主界面，在本主界面单击表格中的某一控件，显示单击的信息。

【实现过程】

GridView 是一个类似于表格化的二维排版配置 View，当在 GridView 里面存储的数据在窗

口中无法全部显示时，则会出现滚动条效果。

【代码解析】

在本部分首先介绍的是本程序中的 main.xml 文件的设置，代码如下。

代码位置：见本书随书光盘中源代码/第 3 章/Sample3_2/res/layout 目录下的 main.xml。

```
<?xml version="1.0" encoding="utf-8"?>                    <!--版本号和编码方式-->
<LinearLayout
  android:id="@+id/LinearLayout01"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:orientation="horizontal"
  xmlns:android="http://schemas.android.com/apk/res/android">
                                                          <!--LinearLayout-->
    <!--建一个 ImageView-->
    <ImageView
      android:id="@+id/ImageView01"
      android:scaleType="fitXY"
      android:layout_width="100dip"
      android:layout_height="98dip"
    >                                                     <!--ImageView-->
    </ImageView>
    <!--建一个 TextView-->
    <TextView
       android:id="@+id/TextView02"
       android:layout_width="100dip"
       android:layout_height="wrap_content"
       android:textColor="@color/white"
           android:textSize="24dip"
           android:paddingLeft="5dip"
    >                                                     <!--TextView 控件-->
    </TextView>
    <!--建一个 TextView-->
    <TextView
       android:id="@+id/TextView03"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:textColor="@color/white"
           android:textSize="24dip"
           android:paddingLeft="5dip"
    >                                                     <!--TextView 控件-->
    </TextView>
</LinearLayout>                                           <!--LinearLayout-->
```

提示：上述代码表示的是主界面的搭建，首先设置 LinearLayout，即排列方式，之后依次摆放 ImageView、TextView 与 TextView，并对各项进行详细设置。

在 xml 文件中搭建界面完成后，要做的就是在 Sample3_2_Activity 类中实现本程序单击 GridView 中的每一项，在 TextView 中显示自己选择的功能，代码如下。

代码位置：见本书随书光盘中源代码/第 3 章/Sample3_2/src/com/bn/ex3b 目录下的 Sample3_2_Activity。

```
package com.bn.chap3.gv;                                  //声明包
import java.util.ArrayList;                               //引入相关类
……//该处省略了部分类的导入代码，读者可自行查看随书光盘中源代码
public class Sample3_2_Activity extends Activity {        //创建继承 Activity 的类
int[] drawableIds=                                        //所有图片 id 的数组
```

```
{R.drawable.andy,R.drawable.bill,R.drawable.edgar,          //图片的 id
    R.drawable.torvalds,R.drawable.turing};
int[] nameIds=                                              //所有字符串 id 的数组
{R.string.andy,R.string.bill,R.string.edgar,                //字符串的 id
    R.string.torvalds,R.string.turing};
int[] msgIds=                                               //所有字符串 id 的数组
{R.string.andydis,R.string.billdis,R.string.edgardis,       //字符串的 id
    R.string.torvaldsdis,R.string.turingdis};
public List<? extends Map<String, ?>> generateDataList(){    //方法的声明
    ArrayList<Map<String,Object>> list=new ArrayList<Map<String,Object>>();;
    int rowCounter=drawableIds.length;                      //得到表格的行数
    or(int i=0;i<rowCounter;i++){                           //循环生成对应各个列数据
    HashMap<String,Object> hmap=new HashMap<String,Object>();//HashMap 集合
    hmap.put("col1", drawableIds[i]);                       //第一列为图片
    hmap.put("col2", this.getResources().getString(nameIds[i]));//第二列为姓名
    hmap.put("col3", this.getResources().getString(msgIds[i]));//第三列为描述
    ist.add(hmap);                                          //加入到集合中
}
return list;                                                //返回值
}
@Override
public void onCreate(Bundle savedInstanceState) {//继承 Activity 要重写的方法
super.onCreate(savedInstanceState);                         //调用父类
setContentView(R.layout.main);                              //跳转到主界面
GridView gv=(GridView)this.findViewById(R.id.GridView01); //获得 GridView 的引用
SimpleAdapter sca=new SimpleAdapter(                        //SimpleAdapter 适配器
this,
generateDataList(),                                         //数据 List
R.layout.grid_row, //行对应 layout id
new String[]{"col1","col2","col3"},                         //列名列表
new int[]{R.id.ImageView01,R.id.TextView02,R.id.TextView03}//列对应控件 id 列表
);
    gv.setAdapter(sca);
    gv.setOnItemSelectedListener(                           //设置选项选中的监听器
    new OnItemSelectedListener(){                           //匿名内部类
@Override
public void onItemSelected(AdapterView<?> arg0, View arg1,
    int arg2, long arg3) {                                  //重写选中事件的处理方法
    TextView tv=(TextView)findViewById(R.id.TextView01);//获取主界面 TextView
    LinearLayout ll=(LinearLayout)arg1;
    TextView tvn=(TextView)ll.getChildAt(1);                //获取其中的 TextView
    TextView tvnL=(TextView)ll.getChildAt(2);               //获取其中的 TextView
    StringBuilder sb=new StringBuilder();
    sb.append(tvn.getText());                               //获取姓名信息
    sb.append(" ");
    sb.append(tvnL.getText());                              //获取描述信息
    tv.setText(sb.toString());                              //信息设置
}
@Override
public void onNothingSelected(AdapterView<?> arg0) { }}
                                                        //重写的 onNothingSelected
);
gv.setOnItemClickListener(                                  //设置选项被单击的监听器
new OnItemClickListener(){                                  //匿名内部类
@Override
public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,
    long arg3) {                                            //重写单击事件的处理方法
```

```
62  TextView tv=(TextView)findViewById(R.id.TextView01);      //获取主界面 TextView
63  LinearLayout ll=(LinearLayout)arg1;
64  TextView tvn=(TextView)ll.getChildAt(1);                  //获取其中的 TextView
65  TextView tvnL=(TextView)ll.getChildAt(2);                 //获取其中的 TextView
66  StringBuilder sb=new StringBuilder();
67  sb.append(tvn.getText());                                 //获取姓名信息
68  sb.append(" ");
69  sb.append(tvnL.getText());                                //获取描述信息
70  tv.setText(sb.toString());                                //信息设置
71  }} );
72  }}
```

其中：

- 第 5～13 行表示的是本程序的成员变量的初始化，分别存放对应的数据。
- 第 17～23 行表示的是在循环中创建 HashMap 对象，并存入相应的数据。
- 第 41～53 行表示的是设置获得鼠标焦点的监听器，如果有选项获得鼠标焦点，则在 TextView 中显示对应的信息。
- 第 60～72 行表示的是对单击事件的监听，如果有单击事件发生，则在 ListView 中输出对应的信息。

实例 3　选择喜欢的体育运动

在不同的软件中经常会出现下拉菜单，在其中可以选择自己需要的信息。在本节中将要详细介绍如何使用自定义的下拉菜单，主要是对 Spinner 的应用。

【实例描述】

本程序界面简洁，在运行本程序后，在界面上会有一个下拉菜单，在下拉菜单中可以选择自己喜欢的体育运动。当单击选择某一项后，会在界面的上方显示自己的选择。本实例的运行效果图，如图 3-3 所示。

图 3-3　程序的运行效果图

提示：在图 3-3 中依次表示的是运行本程序进入主界面，在主界面单击下拉列表按钮弹出下拉列表，之后选中下拉列表中的某一项进入主界面，并可以显示自己的选择。

【实现过程】

该界面继承自 Activity，通过初始化 Spinner 并为该 Spinner 设置适配器，单击下拉菜单按钮弹出下拉菜单，并且对该下拉菜单添加监听，单击菜单中的一项，在主界面的上方即可以显示单击项名称。

【代码解析】

在本部分首先介绍的是本程序中的 main.xml 文件的设置，代码如下。

代码位置：见本书随书光盘中源代码/第 3 章/Sample3_3/res/laoyout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>      <!--版本号和编码方式-->
2   <LinearLayout
3     android:id="@+id/LinearLayout01"
4     android:layout_width="fill_parent"
5     android:layout_height="fill_parent"
6     android:orientation="vertical"
7     xmlns:android="http://schemas.android.com/apk/res/android">
                                                <!--LinearLayout-->
8     <TextView
9       android:text="@string/ys"
10      android:id="@+id/TextView01"
11      android:layout_width="fill_parent"
12      android:layout_height="wrap_content"
13      android:textSize="28dip"
14      >
15    </TextView>                               <!--TextView 控件-->
16    <Spinner
17      android:id="@+id/Spinner01"
18      android:layout_width="fill_parent"
19      android:layout_height="wrap_content">   <!-- Spinner 的大小-->
20    </Spinner>
21  </LinearLayout>                             <!--LinearLayout-->
```

提示：上述代码表示本程序的主界面的搭建，首先设置 LinearLayout 的排列方式，然后依次设置 TextView 与 Spinner 的属性。

上面的 main.xml 部分介绍的是本程序的界面的搭建，接下来将要为读者介绍的是实现下拉列表并对其添加监控的功能，代码如下。

代码位置：见本书随书光盘中源代码/第 3 章/Sample3_3/src/com/bn/ex3c 目录下的 Sample3_3_Activity。

```
1   package com.bn.ex3c;                                          //声明包
2   ……//该处省略了部分类的导入代码，读者可自行查阅随书光盘源代码
3   import android.widget.AdapterView.OnItemSelectedListener;     //导入相关类
4   public class Sample3_3_Activity extends Activity {            //继承 Activity 的类
5       final static int WRAP_CONTENT=-2;                         //WRAP_CONTENT 常量
6     int[] drawableIds={R.drawable.football,R.drawable.basketball,R.drawable.
volleyball};
7       int[] msgIds={R.string.zq,R.string.lq,R.string.pq}; //得到 string 的 id
8     @Override
9     public void onCreate(Bundle savedInstanceState) {           //重写的方法
10      super.onCreate(savedInstanceState);                       //调用父类
11      setContentView(R.layout.main);                            //跳转到主界面
12      Spinner sp=(Spinner)this.findViewById(R.id.Spinner01);
                                                                  //得到 Spinner 的引用
```

```
13        BaseAdapter ba=new BaseAdapter(){                    //设置适配器
14              public int getCount() {
15                  return 3;                                    //总共三个选项
16              }
17              public Object getItem(int arg0) { return null; }//重写的方法
18              public long getItemId(int arg0) { return 0; }   //重写的方法
19              public View getView(int arg0, View arg1, ViewGroup arg2) {
                                                                 //重写的方法
20                  LinearLayout ll=new LinearLayout(Sample3_3_Activity.this);
21                  ll.setOrientation(LinearLayout.HORIZONTAL);//设置朝向
22                  ImageView  ii=new ImageView(Sample3_3_Activity.this);
                                                                 //得到 ImageView 的引用
23
ii.setImageDrawable(getResources().getDrawable(drawableIds[arg0]));//设置图片
24                  ll.addView(ii);                              //添加到 LinearLayout 中
25                  TextView tv=new TextView(Sample3_3_Activity.this);
                                                                 //得到 TextView 的引用
26                  tv.setText(" "+getResources().getText(msgIds[arg0]));
                                                                 //设置内容
27                  tv.setTextSize(24);                          //设置字体大小
28                  tv.setTextColor(R.color.black);              //设置字体颜色
29                  ll.addView(tv);                              //添加到 LinearLayout 中
30                  return ll;
31              }
32        };
33        sp.setAdapter(ba);                                     //为 Spinner 设置内容适配器
34        sp.setOnItemSelectedListener(                          //添加监听
35          new OnItemSelectedListener(){                        //匿名内部类
36              public void onItemSelected(AdapterView<?> arg0, View arg1,
37                  int arg2, long arg3) {                       //重写选项被选中的处理方法
38                  TextView tv=(TextView)findViewById(R.id.TextView01);
                                                                 //获取主界面 TextView
39                  LinearLayout ll=(LinearLayout)arg1;//获取选中的 LinearLayout
40                  TextViewtvn=(TextView)ll.getChildAt(1);//获取其中的 TextView
41                  StringBuilder sb=new StringBuilder();//得到 StringBuilder 对象
42                  sb.append(getResources().getText(R.string.ys));
                                                                 //添加信息到 sb 中
43                  sb.append(":");                              //添加信息到 sb 中
44                  sb.append(tvn.getText());                    //添加信息到 sb 中
45                  tv.setText(sb.toString());                   //设置主界面 TextView 信息
46              }
47              public void onNothingSelected(AdapterView<?> arg0) { }
                                                                 //重写的方法
48    });}}
```

其中：

- 第 5～7 行表示对成员变量的声明，并初始化。
- 第 13～32 行表示设置适配器，并自定义显示的下拉列表选项的布局方式。
- 第 33 行表示为下拉列表设置适配器。
- 第 34～48 行表示对下拉列表的监听，判断是否有选项被选中，如果下拉列表中有选项被选中，则在主界面中显示对应的信息。

实例 4　向菜单中添加选项

本节介绍在下拉列表中添加选项，主要是对 ArrayList 以及 Spinner 的应用。

【实例描述】

本程序结构简单，界面由一个下拉列表、可输入的文本框与“确定”按钮组成，在文本框中输入信息单击“确定”按钮，可以将输入的信息添加到下拉列表中，单击下拉列表可以查看添加的选项，但是一次只可以添加一个选项。本实例的运行效果图，如图 3-4 所示。

图 3-4　程序的运行效果图

提示：在图 3-4 中依次表示的是运行本程序进入主界面，在主界面单击下拉列表按钮弹出下拉列表。

【实现过程】

在本程序中控件部分用到了 Spinner、EditText 与 Button，在 EditText 中输入信息，单击“确定”按钮，添加到 ArrayList 中，而 Spinner 中显示的信息为 ArrayList 中的数据。

【代码解析】

在本部分首先介绍本程序中的 main.xml 文件的设置，其代码如下所示。

代码位置：见本书随书光盘中源代码/第 3 章/Sample3_4/res/laoyout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>          <!--版本号和编码方式-->
2   <LinearLayout
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      android:orientation="vertical"
5      android:layout_width="fill_parent"
6      android:layout_height="fill_parent"
7      >                                            <!--LinearLayout-->
8       <Spinner
9           android:id="@+id/Spinner01"
10          android:layout_width="fill_parent"
11          android:layout_height="wrap_content"
12      >
13      </Spinner>
14      <EditText
15          android:id="@+id/EditText01"
16          android:layout_width="fill_parent"
```

```
        android:layout_height="wrap_content"
    >                                              <!--EditText 控件-->
    </EditText>
    <Button
        android:text="增加"
        android:id="@+id/Button01"
        android:layout_width="100dip"
        android:layout_height="wrap_content"
       android:layout_marginLeft="100dip"
    >                                              <!--Button 控件-->
    </Button>
</LinearLayout>                                    <!--LinearLayout-->
```

提示：上述代码表示本程序的主界面的搭建，首先需要设置控件的摆放顺序，然后按照摆放顺序一次排列 Spinner、EditText 与 Button 控件。

上面已经介绍了本程序的主界面的搭建，接下来将要介绍的是实现下拉列表中添加选项的具体功能，代码如下。

代码位置：见本书随书光盘中源代码/第 3 章/Sample3_4/src/com/bn/ex3d 目录下的 Sample3_4_Activity。

```
package com.bn.ex3d;                                   //声明包
import java.util.ArrayList;                            //导入相关类
……//该处省略了部分类的导入代码，读者可自行查阅随书光盘源代码
import android.widget.Toast;                           //导入相关类
public class Sample3_4_Activity extends Activity{      //创建继承 Activity 的类
    ArrayList<String> alItem=new ArrayList<String>();    //创建 ArrayList 对象
     @Override
     public void onCreate(Bundle savedInstanceState){    //继承 Activity 时重写
         super.onCreate(savedInstanceState);               //调用父类
         alItem.add("游泳");                           //在 ArrayList 中添加数据
         alItem.add("篮球");                           //在 ArrayList 中添加数据
         alItem.add("足球");                           //在 ArrayList 中添加数据
         setContentView(R.layout.main);                //切屏到主界面
         Spinner sp=(Spinner)this.findViewById(R.id.Spinner01);
                                                       //得到 Spinner 的引用
         BaseAdapter ba=new BaseAdapter(){             //设置适配器
          public int getCount(){
                 return alItem.size();                 //总共三个选项
          }
          public Object getItem(int arg0) { return null;}    //重写的方法
          public long getItemId(int arg0) { return 0; }      //重写的方法
          public View getView(int arg0, View arg1, ViewGroup arg2){
                                                             //重写的方法
          LinearLayout ll=new LinearLayout(Sample3_4_Activity.this);
                                                       //创建 LinearLayout 的引用
             ll.setOrientation(LinearLayout.HORIZONTAL);     //设置排列方向
             TextView tv=new TextView(Sample3_4_Activity.this);
             tv.setText(alItem.get(arg0));                   //设置内容
             tv.setTextSize(24);                             //设置字体大小
             tv.setTextColor(R.color.black);                 //设置字体颜色
             ll.addView(tv);                           //添加到 LinearLayout 中
             return ll;
            }
         };
         sp.setAdapter(ba);                            //为 Spinner 设置内容适配器
```

```
33          Button addbutton01=(Button)findViewById(R.id.Button01);
                                                          //得到 Button 的引用
34           addbutton01.setOnClickListener(              //为按钮添加监听器
35               new OnClickListener(){                   //匿名内部类
36                    public void onClick(View v){        //重写的方法
37                         EditText edit01=(EditText)findViewById(R.id.EditText01);
38                         String edit=edit01.getText().toString();
                                                          //得到文本框中的内容
39                         alItem.add(edit);          //ArrayList 中添加文本框内容
40                         Toast.makeText(Sample3_4_Activity.this,  //弹出 Toasst
41                                   edit, Toast.LENGTH_SHORT).show();
42  } } ); } }
```

其中：

- 第 10～13 行表示在 ArrayList 中添加数据。
- 第 15～31 行表示设置 BaseAdapter 适配器，并自定义显示的下拉列表选项的布局方式。
- 第 32 行表示为下拉列表添加适配器。
- 第 34～42 行表示为按钮添加监听器，当单击“确定”按钮时将得到的输入文本框的数据添加到 ArrayList 中，同时弹出 Toast。

实例 5　单击改变图片透明度

在本节中将要介绍的是单击图片改变图片的透明度的值，主要是对 ImageView 的应用。

【实例描述】

运行本程序，首先显示的是一幅透明度 100 的图片，再单击该图片区域内的任意处，将改变该图片的透明度，并在界面显示更改透明度后的图片。本实例的运行效果图，如图 3-5 所示。

图 3-5　程序运行后的效果图

提示：在图 3-5 中依次表示的是运行本程序进入主界面，在主界面中单击图片区域改变图片透明度。

【实现过程】

在本程序中，所用到的控件主要是 ImageView，并且为该图片添加监听，当单击图片区域的任意一处时，更改图片的透明度为 255。

【代码解析】

在本部分首先介绍的是本程序中 main.xml 文件的设置，其代码如下。

代码位置：见本书随书光盘中源代码/第 3 章/Sample3_5/res/laoyout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>                    <!--版本号和编码方式-->
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="vertical"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent">                   <!--LinearLayout-->
6      <ImageView
7       android:id="@+id/ImageView01"
8       android:layout_width="wrap_content"
9       android:layout_height="wrap_content"
10      android:paddingLeft="30dip"
11      android:paddingRight="30dip"
12      android:visibility="visible"
13      >
14      </ImageView>                                           <!--ImageView--
15  </LinearLayout>                                            <!--LinearLayout-->
```

提示：上述代码主要是主界面的搭建，首先需要设置 LinearLayout 的摆放顺序，然后需要对 ImageView 的属性进行设置。

上面已经对本程序的主界面做了详细的介绍，接下来介绍的是本程序功能的实现，代码如下。

代码位置：见本书随书光盘中源代码/第 3 章/Sample3_5/src/com/bn/ex3e 目录下的 Sample3_5_Activity。

```
1   package com.bn.ex3e;                                       //声明包
2   ……//该处省略了部分类的导入代码，读者可自行查阅随书光盘源代码
3   import android.widget.ImageView;                           //导入相关类
4   public class Sample3_5_Activity extends Activity{          //创建继承 Activity 的类
5        ImageView iv;                                         //ImageView 的引用
6      @Override
7      public void onCreate(Bundle savedInstanceState){ //继承 Activity 重写的方法
8         super.onCreate(savedInstanceState);                 //调用父类
9         setContentView(R.layout.main);                      //跳转到主界面
10        iv=(ImageView)findViewById(R.id.ImageView01); //得到 ImageView 对象
11        iv.setImageDrawable(getResources().getDrawable(R.drawable.bg));
                                                              //为 ImageView 设置图片
12        iv.setAlpha(100);                                   //设置不透明度为 100
13        iv.setOnClickListener(                              //对该图片添加监听
14              new OnClickListener(){                        //匿名内部类
15                      @Override
16                      public void onClick(View v){   //重写的方法
17                          iv.setImageDrawable(getResources().getDrawable
(R.drawable.bg));
18                          iv.setAlpha(255);           //设置不透明度为 255
19  }});}}
```

提示：该类的主要作用是跳转到主界面，并且对主界面中的图片进行监听，如果在图片区域内有单击事件发生，则设置图片的透明度为 255。

实例 6　动态改变图片大小

在一些软件中经常用到图片放大或者缩小的功能，使图片达到较为理想的视觉效果。在本程序中将实现这项功能，主要是对 ImageView 这个控件的应用。

【实例描述】

运行该程序进入程序主界面，在主界面中单击“放大”按钮，图片会增大；单击“缩小”按钮，图片会变小。本实例的运行效果图，如图 3-6 所示。

图 3-6　程序运行效果图

提示：在图 3-6 中依次表示的是运行本程序进入主界面，在主界面单击“放大”按钮将该图片放大，单击“缩小”按钮使该图片缩小。

【实现过程】

在本程序中用到的主要控件有 ImageView 与 Button，通过对放大和缩小两个按钮的监听，使图片等比例地放大至原图片的 1.2 倍或者缩小至原图片的 0.8 倍。

【代码解析】

首先介绍的是本程序中 main.xml 文件的设置，其代码如下。

代码位置：见本书随书光盘中源代码/第 3 章/Sample3_6/res/laoyout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>                <!--版本号和编码方式-->
2   <FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
3       android:orientation="vertical"
4       android:layout_width="fill_parent"
```

```
    android:layout_height="fill_parent"
    >
    <LinearLayout
     android:id="@+id/LinearLayout01"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     android:orientation="horizontal"
     android:gravity="left|bottom"
     >                                                    <!--LinearLayout-->
         <Button
         android:text="缩小"
         android:id="@+id/Button01"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         >
         </Button>                                        <!--Button 控件-->
         <Button
         android:text="放大"
         android:id="@+id/Button02"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         >
         </Button>                                        <!--Button 控件-->
     </LinearLayout>
    <LinearLayout
     android:id="@+id/LinearLayout03"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     >                                                    <!--LinearLayout-->
         <ImageView
         android:id="@+id/ImageView01"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         >
         </ImageView>                                     <!--ImageView-->
     </LinearLayout>                                      <!-- ListView 控件-->
</FrameLayout>
```

提示：上述 main.xml 部分主要是主界面的搭建，总的布局方式为帧布局，摆放方向为横向，然后设置 LinearLayout 布局，并在该布局中设置两个 Button 按钮的位置。最后设置另一个 LinearLayout 布局，并在该布局中设置 ImageView 的具体属性。

上面已经介绍了本程序主界面的搭建，接下来将要介绍的是单击“放大”或者“缩小”按钮，实现图片的放大和缩小的功能，代码如下。

代码位置：见本书随书光盘中源代码/第 3 章/Sample3_6/src/com/bn/ex3f 目录下的 Sample3_6_Activity。

```
package com.bn.ex3f;                                    //声明包
……//该处省略了部分类的导入代码，读者可自行查阅随书光盘源代码
import android.widget.ImageView;                        //导入相关类
public class Sample3_6_Acitvity extends Activity{       //创建继承 Activity 的类
    ImageView iv;                                       //声明 ImageView 的引用
    Bitmap bmp;                                         //声明 Bitmap 的引用
    int screenWidth;                                    //屏幕宽度
    int secrenHeight;                                   //屏幕的高度
    float scaleWidth=1f;                                //宽度比例
    float scaleHeight=1f;                               //高度比例
```

```
11      @Override
12      public void onCreate(Bundle savedInstanceState){        //重写的方法
13         super.onCreate(savedInstanceState);                  //调用父类
14         setContentView(R.layout.main);                       //切屏到主界面
15         bmp=BitmapFactory.decodeResource(getResources(), R.drawable.pic);
                                                                //创建 Bitmap 对象
16         DisplayMetrics dm=new DisplayMetrics();              //创建矩阵
17         getWindowManager().getDefaultDisplay().getMetrics(dm);
18         screenWidth=dm.widthPixels;                          //得到屏幕的宽度
19         screenWidth=dm.heightPixels-80;                      //得到屏幕的高度
20         iv=(ImageView)findViewById(R.id.ImageView01);        //ImageView 控件
21         Button b1=(Button)findViewById(R.id.Button01);       //缩小按钮
22         Button b2=(Button)findViewById(R.id.Button02);       //放大按钮
23         iv.setImageBitmap(bmp);                          //为 ImageView 设置图片
24         b1.setOnClickListener(                               //为缩小按钮添加监听
25               new OnClickListener(){                         //匿名内部类
26                     @Override
27                     public void onClick(View v){             //重写的方法
28                         iv.setImageBitmap(scaleToFit(bmp,0.8f));    //缩小
29                         bmp=scaleToFit(bmp,0.8f);           //图片变为原图的 0.8 倍
30           }});
31         b2.setOnClickListener(                               //为放大按钮添加监听
32               new OnClickListener(){                         //匿名内部类
33                     @Override
34                     public void onClick(View v){             //重写的方法
35                         iv.setImageBitmap(scaleToFit(bmp,1.2f));    //放大
36                         bmp=scaleToFit(bmp,1.2f);           //图片变为原图的 1.2 倍
37       }});}
38      public static Bitmap scaleToFit(Bitmap bm,float scale){
39       int width = bm.getWidth();                             //图片宽度
40       int height = bm.getHeight();                           //图片高度
41       Matrix matrix = new Matrix();                          //创建矩阵
42       matrix.postScale(scale, scale);                        //图片等比例缩小
43       Bitmap bmResult = Bitmap.createBitmap(bm, 0, 0, width, height, matrix,
true);//声明位图
44         return bmResult;                                     //返回图片
45   }}
```

其中：

- 第 24～30 行表示对“缩小”按钮添加监听的方法，如果单击的是“缩小”按钮，则绘制缩小的图片，并将图片缩小至原图片的 0.8 倍。
- 第 31～37 行表示对“放大”按钮添加监听的方法，如果单击的是“放大”按钮，则绘制放大的图片，并将图片放大至原图片的 1.2 倍。
- 第 38～45 行表示实现图片的放大或者缩小，主要是通过调用 Bitmap.createBitmap()方法乘以相应的矩阵，从而达到改变图片大小的效果。

实例 7　旋转图片的技巧

在本节中将为读者讲述旋转图片的实现，主要是 ImageView 与 Matrix 的应用。

【实例描述】

在本程序运行后进入主界面，在主界面中单击“向右转”按钮即可实现图片向右转，单击

“向左转”按钮即可实现图片向左转。本实例的运行效果图，如图 3-7 所示。

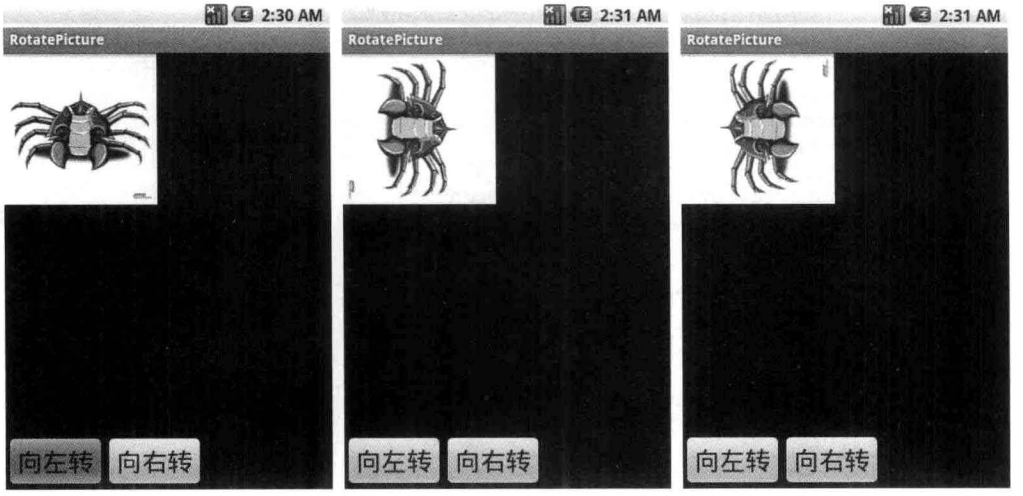


图 3-7　程序运行效果图

提示：在图 3-7 中依次表示的是运行本程序进入主界面，在主界面单击“向右转”按钮图片向右旋转，单击“向左转”按钮图片向左旋转。

【实现过程】

在本程序中用到的主要控件有 ImageView 与 Button，通过“向右转”和“向左转”两个按钮的监听，当单击按钮后生成矩阵，再调用 Bitmap.createBitmap()方法，即可以实现图片的旋转。

【代码解析】

首先介绍的是本程序中 main.xml 文件的设置，其代码如下。

代码位置：见本书随书光盘中源代码/第 3 章/Sample3_7/res/laoyout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>          <!--版本号和编码方式-->
2   <FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="vertical"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent"
6      >                                            <!--FrameLayout-->
7      <LinearLayout
8       android:id="@+id/LinearLayout01"
9       android:layout_width="fill_parent"
10      android:layout_height="fill_parent"
11      android:orientation="horizontal"
12      android:gravity="left|bottom"
13      >                                            <!--LinearLayout-->
14          <Button
15          android:text="向左转"
16          android:id="@+id/Button01"
17          android:layout_width="wrap_content"
18          android:layout_height="wrap_content"
19          >
20          </Button>                                <!--Button 控件-->
21          <Button
```

```
22          android:text="向右转"
23          android:id="@+id/Button02"
24          android:layout_width="wrap_content"
25          android:layout_height="wrap_content"
26          >                                          <!--Button 控件-->
27          </Button>
28      </LinearLayout>
29      <LinearLayout
30       android:id="@+id/LinearLayout03"
31       android:layout_width="wrap_content"
32       android:layout_height="wrap_content"
33       >                                          <!--LinearLayout-->
34          <ImageView
35          android:id="@+id/ImageView01"
36          android:layout_width="wrap_content"
37          android:layout_height="wrap_content"
38          >
39          </ImageView>                           <!--ImageView-->
40      </LinearLayout>                            <!--LinearLayout-->
41  </FrameLayout>
```

提示：上述 main.xml 部分主要是主界面的搭建，总的布局方式为帧布局，摆放方向为横向，然后设置 LinearLayout 布局，并在该布局中设置两个 Button 按钮的位置。最后设置另一个 LinearLayout 布局，并在该布局中设置 ImageView 的具体属性。

上面已经介绍了本程序主界面的搭建，接下来将要介绍的是单击“向右转”或者“向左转”按钮，实现图片的向右转和向左转的功能，代码如下。

代码位置：见本书随书光盘中源代码/第 3 章/Sample3_7/src/com/bn/ex3g 目录下的 Sample3_7_Activity。

```
1   package com.bn.ex3g;                                    //声明包
2   ……//该处省略了部分类的导入代码，读者可自行查阅随书光盘源代码
3   import android.widget.ImageView;                        //导入相关类
4   public class Sample3_7_Activity extends Activity{       //创建继承 Activity 的类
5       ImageView iv;                                       //声明 ImageView
6       Bitmap bmp;                                         //声明 Bitmap
7       int screenWidth;                                    //屏幕宽度
8       int secrenHeight;                                   //屏幕高度
9       float scaleWidth=1f;                                //屏幕比例
10      float scaleHeight=1f;                               //屏幕高度
11    @Override
12    public void onCreate(Bundle savedInstanceState){ //继承 Activity 重写的方法
13       super.onCreate(savedInstanceState);                //调用父类
14       setContentView(R.layout.main);                     //跳转到主界面
15       bmp=BitmapFactory.decodeResource(getResources(), R.drawable.pic);
                                                            //创建 Bitmap
16       DisplayMetrics dm=new DisplayMetrics();            //创建显示矩阵
17       getWindowManager().getDefaultDisplay().getMetrics(dm);
18       screenWidth=dm.widthPixels;                        //得到该显示矩阵的宽度
19       screenWidth=dm.heightPixels-80;                    //得到该显示矩阵的高度
20       iv=(ImageView)findViewById(R.id.ImageView01); //拿到 ImageView
21       Button b1=(Button)findViewById(R.id.Button01);//左转按钮
22       Button b2=(Button)findViewById(R.id.Button02);//右转按钮
23       iv.setImageBitmap(bmp);                            //为 ImageView 设置图片
24       b1.setOnClickListener(                             //向左转添加监听
25             new OnClickListener(){                       //匿名内部类
```

```
26                      @Override
27                      public void onClick(View v) {          //重写的方法
28                          iv.setImageBitmap(rotateToFit(bmp,-90f));   //左转
29                          bmp=rotateToFit(bmp,-90f);//调用 rotateToFit 方法
30          }});
31        b2.setOnClickListener(                              //向右转添加监听
32              new OnClickListener(){                        //匿名内部类
33                      @Override
34                      public void onClick(View v) { //重写的方法
35                          iv.setImageBitmap(rotateToFit(bmp,90f));    //右转
36                          bmp=rotateToFit(bmp,90f); //调用 rotateToFit 方法
37      }});}
38      public static Bitmap rotateToFit(Bitmap bm,float degrees){
39        int width = bm.getWidth();                          //图片宽度
40        int height = bm.getHeight();                        //图片高度
41        Matrix matrix = new Matrix();                       //得到 Matrix 引用
42        matrix.postRotate(degrees);                         //矩阵旋转 degrees 角度
43        Bitmap bmResult = Bitmap.createBitmap(bm, 0, 0, width, height, matrix,
true);//声明位图
44        return bmResult;
45    }}
```

其中：

- 第 5～10 行表示本程序的成员变量的声明。
- 第 31～37 行表示“向左转”按钮的监听，单击“向左转”按钮，使对应的矩阵向左旋转相应的角度，并使图片向左旋转相应的角度。
- 第 24～30 行表示“向右转”按钮的监听，单击“向右转”按钮，使对应的矩阵向右旋转相应的角度，并使图片向右旋转相应的角度。
- 第 38～45 行表示在创建 Matrix 对象的同时使矩阵旋转一定的角度，之后设置位图。

实例 8　制作自己的相片集

当今大部分的手机都具有照相功能，但是其对相片的管理还是停留在单个相片的管理上，很少有实现相片集管理的功能，而本节则为读者介绍相片集管理的实现，主要是对 Gallery 的应用。

【实例描述】

运行本程序进入主界面，在主界面中可以拖动图片，从而实现图片的移动，方便用户的查看。本实例的运行效果图，如图 3-8 所示。

提示：在图 3-8 中依次表示的是运行本程序进入主界面，在主界面向左滑动屏幕图片随之向左移动，之后再次滑动屏幕图片随之向左移动。

【实现过程】

该界面继承自 Activity，在本程序中通过创建适配器，开发适配器，为 Gallery 使用适配器，并对 Gallery 添加监听，从而达到相片集的管理。

【代码解析】

下面介绍实现界面搭建的 main.xml 文件的设置，其代码如下。

图 3-8　程序的运行效果图

代码位置：见本书随书光盘中源代码/第 3 章/Sample3_8/res/laoyout 目录下的 main.xml。

```
<?xml version="1.0" encoding="utf-8"?>          <!--版本号和编码方式-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:gravity="center_vertical"
   >                                            <!--LinearLayout-->
  <Gallery
    android:id="@+id/Gallery01"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:spacing="10dip"
    android:unselectedAlpha="1"
  >
  </Gallery>
</LinearLayout>                                 <!--LinearLayout-->
```

提示：上面介绍了本程序主界面的搭建，首先是 LinearLayout 并设置其摆放方式为竖直摆放，然后是设置 Gallery 的属性，主要是设置图片间隔与未选中条目透明度的值。

上面已经介绍了本程序的主界面的搭建，接下来将要介绍如何实现相片集管理功能，代码如下。

代码位置：见本书随书光盘中源代码/第 3 章/Sample3_8/ src/com/bn/ex3h 目录下的 Sample3_8_Activity。

```
package com.bn.ex3h;                                    //声明包
……//该处省略了部分类的导入代码，读者可自行查阅随书光盘源代码
import android.widget.AdapterView.OnItemClickListener;  //导入相关类
public class Sample3_8_Activity extends Activity{       //继承 Activity 的类
    int[] imageIDs={                                    //int 数组
            R.drawable.a,R.drawable.b,R.drawable.c,     //得到图片的 id
            R.drawable.d,R.drawable.e,R.drawable.f,     //得到图片的 id
            R.drawable.g
    };
  @Override
  public void onCreate(Bundle savedInstanceState){//继承 Activity 需重写的方法
```

```
12          super.onCreate(savedInstanceState);                    //调用父类
13          setContentView(R.layout.main);                         //切屏到主界面
14          Gallery gl=(Gallery)this.findViewById(R.id.Gallery01); //初始化 Gallery
15          BaseAdapter ba=new BaseAdapter(){                      //创建适配器
16                @Override
17                public int getCount() {
18                    return imageIDs.length;                      //长度为数组长度
19                }
20                @Override
21                public Object getItem(int arg0){return null;}
                                                                   //创建适配器需重写的方法
22                @Override
23                public long getItemId(int arg0){return 0;}//创建适配器需重写的方法
24                @Override
25                public View getView(int arg0, View arg1, ViewGroup arg2){
                                                                   //创建适配器需重写的方法
26                    ImageView iv = new ImageView(Sample3_8_Activity.this);
                                                                   //创建 ImageView
27                    iv.setImageResource(imageIDs[arg0]);   //设置图片来源
28                    iv.setScaleType(ImageView.ScaleType.FIT_XY);//设置比例类型
29                    iv.setLayoutParams(new Gallery.LayoutParams(188,250));
30                    return iv;
31                }
32          };
33          gl.setAdapter(ba);                                     //添加适配器
34          gl.setOnItemClickListener(                             //添加监听
35                new OnItemClickListener(){                       //匿名内部类
36                      @Override
37                      public  void  onItemClick(AdapterView<?>  arg0,  View
arg1,int arg2, long arg3){
38                          Gallery gl=(Gallery)findViewById(R.id.Gallery01);
                                                                   //初始化 Gallery
39                          gl.setSelection(arg2);     //设置选择项
40    }});}}
```

其中：

- 第 5～9 行表示 int 数组的初始化，主要存放图片的 id。
- 第 15～32 行表示创建适配器，并且设置布局。
- 第 33 行表示为 Gallery 设置适配器。
- 第 34～40 行表示为 Gallery 添加监听事件，创建匿名内部类并重写方法。

实例 9　重要消息提醒

手机在某种特定的情况下需要弹出重要消息对话框，从而提醒使用者。在本节中将要介绍的就是重要消息对话框的应用，主要是 AlertDialog 的应用。

【实例描述】

运行本程序进入主界面，单击“按钮”可以弹出 AlertDialog 对话框，再单击该 AlertDialog 对话框中的“确定”按钮可以返回主界面。本实例的运行效果图，如图 3-9 所示。

提示：在图 3-9 中依次表示的是运行本程序进入主界面，在主界面单击“确定”按钮弹出消息提示对话框，起到提醒用户的作用。

【实现过程】

本程序界面继承自 Activity，在界面添加一个 Button 按钮，并为该按钮添加监听器，当单击该按钮时弹出 AlertDialog 对话框，并对该对话框中的“确定”按钮添加监听，单击“确定”按钮可以返回主界面。

图 3-9　程序的运行效果图

【代码解析】

下面介绍本程序主界面搭建 main.xml 文件的设置，其代码如下。

代码位置：见本书随书光盘中源代码/第 3 章/Sample3_9/res/laoyout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>          <!--版本号和编码方式-->
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="horizontal"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent"
6      android:gravity="center_horizontal">
7       <Button
8       android:text="按钮"
9       android:id="@+id/Button01"
10      android:layout_width="fill_parent"
11      android:layout_height="wrap_content"
12      >
13      </Button>                                   <!--TextView 控件-->
14  </LinearLayout>                                 <!--LinearLayout-->
```

提示：上述 main.xml 的代码主要是搭建程序开始运行的主界面，首先是设置 LinearLayout 的摆放方式为水平摆放，然后是 Button 按钮的详细设置。

上面已经介绍了本程序主界面的搭建，接下来将要介绍如何实现单击“按钮”弹出 AlertDialog 对话框，并且在该对话框中单击“确定”按钮，返回主界面的功能，代码如下。

代码位置：见本书随书光盘中源代码/第 3 章/Sample3_9/src/com/bn/ex3i 目录下的 Sample3_9_Activity。

```
1   package com.bn.ex3i;                            //声明包
```

```
2    ……//该处省略了部分类的导入代码，读者可自行查阅随书光盘源代码
3    import android.widget.Button;                          //导入相关类
4    public class Sample3_9_Activity extends Activity{      //创建继承 Activity 的类
5       @Override
6       public void onCreate(Bundle savedInstanceState){//继承 Activity 需重写的方法
7          super.onCreate(savedInstanceState);             //调用父类
8          setContentView(R.layout.main);                  //切屏到主界面
9          Button b=(Button)findViewById(R.id.Button01); //得到 Button 按钮
10         b.setOnClickListener(                           //添加监听
11                new OnClickListener(){                   //匿名内部类
12                   @Override
13                   public void onClick(View v){          //重写的方法
14                      new AlertDialog.Builder(Sample3_9_Activity.this)
                                                           //创建 AlertDialog
15                      .setTitle("消息提示")               //设置标题
16                      .setMessage("这时一个 AlertDialog!") //设置消息内容
17                      .setPositiveButton(                //设置 Button 按钮
18                            "确定",
19                            new DialogInterface.OnClickListener(){
                                                           //设置单击事件
20                               @Override
21                               public void onClick(DialogInterface dialog,
int which){
22                                  //在这里设计当对话按钮单击之后要运行的事件
23                               }
24                            }
25                      ).show();
26   }});}}
```

其中：

- 第 10～25 行表示对按钮添加监听，当单击“按钮”时弹出对话框。
- 第 14～24 行表示对话框的设置，设置其标题、信息以及为其设置一个 Button 按钮，并对该按钮添加监听，判断是否发生单击事件。

实例 10　后台程序安装进度提示

在安装程序时，经常会看到安装进度条，在本节中将模拟实现程序安装进度条，主要是 ProgressDialog 的应用。

【实例描述】

运行本程序后进入程序主界面，在主界面单击“确定”按钮，将弹出 ProgressDialog 对话框，模拟并显示程序的安装进度，当模拟安装进度结束后回到主界面。本实例的运行效果图，如图 3-10 所示。

提示：在图 3-10 中依次表示的是运行本程序进入主界面，在主界面单击“确定”按钮弹出安装进度对话框，便可以查看安装的进度。

【实现过程】

本程序界面继承自 Activity，为主界面添加 Button 按钮，为其添加监听，当单击该 Button

按钮时，显示安装进度对话框。同时建立一个新线程，发送消息使得安装进度时时更新，并且在发送消息前需要判断该模拟程序安装是否已经完成。

图 3-10　程序运行效果图

【代码解析】

下面介绍实现本程序主界面搭建 main.xml 文件的设置，其代码如下。

代码位置：见本书随书光盘中源代码/第 3 章/Sample3_10/res/laoyout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>          <!--版本号和编码方式-->
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="vertical"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent"
6      >                                            <!--LinearLayout-->
7     <Button
8       android:text="@string/ok"
9       android:id="@+id/Button01"
10      android:layout_width="wrap_content"
11      android:layout_height="wrap_content">
12    </Button>                                     <!--Button 控件-->
13  </LinearLayout>                                 <!--LinearLayout-->
```

提示：上述 main.xml 代码主要是搭建本程序主界面，首先需要设置 LinearLayout 的排列方式为竖直排列，然后为该界面添加一个 Button 按钮，并设置其属性。

上面已经介绍了本程序的主界面的搭建，接下来将要介绍如何实现单击按钮弹出 ProgressDialog 对话框，并且显示模拟安装进度功能，代码如下。

代码位置：见本书随书光盘中源代码/第 3 章/Sample3_10/src/com/bn/ex3j 目录下的 Sample3_10_Activity。

```
1   package com.bn.ex3j;                                    //声明包
2   ……//该处省略了部分类的导入代码，读者可自行查阅随书光盘源代码
3   import android.widget.Button;                           //导入相关类
4   public class Sample3_10_Activity extends Activity{      //创建继承 Activity 的类
5      final int PROGRESS_DIALOG=0;                         //常量的声明
```

```
6       final int INCREASE=0;                                    //常量的声明
7       ProgressDialog pd;                                       //声明进度条引用
8        Handler hd;                                             //声明消息处理器
9       @Override
10      public void onCreate(Bundle savedInstanceState) { //继承 Activity 需重写的方法
11         super.onCreate(savedInstanceState);                   //调用父类
12         setContentView(R.layout.main);                        //切换到主界面
13         Button bok=(Button)this.findViewById(R.id.Button01);//得到确定按钮的使用
14         bok.setOnClickListener(                               //为确定按钮添加监听
15           new OnClickListener(){                              //匿名内部类
16                   public void onClick(View v) {               //添加监听需重写的方法
17                       showDialog(PROGRESS_DIALOG);            //弹出对话框
18           }});
19         hd=new Handler(){                                     //handler
20           @Override
21           public void handleMessage(Message msg){             //重写的方法
22                super.handleMessage(msg);                      //调用父类
23                switch(msg.what){                              //判断得到的消息
24                  case INCREASE:
25                     pd.incrementProgressBy(1);                //进度每次加 1
26                          if(pd.getProgress()>=100){
27                              pd.dismiss();                     //关闭进度条
28                          }
29                  break;
30      }}};}
31      @Override
32      public Dialog onCreateDialog(int id){                    //创建 Dialog
33       switch(id){
34         case  PROGRESS_DIALOG:
35             pd=new ProgressDialog(this);                      //创建进度条
36             pd.setMax(100);                                   //设置最大值
37           pd.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);//设置样式
38           pd.setTitle(R.string.title);                        //设置标题
39           pd.setCancelable(false);                            //对话框不能用返回按钮关闭
40           break;
41       }
42       return pd;                                              //返回 pd
43      }
44      @Override
45      public void onPrepareDialog(int id, Dialog dialog){//每次弹出对话框时被回调
46       super.onPrepareDialog(id, dialog);                      //调用父类
47           switch(id){                                         //判断 id
48         case  PROGRESS_DIALOG:
49           pd.incrementProgressBy(-pd.getProgress());          //对话框进度清零
50           new Thread(){                                       //开启新线程
51                public void run(){                             //重写 run() 方法
52                  while(true){                                //一直循环
53                     hd.sendEmptyMessage(INCREASE);           //发送消息
54                     if(pd.getProgress()>=100){               //判断进度是否大于 100%
55                         break;
56                     }
57                     try{
58                         Thread.sleep(40);                    //睡眠 40ms
59                     }catch(Exception e){                     //捕获异常
60                         e.printStackTrace();                 //打印堆栈信息
61       }}}}.start();                                         //开启线程
62         break;
```

```
63    }}}
```

其中：

- 第 14～18 行表示对 Button 按钮的监听，单击 Button 按钮则弹出对话框。
- 第 19～30 行表示接收消息，并判断消息是否为 INCREASE，如果是 INCREASE 则显示安装进度，并且判断安装进度是否已经达到 100%。
- 第 32～43 行表示创建 ProgressDialog 对话框，并对该对话框的一些属性进行设置。
- 第 45～63 行表示每次弹出对话框时被回调以动态更新对话框内容的方法，同时在进度条未达到最大之前每隔 40ms 发送消息，保证进度条的时时更新。

实例 11　用单选框实现选择个人特长

本节介绍单选列表对话框，主要是 AlertDialog 对话框的应用。

【实例描述】

运行本程序进入主界面，在主界面单击“确定”按钮，弹出单选列表对话框，在单选列表中可以选择自己的爱好，在主界面中的文本框中可以显示自己的选择，单击“确定”按钮可以退出该对话框。本实例的运行效果图，如图 3-11 所示。

图 3-11　程序运行效果图

提示：在图 3-11 中依次表示的是运行本程序进入主界面，在主界面单击“确定”按钮弹出单选列表，选中单选列表中的某一项，单击“确定”按钮跳至主界面并在主界面的文本框中显示自己的选择。

【实现过程】

本程序界面继承自 Activity，在主界面有 EditText 文本框与 Button 按钮等控件。为按钮添加监听器，单击 Button 按钮并创建 AlertDialog，设置其为单选列表框，并在 array.xml 中取出数据添加到单选列表对话框中。单击列表在 EditText 中显示自己的选择，并为该对话框的“确定”按钮添加监听。

【代码解析】

下面介绍实现本程序主界面搭建 main.xml 文件的设置，其代码如下。

代码位置：见本书随书光盘中源代码/第 3 章/Sample3_11/res/laoyout 目录下的 main.xml。

```
<?xml version="1.0" encoding="utf-8"?>                    <!--版本号和编码方式-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   >                                                      <!--LinearLayout-->
   <EditText
      android:text=""
      android:id="@+id/EditText01"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content">
   </EditText>                                            <!--EditText 控件-->
   <Button
      android:text="确定"
      android:id="@+id/Button01"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content">
   </Button>                                              <!--Button 控件-->
</LinearLayout>
```

提示：上述 main.xml 表示的是本程序的主界面的搭建，首先是设置 LinearLayout 的排列方式为竖直排列，然后依次添加 EditText 与 Button 并设置各个控件的属性。

上面已经介绍了本程序的主界面的搭建，接下来将要介绍如何实现单击按钮弹出单选列表对话框，单击其中的单选按钮，在主界面的 EditText 中显示自己选择的功能，代码如下。

代码位置：见本书随书光盘中源代码/第 3 章/Sample3_11/src/com/bn/ex3k 目录下的 Sample3_11_Activity。

```
package com.bn.ex3k;                                      //声明包
……//该处省略了部分类的导入代码，读者可自行查阅随书光盘源代码
import android.widget.EditText;                           //导入相关类
public class Sample3_11_Activity extends Activity{        //创建继承 Activity 的类
    final int List_DIALOG_SIMPLE=0;                       //设置常量
   @Override
   public void onCreate(Bundle savedInstanceState){//继承 Activity 需重写的方法
      super.onCreate(savedInstanceState);                 //调用父类
      setContentView(R.layout.main);                      //切换到主界面
      Button b =(Button) this.findViewById(R.id.Button01);//得到确定按钮的引用
      b.setOnClickListener(                               //为确定按钮添加监听
            new OnClickListener(){                        //匿名内部类
                  public void onClick(View v){            //添加监听需重写的方法
                     showDialog(List_DIALOG_SIMPLE);      //显示对话框
    }});}
   @Override
   public Dialog onCreateDialog(int id){                  //创建对话框
    Dialog dialog=null;
    switch(id)    {                                       //判断对话框 id
    case List_DIALOG_SIMPLE:                              // List_DIALOG_SIMPLE
        Builder b=new AlertDialog.Builder(this);          //创建对话框
        b.setIcon(R.drawable.hobby);                      //设置图标
            b.setTitle("单选列表对话框");                  //设置标题
```

```
24                b.setSingleChoiceItems(                    //为对话框添加适配器
25                 R.array.msa,                              //列表中的数据
26                 0,
27                 new DialogInterface.OnClickListener(){//添加监听事件
28                    public void onClick(DialogInterface dialog, int which) {
29                        //可以开发对话框关闭时的业务代码
30                        EditText et=(EditText) findViewById(R.id.EditText01);
                                                            //得到文本框的引用
31                        et.setText("您选择了: "+
32                        getResources().getStringArray(R.array.msa)[which]
                                                            //取出单击了列表中对应的数据
33                 );
34              } });
35              b.setPositiveButton(                          //为对话框添加按钮
36                     "确定",                                //按钮名称
37                     new DialogInterface.OnClickListener(){ //为按钮添加监听
38                        public void onClick(DialogInterface dialog, int which)
{}
39                 }
40              );
41              dialog=b.create();                            //创建 Dialog
42            break;
43        }
44            return dialog;                                  //返回 dialog
45    }}
```

其中：

- 第 11～15 行表示 Button 按钮的监听，单击 Button 按钮则弹出对话框。
- 第 21～23 行表示创建对话框，并且为对话框设置图标和标题。
- 第 24～34 行表示为对话框添加适配器，同时为该对话框添加数据，并对列表进行监听。
- 第 35～40 行表示为对话框添加按钮，设置名称同时添加监听。

实例 12　用复选框实现选择喜欢的城市

本节介绍的是复选列表对话框，主要是 AlertDialog 对话框的应用。

【实例描述】

运行本程序进入主界面，在主界面单击“确定”按钮，弹出复选列表对话框，在复选列表中可以选择相应的城市，在主界面的文本框中可以显示自己的选择，单击“确定”按钮可以退出该对话框。本实例的运行效果图，如图 3-12 所示。

说明：在图 3-12 中依次表示的是运行本程序进入主界面，在主界面单击“确定”按钮弹出复选列表，选中复选列表中的某几项，单击“确定”按钮跳至主界面并在主界面的文本框中显示自己的选择。

【实现过程】

本程序界面有 EditText 文本框与 Button 按钮，为 Button 按钮添加监听器，单击按钮创建对话框，同时设置其为复选列表框并在 array.xml 中取出数据添加到单选列表对话框中，单击该按钮在 EditText 中显示自己的选择，并为该复选对话框的“确定”按钮添加监听。

图 3-12　程序运行效果图

【代码解析】

下面介绍实现本程序主界面搭建 main.xml 文件的设置，其代码如下。

代码位置：见本书随书光盘中源代码/第 3 章/Sample3_12/res/laoyout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>                    <!--版本号和编码方式-->
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="vertical"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent"
6      >                                                      <!--LinearLayout-->
7      <EditText
8         android:text=""
9         android:id="@+id/EditText01"
10        android:layout_width="fill_parent"
11        android:layout_height="wrap_content">
12     </EditText>                                            <!--EditText 控件-->
13     <Button
14        android:text="确定"
15        android:id="@+id/Button01"
16        android:layout_width="fill_parent"
17        android:layout_height="wrap_content">
18     </Button>                                              <!--Button 控件-->
19  </LinearLayout>
```

提示：上述 main.xml 表示的是本程序的主界面的搭建，首先是设置 LinearLayout 的排列方式为竖直，然后依次添加 EditText 与 Button，并设置各个控件的属性。

上面已经介绍了本程序的主界面的搭建，接下来将要介绍如何实现单击按钮弹出复选列表对话框，单击其中的复选按钮，在主界面的 EditText 中显示自己选择的功能，代码如下。

代码位置：见本书随书光盘中源代码/第 3 章/Sample3_12/src/com/bn/ex31 目录下的 Sample3_12_Activity。

```
1   package com.bn.ex31;                                      //声明包
2   ……//该处省略了部分类的导入代码，读者可自行查阅随书光盘源代码
3   import android.widget.EditText;                           //导入相关类
```

```
4   public class Sample3_12_Activity extends Activity{  //创建继承 Activity 的类
5       final int List_DIALOG_MULTIPLE=0;               //声明常量
6       String[] items=null;                            //选项数组
7       boolean[] mulFlags=new boolean[]{false,false,false}; //初始复选情况
8       @Override
9       public void onCreate(Bundle savedInstanceState){//继承 Activity 需重写的方法
10          super.onCreate(savedInstanceState);         //调用父类
11          setContentView(R.layout.main);              //切换到主界面
12          items=getResources().getStringArray(R.array.msa);
                                                        //得到复选框中对应项的数据
13          Button b =(Button) this.findViewById(R.id.Button01);//拿到确定按钮的引用
14          b.setOnClickListener(                       //为确定按钮添加监听
15                  new OnClickListener(){              //匿名内部类
16                          public void onClick(View v){ //添加监听需重写的方法
17                              showDialog(List_DIALOG_MULTIPLE);
18      }});}
19      @Override
20      public Dialog onCreateDialog(int id){           //创建对话框的方法
21       Dialog dialog=null;
22       switch(id){
23       case List_DIALOG_MULTIPLE:                     //生成复选列表对话框的代码
24           Builder b=new AlertDialog.Builder(this);   //创建 Builder 对象
25               b.setIcon(R.drawable.icon);            //设置图标
26               b.setTitle("复选列表对话框");           //设置标题
27               b.setMultiChoiceItems(                 //为复选对话框添加适配器
28                   R.array.msa,                       //得到各项的数据
29                   mulFlags,                          //选中标准
30                   new DialogInterface.OnMultiChoiceClickListener(){ //监听器
31                       public  void  onClick(DialogInterface  dialog,  int
which,boolean isChecked) {
32                           mulFlags[which]=isChecked;     //第几个被选中
33                           String resultMsg="您选择了： "; //字符串
34                           for(int i=0;i<mulFlags.length;i++){    //for 循环
35                               if(mulFlags[i]){           //判断是否选中
36                                   resultMsg=resultMsg+items[i]+"、";
                                                           //生成字符串
37                               }
38                           }
39                   EditText et=(EditText) findViewById(R.id.EditText01);
                                                           //拿到文本框的引用
40                   et.setText(resultMsg.substring(0,  resultMsg.length()-1));
                                                           //文本框中添加数据
41           }});
42           b.setPositiveButton(                       //为该对话框设置按钮
43                   "确定",                            //按钮名称
44                   new DialogInterface.OnClickListener(){ //对该按钮添加监听
45                       public void onClick(DialogInterface dialog, int which) {}
46               }
47           );
48           dialog=b.create();                         //创建对话框
49           break;
50       }
51           return dialog;                             //返回 dialog
52   }}
```

其中：

- 第 14～18 行表示对 Button 按钮的监听，单击“Button”按钮则弹出对话框。

- 第 24～41 行表示创建对话框，并且为对话框设置图标和标题。为该复选对话框添加适配器，同时为该对话框添加数据，并对列表添加监听。
- 第 42～47 行表示为对话框添加按钮，设置名称，同时为该按钮添加监听。

实例 13　单击“确定”按钮弹出对话框

在本节中介绍的是单击“确定”按钮弹出对话框，主要是对话框的应用。

【实例描述】

运行本程序进入主界面，单击“确定”按钮可以弹出对话框。本实例的运行效果图，如图 3-13 所示。

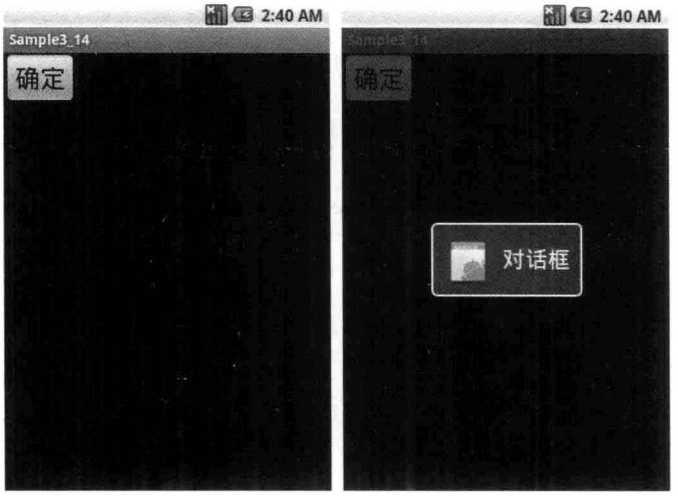


图 3-13　程序的运行效果图

提示：在图 3-13 中依次表示的是运行本程序进入主界面，在主界面单击“确定”按钮弹出对话框。

【实现过程】

本程序界面继承自 Activity，在主界面添加一个“Button”按钮，并且为该“确定”按钮添加监听器，当单击“确定”按钮后弹出对话框。

【代码解析】

下面介绍实现本程序主界面搭建 main.xml 文件的设置，其代码如下。

代码位置：见本书随书光盘中源代码/第 3 章/Sample3_13/res/laoyout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>                <!--版本号和编码方式-->
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="vertical"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent"
6      >                                                  <!--LinearLayout-->
7   <Button
```

```
8        android:id="@+id/Button01"
9        android:layout_width="wrap_content"
10       android:layout_height="wrap_content"
11       android:text="确定">                              <!--显示名称-->
12   </Button>
13   </LinearLayout>                                       <!--LinearLayout-->
```

提示：上述 main.xml 的代码主要是搭建程序开始运行的主界面，首先是设置 LinearLayout 的摆放方式为水平摆放，然后详细设置 Button 按钮。

上面已经介绍了本程序的主界面的搭建，接下来将要介绍如何实现单击按钮弹出对话框，代码如下。

代码位置：见本书随书光盘中源代码/第 3 章/Sample3_13/src/com/bn/ex3m 目录下的 Sample3_13_Activity。

```
1    package com.bn.ex3m;                                  //声明包
2    ……//该处省略了部分类的导入代码，读者可自行查阅随书光盘源代码
3    import android.widget.Button;                         //导入相关类
4    public class Sample3_13_Activity extends Activity{    //创建继承 Activity 的类
5        final int List_DIALOG_MULTIPLE=0;                 //声明常量
6       @Override
7       public void onCreate(Bundle savedInstanceState){//继承 Activity 需重写的方法
8          super.onCreate(savedInstanceState);            //调用父类
9          setContentView(R.layout.main);                 //切换到主界面
10         Button bb=(Button)this.findViewById(R.id.Button01);//取得 Button 的引用
11         bb.setOnClickListener(                         //为“确定”按钮添加监听器
12               new OnClickListener(){                   //匿名内部类
13                     public void onClick(View v){       //重写的方法
14                         showDialog(List_DIALOG_MULTIPLE);  //打开对话框
15       }});}
16      @Override
17      public Dialog onCreateDialog(int id){             //创建对话框的方法
18       Dialog dialog=null;
19       switch(id){
20           case List_DIALOG_MULTIPLE:                   //生成对话框的代码
21               Builder b=new AlertDialog.Builder(this);
22                   b.setIcon(R.drawable.icon);          //设置图标
23                   b.setTitle("对话框");                //设置标题
24                   dialog=b.create();                   //创建对话框
25                   break;
26       }
27           return dialog;                               //返回对话框
28   }}
```

其中：

- 第 11～15 行表示对“确定”按钮添加监听，当单击“确定”按钮时弹出对话框。
- 第 17～28 行表示创建对话框，并设置对话框，包括设置图片与设置其标题。

实例 14　查看时间日期的应用

Android 手机关于时间、日期的控件有 TimePicker、DatePicker，本节将详细介绍如何通过此类控件查看在一般模式下一天的时间，以及某种模式下的日期。

【实例描述】

在该软件中，首先 TextView 显示的是当前手机的时间，然后是 DatePicker 和 TimePicker 控件。DatePicker 控件中显示的是手机的日期，可以通过 DatePicker 控件中的上下键调整手机用户需要的日期，TimePicker 控件中显示的是手机的时间，同样通过控件中的上下键调整用户需要的时间。本实例的运行效果图，如图 3-14 所示。

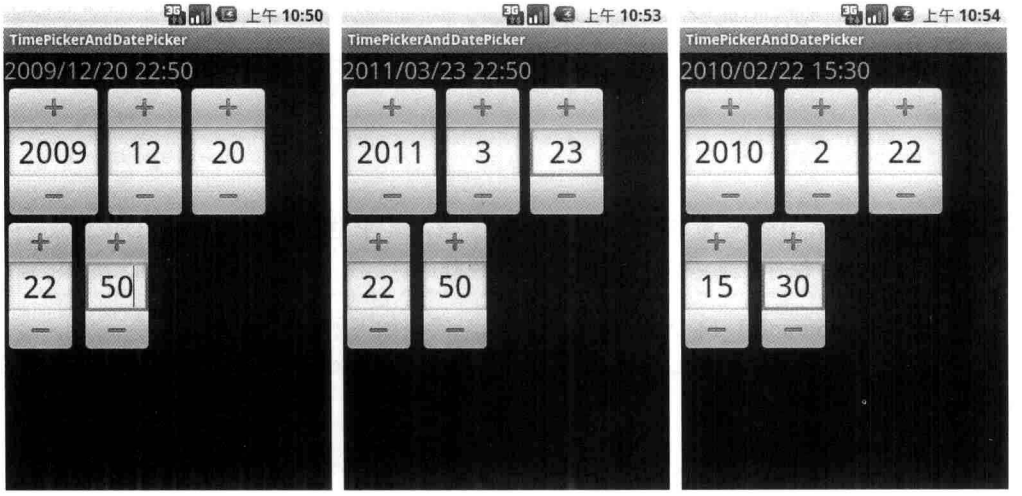


图 3-14　程序运行效果图

提示：在图 3-14 中，为手机日期、时间控件的效果图，手机用户通过单击控件上的增减控制键调整所需要的日期和时间，同时 TextView 同步显示用户调整的时间和日期。

【实现过程】

在本实例中首先创建 Calendar 对象，通过此对象获取手机当前的日期和时间，并赋值给整型变量，然后调用 showTime()方法，将获取的手机日期和时间赋值给 TextView，并显示出来。

在 DatePicke 控件的监听方法中，首先把获取的日期赋值给此控件，用户单击改变日期按钮来调用 onDateChanged()方法改变手机的日期，Timepicker 监听方法中将时间设置为 24 小时制，用户单击改变时间按钮来调用 onTimeChanged 方法改变手机的时间。

【代码解析】

首先要介绍 main.xml 布局文件，该文件代码如下所列。

代码位置：见本书随书光盘中源代码/第 3 章/Sample3_14/res/laoyout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>          <!--版本号和编码方式-->
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="vertical"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent"
6      >                                            <!--LinearLayout-->
7        <TextView
8        android:id="@+id/TextView01"
9        android:layout_width="fill_parent"
10       android:layout_height="wrap_content"
```

```
    android:textSize="24dip"
    >                                          <!--TextView 控件-->
    </TextView>
    <DatePicker
    android:id="@+id/DatePicker01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    >
   </DatePicker>
   <TimePicker
    android:id="@+id/TimePicker01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    >
   </TimePicker>
</LinearLayout>                                <!--LinearLayout-->
```

提示：上面代码为本软件的总布局代码，其中包含 TextView、DatePicker 和 TimePicker 等控件，主要显示当前的日期和时间。

下面介绍的是 Sample3_14_Activity 类的实现，代码如下所列。

代码位置：见本书随书光盘中源代码/第 3 章/Sample3_14/src/com/bn/ex3n 目录下的 Sample3_14_Activity。

```
package com.bn.ex3n;                                    //包名的声明
import android.app.Activity;                            //部分类的引入
……//该处省略了部分类的导入代码，请读者自行查看随书光盘的源代码
import android.os.Bundle;                               //部分类的引入
public class Sample3_14_Activity extends Activity {
    private int mYear;                                  //年份变量
    private int mMonth;                                 //月份变量
    private int mDay;
    private int mHour;                                  //小时变量
    private int mMinute;                                //分钟变量
public void onCreate(Bundle savedInstanceState) {
    Calendar c=Calendar.getInstance();                  //获取 Calender 对象
    mYear=c.get(Calendar.YEAR);                         //得到年份
    mMonth=c.get(Calendar.MONTH);                       //得到月份
    mDay=c.get(Calendar.DAY_OF_MONTH);                  //得到天数
    mHour=c.get(Calendar.HOUR_OF_DAY);                  //得到小时
    Minute=c.get(Calendar.MINUTE);                      //得到分钟
    super.onCreate(savedInstanceState);
    etContentView(R.layout.main);                       //设置布局
    showTime();                                         //调用设置 TextView 的方法
DatePicker dp=(DatePicker)findViewById(R.id.DatePicker01);//DatePicker 的创建
dp.init (
 mYear,                                                 //年份
 mMonth,                                                //月份
 mDay,                                                  //日
 new DatePicker.OnDateChangedListener(){
 public void onDateChanged(DatePicker view,             //日期变化的监听方法
 int year, int monthOfYear{
 mYear=year;                                            //改变的年份
 mMonth=monthOfYear;                                    //改变的月份
 mDay=dayOfMonth;
 showTime();                                            //设置文本框中的内容
   }    }    );
```

```
34  TimePicker tp=(TimePicker)findViewById(R.id.TimePicker01);
                                                          //得到 TimePicker 的引用
35    tp.setIs24HourView(true);
36    tp.setOnTimeChangedListener (                       //为该控件添加监听器
37    new TimePicker.OnTimeChangedListener() {
38    public void onTimeChanged(TimePicker view,
39   int hourOfDay, int minute) {                         //当其中的数据改变时
40   mHour=hourOfDay;                                     //从中得到小时
41   mMinute=minute;                                      //从中得到分钟数
42   showTime();                                          //调用函数设置文本的内容
43     }   }    );    }
44  private void showTime(){                              //设置文本框
45  TextView tv=(TextView)findViewById(R.id.TextView01);//拿到文本框中的引用
46  tv.setText(
47    new StringBuilder().append(mYear).append("/")       //将内容添加到文本框中
48    .append(formatTime(mMonth+1)).append("/")           //添加月数
49    .append(formatTime(mDay)).append(" ")               //添加天数
50    .append(formatTime(mHour)).append(":")              //添加小时数
51     .append(formatTime(mMinute))                       //添加分钟数
52    );       }
53  private String formatTime(int i){                     //转换字符串的方法
54    String s=""+i;                                      //转换类型
55    if(s.length()==1){
56      s="0"+s;                                          //字符串等于 1 的情况
57     }
58  return s;                                             //返回结果
59      }      }
```

其中：

- 第 6～10 行是成员变量年、月、日、小时、分钟的声明。
- 第 12～17 行是创建 Calendar 对象，并获取手机日期和手机时间。
- 第 21～33 行是 DatePicker 的创建和此控件的监听方法。
- 第 34～43 行是 TimePicker 的创建和此控件的监听方法。
- 第 44～59 行是 TextView 的赋值和日期、时间类型的转换方法。

实例 15　时钟模拟设计的应用

Android UI 设计模拟时钟 AnalogClock 和数字时钟 DigitalClock，此控件的功能实现非常简单，主要是为了获取系统时间动态更新 TextView 控件，显示系统时间。

【实例描述】

在本软件中，首先显示的是 AnalogClock 模拟时钟，模拟时钟随系统时间改变而更新，然后在 AnalogClock 下边显示的是 DigitalClock 数字时钟，同样随系统时间改变而更新，在 DigitalClock 下面为 TextView 控件，此控件动态显示系统时间。本实例的运行效果图，如图 3-15 所示。

提示：在图 3-15 中，为程序运行过程中 AnalogClock 模拟时钟、DigitalClock 数字时钟和 TextView 动态显示系统时间的效果。

图 3-15　程序的运行效果图

【实现过程】

在本实例中主要用到 AnalogClock 模拟时钟、DigitalClock 数字时钟 和 TextView 控件。其中模拟时钟和数字控件的功能实现非常简单，在本软件中直接加入控件就可以动态显示系统时间，而 TextView 动态显示系统时间比较复杂。

实现 TextView 的时候首先创建一个线程类，通过此线程的 Handler 发送消息，主控类接收消息重新获取时间，然后再赋值给 TextView，实现此控件的动态显示。

【代码解析】

首先要介绍的是 main.xml 布局文件，该文件代码如下所列。

代码位置：见本书随书光盘中源代码/第 3 章/Sample3_15/res/laoyout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>          <!--版本号和编码方式-->
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="vertical"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent"
6      >                                            <!--LinearLayout-->
7     <AnalogClock
8     android:id="@+id/AnalogClock01"
9     android:layout_width="wrap_content"
10    android:layout_height="wrap_content"
11    android:layout_marginLeft="90dip"
12    android:layout_marginTop="30dip"
13    >                                            <!--AnalogClock-->
14    </AnalogClock>
15    <DigitalClock
16    android:text="@+id/DigitalClock01"
17    android:id="@+id/DigitalClock01"
18    android:layout_width="wrap_content"
19    android:layout_height="wrap_content"
20    android:layout_marginTop="20dip"
21    android:layout_marginLeft="120dip"
22    >                                            <!--DigitalClock-->
23    </DigitalClock>
24    <TextView
```

```
    android:id="@+id/TextView01"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textSize="30dip"
    android:textColor="#ff0000"
    android:gravity="center"
    android:layout_marginTop="20dip"
    >                                                    <!--TextView-->
  </TextView>                                            <!--TextView 控件-->
  </LinearLayout>
```

提示：以上代码为本软件的总布局代码，此 xml 文件中包含 AnalogClock 模拟时钟控件、DigitalClock 数字时钟控件和 TextView 控件。

下面介绍的是 Sample3_15_Activity 类的实现，代码如下所列。

代码位置：见本书随书光盘中源代码/第 3 章/Sample3_15/src/com/bn/ex3o 目录下的 Sample3_15_Activity。

```
package com.bn.ex3o;                                     //包的声明
import android.app.Activity;                             //相关类的引入
……//此处省略了部分类的引入代码，请读者自行查看随书光盘源代码
import android.os.Bundle;                                //相关类的引入
public class Sample3_15_Activity extends Activity {
    public int mHour;                                    //小时
    public int mMinute;                                  //分钟
    public int mSecond;                                  //秒
    Handler hd=new Handler() {                           //Handler 接收消息
public void handleMessage(Message msg){
    witch(msg.what){
    case 0:
        ong time=System.currentTimeMillis();             //得到系统时间
        al Calendar c=Calendar.getInstance();            //得到 Calendar 引用
        c.setTimeInMillis(time);                    //把系统时间设置到 Calendar 中
        mHour=c.get(Calendar.HOUR);                      //得到系统的小时数
        mMinute=c.get(Calendar.MINUTE);                  /得到当前时间中的分数
        mSecond=c.get(Calendar.SECOND);
        TextView tv=(TextView)findViewById(R.id.TextView01);
                                                         //TextView 内容设置
        tv.setText("现在时间是: "+String.valueOf(mHour)+
        ":"+String.valueOf(mMinute)+":"+String.valueOf(mSecond));
}   }   };
public void onCreate(Bundle savedInstanceState) {        //主控制类的创建
super.onCreate(savedInstanceState);
setContentView(R.layout.main);                           //设置布局
MyThread thread=new MyThread(this);                      //创建线程类
thread.start();                                          //开始线程
 }      }
```

提示：以上代码是 Sample3_15_Activity 主控制类，本类首先设置其布局，然后创建线程类，调用此线程类 Handler 发送消息，动态更改 TextView 显示的系统时间。

接下来介绍的是发送消息的线程类，代码如下。

代码位置：见随书光盘中源代码/第 3 章/Sample3_15/src/com.bn.ex3o/ MyThread.java。

```
package com.bn.ex3o;                                     //声明包
public class MyThread extends Thread{                    //创建继承 Thread 的类
```

```
3       boolean flag=true;                                  //flag 为 true
4       Sample3_15_Activity activity;
5       public MyThread(Sample3_15_Activity activity){      //构造器
6       this.activity=activity;                             //为当前的 activity 赋值
7   }
8   public void run(){                                      //重写的 run() 方法
9   while(flag){                                            //判断 flag 是否为 true
10      try{
11          MyThread.sleep(1000);                           //休息 1000ms
12          }
13      catch(Exception e){                                 //捕获异常
14      e.printStackTrace();                                //打印堆栈信息
15      }
16      activity.hd.sendEmptyMessage(0);                    //发送消息 0
17  }    }    }
```

提示：此方法为发送消息的线程类，首先获取 Sample3_15_Activity 的对象，然后重写 Thread 的重写方法，此方法不间断地让此线程休眠 1000ms，最后发送消息。

实例 16　动态列表配置选项

Android 手机上的 ListActivity 应用非常方便，为此类增加内容适配器，可以把用户需要的内容以列表的形式罗列出来，下面开始介绍 ListActivity 的应用。

【实例描述】

本软件以列表的形式显示水果的图片和水果的名称，通过单击水果的名称可以弹出 Toast 提示。程序的运行效果如图 3-16 所示。

图 3-16　动态列表配置

提示：在图 3-16 中依次表示的是运行本程序进入主界面，在主界面中单击某一选项，弹出 Toast，显示用户自己选择的信息。

【实现过程】

在本实例中 Activity 类需要继承 ListActivity 类，同时为此类增加 BaseAdapter 适配器，在 BaseAdapter 适配器中设置本软件的布局格式，LinearLayout 为横行布局，设置图片控件 ImageView 的大小，并把此图片控件增加到 LinearLayout 中，然后增加 TextView 控件，显示水果的名称。

通过单击水果的名称调用其监听方法 onListItemClick，此方法获取列表内容并用 Toast 显示出来。

【代码解析】

下面开始介绍 sample3_16Activity 类的实现，代码如下。

代码位置：见本书随书光盘中源代码/第 3 章/Sample3_16/src/com/bn/ex3p 目录下的 Sample3_16_Activity。

```
1   package com.bn.ex2p;                                          //包名的声明
2   import android.app.ListActivity;                              //相关类的引入
3   ……//此处省略了部分类的引入代码，请读者自行查看随书光盘源代码
4   import android.widget.ListView;                               //相关类的引入
5   public class sample3_16Activity extends ListActivity {
6    List<String> items ;                                         //文本内容
7   public void onCreate(Bundle icicle) {
8       uper.onCreate(icicle);
9       tems = fillArray();                                       //文本内容
10      inal int[] drawableIds={R.drawable.app,R.drawable.bpp,
11      .drawable.cpp,};                                          //图片
12      aseAdapter ba=new BaseAdapter() {                         //创建适配器
13      ublic int getCount() {
14      eturn 3;
15  }
16  public Object getItem(int arg0) { return null; }             //重写方法
17  public long getItemId(int arg0) { return 0; }
18  public View getView(int position, View convertView,
19  ViewGroup parent){
20      LinearLayout ll=new LinearLayout(sample3_16Activity.this);
                                                                  //LinearLayout 布局
21  ll.setOrientation(LinearLayout.HORIZONTAL);                   //设置朝向
22  ll.setPadding(5,5,5,5);                                       //设置四周留白
23  ImageView  ii=new ImageView(sample3_16Activity.this);     //初始化 ImageView
24  ii.setImageDrawable(getResources().getDrawable(drawableIds[position]));
                                                                  //设置图片
25  ii.setScaleType(ImageView.ScaleType.FIT_XY);
26  ii.setLayoutParams(new Gallery.LayoutParams(50,50));
27  ll.addView(ii);                                               //添加到 LinearLayout 中
28  TextView tv=new TextView(sample3_16Activity.this);
29  tv.setText(items.get(position));                              //设置内容
30  tv.setTextSize(24);                                           //设置字体大小
31  tv.setTextColor(sample3_16Activity.this.getResources().getColor(R.color.
white));//设置字体颜色
32  tv.setPadding(5,5,5,5);                                       //设置四周留白
33  tv.setGravity(Gravity.LEFT);
34  ll.addView(tv);                                               //添加到 LinearLayout 中
35      return ll;                                                //返回布局
36  }       };
37      this.setListAdapter(ba);                          //为 ListView 设置内容适配器
```

```
38  }
39      private List<String> fillArray()  {                    //调用名称内容方法
40      List<String> items = new ArrayList<String>();
41      items.add("苹果");                                      //集合中增加内容
42      tems.add("香蕉");
43      items.add("草莓");
44      return items;                                          //返回集合
45  }
46  protected void onListItemClick(ListView l, View v,         //监听事件方法
47   int position, long id)  {
48      Toast.makeText(this, items.get(position),
49      Toast.LENGTH_SHORT).show();                            //显示 Toast
50  }    }
```

其中：

- 第 9～11 行是本软件中加载水果名称和水果图片的代码。
- 第 12～38 行是 ListActivity 的内容设配器，此适配器中设置了显示内容的布局方法。
- 第 39～45 行是水果名称的集合方法，调用此方法获取内容。
- 第 46～50 行是列表的监听事件，单击其内容弹出 Toast 提示。

实例 17　在安卓中浏览网页

在 Android 手机中连接网络非常方便，下面将要介绍的是关于 Web 应用的开发，打开此软件进入新浪的主页，单击相关链接跳转到其他界面。

【实例描述】

进入本软件首先显示的是新浪的主页，可以单击其他网页的链接打开相关网页。程序的运行效果，如图 3-17 所示。

图 3-17　新浪主页

提示：在图 3-17 中表示的是运行本程序进入的主界面。

【实现过程】

本实例首先设置布局，在布局中加入 WebView 控件，用来显示新浪的网页界面。然后在 Activity 中获取 WebView 对象，然后调用 setWebViewClient()方法设置浏览器为 WebView 中浏览，最后调用 loadUrl 加载对应的网页。

【代码解析】

首先要介绍的是 main.xml 布局文件，该文件代码如下。

代码位置：见本书随书光盘中源代码/第 3 章/Sample3_17/res/laoyout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>          <!--版本号和编码方式-->
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="vertical"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent"
6      >                                            <!--LinearLayout-->
7      <WebView
8       android:id="@+id/webview"
9       android:layout_width="fill_parent"
10       android:layout_height="fill_parent"
11     />
12  </LinearLayout>                                 <!--LinearLayout-->
```

提示：以上代码为本软件的布局文件，WebView 为布局文件的重点，此控件显示网页。

接下来开始介绍 sample3_17Activity 类的实现，代码如下。

代码位置：见本书随书光盘中源代码/第 3 章/Sample3_17/src/com/bn/ex3q 目录下的 Sample3_17_Activity。

```
1   package com.bn.ex3q;                                    //包的声明
2   import android.app.Activity;                            //相关类的引入
3   ……//此处省略了部分类的引入代码，请读者自行查看随书光盘源代码
4   import android.os.Bundle;                               //相关类的引入
5   public class Sample3_17Activity extends Activity {      //创建继承 Activity 的类
6   public void onCreate(Bundle savedInstanceState) {       //重写的方法
7       super.onCreate(savedInstanceState);
8       etContentView(R.layout.main);
9       WebView webview = (WebView) findViewById(R.id.webview);
                                                            // 获取 WebView 对象
10      webview.getSettings().setJavaScriptEnabled(true);   // 使能 JavaScript
11      webview.setWebViewClient(new WebViewClientDemo());
12      webview.loadUrl("http://www.sina.com.cn/");         // 加载网页
13   }
14  private class WebViewClientDemo extends WebViewClient {
15      public boolean shouldOverrideUrlLoading(WebView view,
16      String url) {                       // 在 WebView 中而不是默认浏览器中显示页面
17      view.loadUrl(url);
18      return true;
19  }    }    }
```

提示：首先设置布局，加载 WebView 控件获取 WebView 控件引用，然后设置网页显示方式为 WebView 控件中显示，最后调用 loadUrl 加载新浪的主页。

实例 18　切换列表显示

在本节中主要介绍的是 TABActivity 的开发与应用。

【实例描述】

运行本程序后进入主界面，将弹出对话框，单击对话框的“确定”按钮可以跳转到介绍界面，在介绍界面的上侧有一行标签，单击其中的某一个标签，可以跳转到其所代表的说明界面。本实例的运行效果图，如图 3-18 所示。

图 3-18　程序的运行效果图

提示：在图 3-18 中依次表示的是运行本程序进入主界面，在主界面中单击某一标签，进入该标签所代表的界面，在标签界面显示该标签的具体信息。

【实现过程】

首先该类继承 TabActivity，实现 OnTabChangeListener 接口，并且在程序开始运行时实现弹出对话框。然后依次创建 TabSpec、设置其图片、标题与对应的 id，并设置 onTabChanged 监听。

【代码解析】

下面介绍实现本程序主界面搭建 main.xml 文件的设置，其代码如下。

代码位置：见本书随书光盘中源代码/第 3 章/Sample3_18/res/laoyout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>                    <!--版本号和编码方式-->
2   <FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
3       android:orientation="vertical" android:layout_width="fill_parent"
4       android:layout_height="fill_parent">
5       <LinearLayout
6       android:id="@+id/tabFood"
7       android:layout_width="fill_parent"
8       android:layout_height="fill_parent"
9       androidrientation="vertical"
10      >                                                     <!--LinearLayout-->
11          <TextView
12          android:id="@+id/TextView01"
13          android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        >                                                    <!--TextView 控件-->
        </TextView>
    </LinearLayout>
    <LinearLayout
    android:id="@+id/tabCloths"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    androidrientation="vertical"
    >                                                    <!--LinearLayout-->
     <TextView
         android:id="@+id/TextView02"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
     >                                                   <!--TextView 控件-->
     </TextView>
    </LinearLayout>
    <LinearLayout
    android:id="@+id/tabOutside"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    androidrientation="vertical"
    >
         <TextView
             android:id="@+id/TextView03"
             android:layout_width="wrap_content"
             android:layout_height="wrap_content"
         >                                               <!--TextView 控件-->
         </TextView>
     </LinearLayout>
</FrameLayout>                                           <!--LinearLayout-->
```

提示：上述 main.xml 代码主要是搭建程序开始运行的主界面，首先是设置 FrameLayout 的摆放方式为水平摆放，然后是对 LinearLayout 设置，最后在每个 LinearLayout 中对 TextView 详细设置。

上面已经介绍了本程序主界面的搭建，接下来将要介绍的是如何实现单击“确定”按钮并跳转到标签界面的功能，以及单击标签界面跳转界面的功能，代码如下。

代码位置：见本书随书光盘中源代码/第 3 章/Sample3_18/ src/com/bn/ex3r 目录下的 Sample3_18_Activity。

```
package com.bn.ex3r;                                     //声明包
……//该处省略了部分类的导入代码，读者可自行查阅随书光盘源代码
import android.widget.TabHost.TabSpec;                   //导入相关类
public class Sample3_18_Activity extends TabActivity
implements OnTabChangeListener{                  //创建继承 TabActivity 的类并实现接口
   @Override
   public void onCreate(Bundle savedInstanceState) {     //需重写的方法
      super.onCreate(savedInstanceState);                //调用父类
       new AlertDialog.Builder(this)                     //创建对话框
       .setTitle("冬季保健小常识")                        //设置对话框标题
       .setMessage("教你如何快乐过冬！")                  //设置对话框内容
       .setPositiveButton("确定", null)                  //为对话框添加按钮
       .show();                                          //显示对话框
       TabHost tabHost = this.getTabHost();              //创建 TabHost 对象
      LayoutInflater.from(this).inflate(R.layout.main,
            tabHost.getTabContentView(), true);
      TabSpec tabFood = tabHost.newTabSpec("food").setIndicator("饮食",
```

```
18              this.getResources().getDrawable(R.drawable.food)).setContent(
19              R.id.tabFood);                                    //设置饮食 TabSpec
20          tabHost.addTab(tabFood);                              //添加到标签中
21          TabSpec tabCloths = tabHost.newTabSpec("cloths").setIndicator("保暖",
22              this.getResources().getDrawable(R.drawable.cloths)).setContent(
23              R.id.tabCloths);                                  //设置保暖 TabSpec
24          tabHost.addTab(tabCloths);                            //添加到标签中
25          TabSpec tabOutside = tabHost.newTabSpec("outside").setIndicator("出行",
26              this.getResources().getDrawable(R.drawable.outside)).setContent(
27              R.id.tabOutside);                                 //设置出行 TabSpec
28          tabHost.addTab(tabOutside);                           //添加到标签
29          tabHost.setOnTabChangedListener(this);                //对该标签添加监听
30          onTabChanged("food");                                 //开始时默认的选择
31      }
32      @Override
33      public void onTabChanged(String tabId){                   //对标签添加监听
34       if(tabId.equals("food")){                                //tabld 与 food 相等
35           TextView tv=(TextView)findViewById(R.id.TextView01);
                                                                  //得到 TextView01 的引用
36            tv.setText(
37                    ......//该处省略了对饮食标签的说明，读者可自行查阅随书光盘源代码中的说明
38                        );
39       }
40       if(tabId.equals("cloths"))                               //tabld 与 cloths 相等
41       {
42           TextView tv2=(TextView)findViewById(R.id.TextView02);
                                                                  //得到 TextView02 的引用
43            tv2.setText(
44                    ......//该处省略了对保暖标签的说明，读者可自行查阅随书光盘源代码中的说明
45                    );
46       }
47       if(tabId.equals("outside"))                              //tabld 与 outside 相等
48       {
49           TextView tv3=(TextView)findViewById(R.id.TextView03);
                                                                  //得到 TextView03 的引用
50            tv3.setText(
51                    .......//该处省略了对出行标签的说明，读者可自行查阅随书光盘源代码中的说明
52                    );
53  }}}
```

其中：

- 第 9～13 行表示程序开始时创建对话框，并且设置该对话框的标题、内容与“确定”按钮。
- 第 17～28 行表示创建选项卡，并将其添加到 tabHost 中生成标签。
- 第 32～53 行表示对标签添加监听，同时判断单击的是哪个选项卡，并给对应的 TextView 添加详细的介绍信息。

小结

本章主要介绍了几种高级控件的开发及应用，其中 ListView 列表控件应用中用了内容适配器 BaseAdapter，在内容适配器中为 ListView 自定义设置布局，并为列表设置内容，同时可以为此控件设置选中或单击监听事件。

接着为读者介绍了 AlertDialog 对话框的使用，该对话框在软件开发中应用非常广泛，用户可以为对话框设置内容，并在特定的情况下弹出重要信息。

第 4 章　手机用户界面

第 3 章主要介绍了 Android 手机控件的开发和使用，合理地运用控件可以摆放出漂亮的界面，用户界面的要求也可以得到满足，在本章将对手机用户界面的开发进行详细介绍。

实例 1　获取手机屏幕的分辨率

了解一款手机往往是从其屏幕的分辨率开始的，这样就能够把握好尺寸，合理地利用手机的屏幕空间。在本节将介绍如何获取 Android 手机的屏幕分辨率。

【实例描述】

在本软件中存在一个获取屏幕分辨率的按钮和用于显示获取数据的 EditText，当单击主界面中的按钮时，系统自动获取分辨率，将分辨率填写到 EditText 中。本实例的运行效果图，如图 4-1 所示。

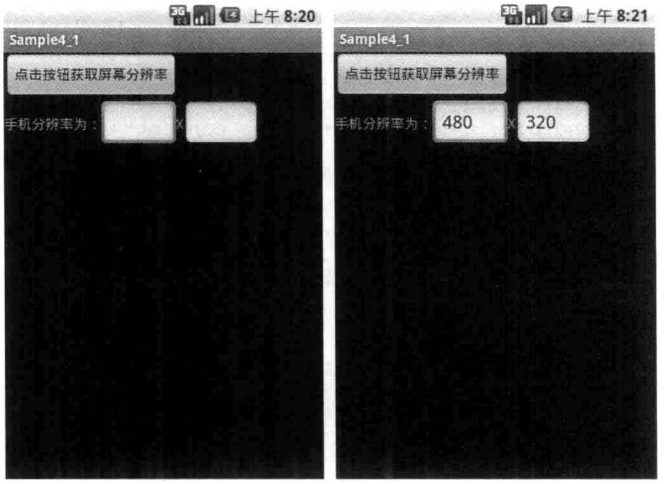


图 4-1　获取手机屏幕分辨率的效果图

提示：在图 4-1 中依次表示的是运行本程序进入主界面，在主界面单击“点击按钮获取屏幕分辨率”按钮，获取该手机的分辨率，并在文本框显示手机分辨率。

【实现过程】

在该实例中主要运用了 Button 按钮的 setOnClickListener 方法，为 Button 按钮添加监听器，使用 DisplayMetrics 创建的对象获取手机屏幕的分辨率。为 EditTExt 设置文本使用的是 EditText 的 setText 方法。

【代码解析】

本部分要介绍的是该实例的核心代码，首先介绍的是 main.xml 文件，该文件完成的是界面的搭建任务，该文件代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_1/ res/layout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="vertical"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent"
6      >                                              <!--LinearLayout-->
7       <Button
8           android:text="单击按钮获取屏幕分辨率"
9           android:id="@+id/Button01"
10          android:layout_width="wrap_content"
11          android:layout_height="wrap_content">
12      </Button>                                      <!--Button 控件-->
13      <LinearLayout
14          android:id="@+id/LinearLayout01"
15          android:layout_width="fill_parent"
16          android:layout_height="wrap_content"
17          android:orientation="horizontal">
18          <TextView
19              android:text="手机分辨率为："
20              android:id="@+id/TextView01"
21              android:layout_width="wrap_content"
22              android:layout_height="wrap_content">
23          </TextView>                                <!--TextView 控件-->
24          <EditText
25              android:id="@+id/EditText01"
26              android:layout_width="75dip"
27              android:layout_height="wrap_content">
28          </EditText>                                <!--EditText 控件-->
29          <TextView
30              android:text="X"
31              android:id="@+id/TextView02"
32              android:layout_width="wrap_content"
33              android:layout_height="wrap_content">
34          </TextView>                                <!--TextView 控件-->
35          <EditText
36              android:id="@+id/EditText02"
37              android:layout_width="75dip"
38              android:layout_height="wrap_content">
39          </EditText>                                <!--EditText 控件-->
40      </LinearLayout>
41  </LinearLayout>                                    <!--LinearLayout-->
```

其中：

- 第 2～12 行主要完成总体 LinearLayout 的设置和 Button 按钮的设置。
- 第 13～40 行为 id 是 LinearLayout01 的设置和其中控件的设置，其中的控件包括 TextView 和 EditText。

经过 main.xml 对界面的设置，下面要做的就是实现获取屏幕分辨率的功能，实现该功能的类为 Sample4_1_Activity，该类的代码如下。

代码位置：见随书光盘中源代码/第 4 章/Sample4_1/src/com/bn/chap4/screen 目录下的 Sample4_1_Activity.class。

```
1   package com.bn.chap4.screen;
```

```
2   ……//该处省略了部分类的导入代码，读者可自行查看随书光盘中源代码。
3   public class Sample4_1_Activity extends Activity {
4      public void onCreate(Bundle savedInstanceState) {
5         super.onCreate(savedInstanceState);
6         setContentView(R.layout.main);                          //设置主界面
7         final DisplayMetrics dm=new DisplayMetrics();           //创建对象
8         getWindowManager().getDefaultDisplay().getMetrics(dm);
9         final EditText et=(EditText)this.findViewById(R.id.EditText01);
                                                                  //EditText 对象
10        final EditText ett=(EditText)this.findViewById(R.id.EditText02);
                                                                  //EditText 对象
11        Button ok=(Button)this.findViewById(R.id.Button01);     //按钮对象
12        ok.setOnClickListener(                                  //设置监听器
13          new OnClickListener(){
14                  public void onClick(View v) {
15                      et.setText(dm.heightPixels+"");     //将获取的值填到 et 中
16                      ett.setText(dm.widthPixels+"");
17  }});}}
```

提示：在上述代码中首先完成的是界面的设置，然后创建 DisplayMetrics 对象，通过 DisplayMetrics 对象将获取的屏幕分辨率设置显示在界面上。

实例 2　实现按钮的界面响应

用户界面除了具有呈现内容给用户的责任，还需要相应用户的各种操作事件，在本节将对界面的相应事件进行详细讲解。

【实例描述】

本软件实现功能简单，在主界面中有一个 Button 按钮，当触摸该 Button 时，可以随意拖动该按钮，当在手机屏幕上随意单击一个位置时，该按钮会移动到触摸点位置。本实例的运行效果图，如图 4-2 所示。

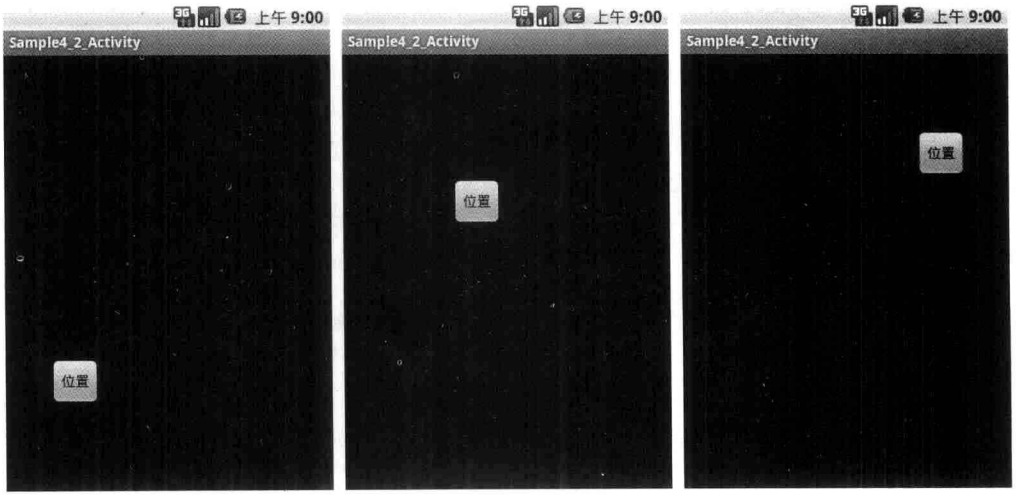


图 4-2　界面响应效果图

提示：在图 4-2 中，在程序运行中，按钮随触摸位置的改变而改变。

【实现过程】

在 Android 平台中，主要使用回调机制来处理界面用户事件。每个 View 都有自己处理事件的回调方法，若事件没有被 Activity 中的任何一个 View 所处理时，系统就会调用 Activity 的事件处理回调方法进行处理。常用的事件处理回调方法如下所列。

- public boolean onKeyDown(int KeyCode,KeyEvent event)
- public boolean onKeyUp(int KeyCode,KeyEvent event)
- public boolean onTouchEvent(MotionEvent event)
- public boolean onTrackballEvent(MotionEvent event)

【代码解析】

经过上面的理论介绍，接下来要介绍的是该软件的核心代码部分，首先要介绍的是 main.xml 文件的设计与实现，该文件完成的是界面的搭建，其代码如下。

代码位置：见随书光盘中源代码/第 4 章/Sample4_2/res/layout 目录下的 main.xml。

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
  android:id="@+id/AbsoluteLayout01"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  xmlns:android="http://schemas.android.com/apk/res/android">
                                                    <!--AbsoluteLayout-->
  <Button
    android:layout_y="123dip"
    android:layout_x="106dip"
    android:text="@string/location"
    android:layout_height="wrap_content"
    android:id="@+id/Button01"
    android:layout_width="wrap_content">
  </Button>                                         <!--Button 控件-->
</AbsoluteLayout>
```

提示：上述代码主要完成的是在 AbsolutelyLayout 布局中设置一个 Button 按钮。

下面要介绍的是该软件的主类 Sample4_2_Activity 的设计与实现，在该类中重写了 onKeyDown、onKeyUp 以及 onTouchEvent 方法，实现按钮的移动，该类的代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_2/src/com/bn/chap4/jmxy 目录下的 Sample4_2_Activity.class。

```
package com.bn.chap4.jmxy;
……//该处省略了部分类的导入代码，读者可自行查看随书光盘中源代码。
public class Sample4_2_Activity extends Activity {
    final static int WRAP_CONTENT=-2;           //表示 WRAP_CONTENT 的常量
    final static int X_MODIFY=4;                //在非全屏模式下 X 坐标的修正值
    final static int Y_MODIFY=52;               //在非全屏模式下 Y 坐标的修正值
    int xSpan;                                  //触控笔单击的 X 位置
    int ySpan;                                  //触控笔单击的 Y 位置
  public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);                    //设置当前界面
```

```
12          Button bok=(Button)this.findViewById(R.id.Button01);        //创建对象
13          bok.setOnTouchListener(                                     //添加监听器
14                  new OnTouchListener(){
15                          public boolean onTouch(View view, MotionEvent event) {
16                              switch(event.getAction()){
17                               case MotionEvent.ACTION_DOWN://触控笔按下
18                                   xSpan=(int)event.getX(); //获取 X 坐标
19                                   ySpan=(int)event.getY(); //获取 Y 坐标
20                               break;
21                               case MotionEvent.ACTION_MOVE: //触控笔移动
22                                   Button bok=(Button)findViewById(R.id.Button01);
23                                   ViewGroup.LayoutParams  lp=
                                                          //让按钮随着触控笔的移动而移动
24                                      new AbsoluteLayout.LayoutParams(
25                                              WRAP_CONTENT,     //创建对象
26                                              WRAP_CONTENT,
27                                              (int)event.getRawX()-xSpan-X_
MODIFY,
28                                              (int)event.getRawY()-ySpan-Y_
MODIFY
29                                      ) ;
30                                   bok.setLayoutParams(lp); //设置按钮位置
31                               break;
32                              }
33                              return true;
34                          }});}
35      public boolean onKeyDown (int keyCode, KeyEvent event) { //键盘键按下的方法
36       Button bok=(Button)this.findViewById(R.id.Button01);
37       bok.setText(keyCode+" Down");                                  //设置文本
38       return true;
39      }
40      public boolean onKeyUp (int keyCode, KeyEvent event) {//键盘键抬起的方法
41       Button bok=(Button)this.findViewById(R.id.Button01);
42       bok.setText(keyCode+" Up");                                    //设置文本
43       return true;
44      }
45      public boolean onTouchEvent (MotionEvent event){
                                                      //让按钮随着触控笔的移动一起移动
46       Button bok=(Button)this.findViewById(R.id.Button01);
47       ViewGroup.LayoutParams  lp=
48               new AbsoluteLayout.LayoutParams(                       //创建对象
49                       WRAP_CONTENT,
50                       WRAP_CONTENT,
51                       (int)event.getRawX()-xSpan-X_MODIFY,    //按钮坐标
52                       (int)event.getRawY()-ySpan-Y_MODIFY
53               ) ;
54           bok.setLayoutParams(lp);                                   //设置按钮位置
55       return true;
56      }}
```

其中：

- 第 4～12 行为声明该类的成员变量以及创建对象。
- 第 13～34 行主要完成的是为按钮添加监听器，若为单击屏幕，则记录触摸点的 XY 坐标，若为触摸后移动事件，则时时设置按钮位置。
- 第 35～39 行为重写 onKeyDown 方法，并设置 Button 的显示文本。
- 第 40～56 行为重写 onKeyUp 和 onTouchEvent 方法，在 onKeyUp 方法中设置 Button 的显示文本，在 onTouchEvent 方法中，设置按钮的新坐标，并将按钮设置在该位置。

实例 3　给控件做背景图的小技巧

一个好的用户界面（UI）不仅需要合理的布局，而且布局中的每个控件还应该都有漂亮的外观。想要使控件的外观贴切于自己的应用程序的环境、定位，仅仅依靠 Android 提供的默认外观是远远不够的。还需要美工人员开发出合适的背景图片，并与编程开发相配合。

【实例描述】

本软件仅仅实现了最基本的功能，在界面中显示的画面主要为提醒开发人员采用背景图片会带来另一个问题，如果仅仅是根据内容多少按比例拉伸背景可能效果会很差，例如，自定义一个圆角矩形图片作为背景，若作为图片按比例拉伸可能会发生变形。

幸运的是，Android 平台的提供者关注到了这个问题，并为其提供了较为完美的解决方案——按需要拉伸图片格式“.9.png”。本实例的运行效果图，如图 4-3-1 所示。

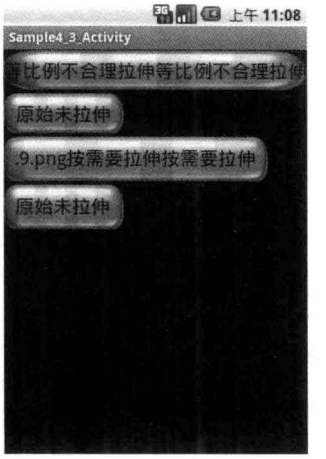


图 4-3-1　控件的外观控制效果

提示：图 4-3-1 是程序运行中使用普通图片和.9.png 图片在控件上的对比情况。

【实现过程】

“.9.png”图片的制作十分简单，只需要找到 E:\Android new\android-sdk-windows\tools（笔者使用的目录）目录下的 draw9patch.bat 文件即可制作自己的“.9.png”图片，其制作过程也十分简单，首先将需要处理的图片导入，然后在需要的位置绘制黑线即可，如图 4-3-2 所示。

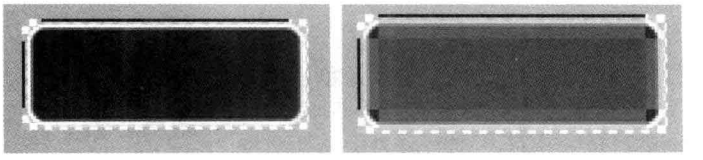


图 4-3-2　“.9.png”图片的制作

提示：图 4-3-2 中红色区域为未来内容所在空间，红色区域外面的部分不会拉伸。

【代码解析】

经过上面的介绍，下面要讲解的是该实例的核心代码部分，首先介绍的是 main.xml 文件的设计与实现，该文件代码如下。

代码位置：见随书光盘中源代码/第 4 章/Sample4_3/res/layout 目录下的 main.xml。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >                                                    <!--LinearLayout-->
   <Button
     android:text="@string/largea"
     android:id="@+id/Button01"
     android:textSize="20dip"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:background="@drawable/myselectora"
   >                                                     <!--Button 控件-->
   </Button>
   <Button
     android:text="@string/comm"
     android:id="@+id/Button02"
     android:textSize="20dip"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:background="@drawable/myselectora"
   >                                                     <!--Button 控件-->
   </Button>
   <Button
     android:text="@string/largeb"
     android:id="@+id/Button03"
     android:textSize="20dip"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:background="@drawable/myselectorb"
   >                                                     <!--Button 控件-->
   </Button>
   <Button
     android:text="@string/comm"
     android:id="@+id/Button04"
     android:textSize="20dip"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:background="@drawable/myselectorb"
   >                                                     <!--Button 控件-->
   </Button>
</LinearLayout>
```

提示：上述代码主要完成的是对不同 Button 按钮的设置，其主要区别在于所选背景图片不一样。

接下来要介绍的是 Sample4_3_Activity 类，该类代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_3/src/com/bn/chap4/kjwgkz 目录下的 Sample4_3_Activity.class。

```
package com.bn.chap4.kjwgkz;
……//该处省略了部分类的导入代码，读者可自行查看随书光盘中源代码。
```

```
3   public class Sample4_3_Activity extends Activity {
4      public void onCreate(Bundle savedInstanceState) {
5         super.onCreate(savedInstanceState);
6         setContentView(R.layout.main);                    //设置当前界面
7      }}
```

提示：上述代码主要完成的是重写 onCreste 方法，设置 main.xml 为当前界面。

实例 4　定时改变提示信息

有时候需要在不同的时间发送不同的消息，在 Android 手机中就可以通过自定义实现该功能，在本节将对其进行详细介绍。

【实例描述】

运行该软件时，后台自动启动线程，每 1 秒钟发送 1 条消息，在手机屏幕上会显示出需要提示的信息，当发到 10 条消息时，系统停止发送消息。本实例的运行效果图，如图 4-4 所示。

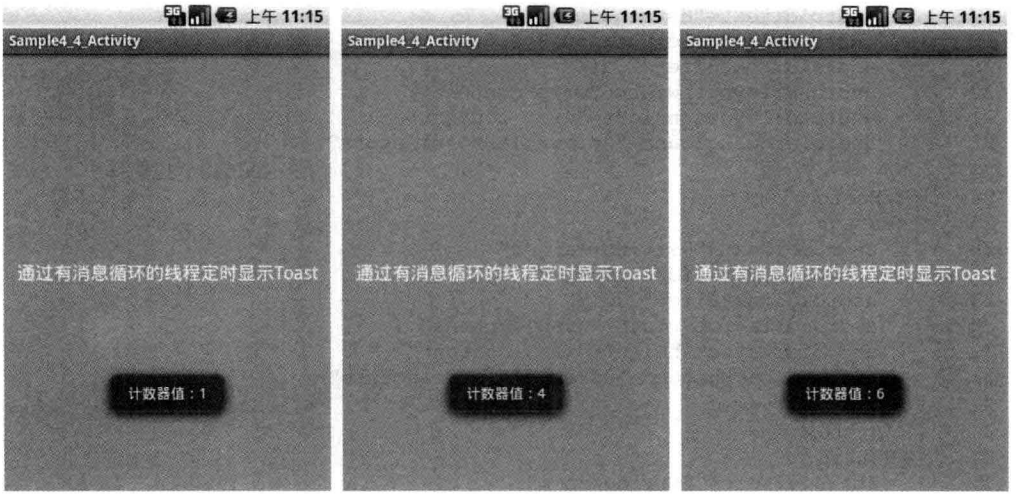


图 4-4　定时改变 Toast 提示信息

提示：图 4-4 是程序运行过程中，每 1 秒钟改变 1 条信息的内容，并使用 Toast 进行提示。

【实现过程】

该软件主要运用了 Handler 接收消息的功能，在这里需要说明的是 Handler 和 Thread 不一定是一一对应的，也就是说，在一个线程中可以有多个 Handler，每个消息都可以指定不同的 Handler，因为每个消息都可以有不同的行为。

在该软件的设计中还使用到 Bundle 绑定消息，向 Handler 发送不同的消息。如果一个线程中调用了 Looper.prepare()，那么系统就会自动地为该线程建立一个消息队列，然后调用 Looper.loop()之后就进入消息循环。

【代码解析】

在本部分要介绍的是该软件的核心代码部分，首先要介绍的是 main.xml 文件，该文件的代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_4/res/layout 目录下的 main.xml。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:background="@color/blue"
   android:gravity="center"
   >                                                    <!--LinearLayout-->
<TextView
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="@string/msg"
   android:textColor="@color/white"
   android:textSize="18dip"
   android:gravity="center_horizontal"
   />                                                   <!--TextView 控件-->
</LinearLayout>                                         <!--LinearLayout-->
```

提示：上述代码主要完成的是界面的搭建。

接下来要介绍的是 Sample4_4_Activity 类的设计与实现，该类的代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_4/src/com/bn/chap4/toast 目录下的 Sample4_4_Activity.class。

```
package com.bn.chap4.toast;
……//该处省略了部分类的导入代码，读者可自行查看随书光盘中源代码。
public class Sample4_4_Activity extends Activity{
    LooperThread lt;                                    //有消息循环的线程
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);                    //设置当前界面
      lt=new LooperThread(this);                        //创建消息循环的线程的对象
      lt.start();                                       //启动线程
      new Thread(){                                     //定时向消息循环线程发送消息
        public void run(){
            for(int i=1;i<=10;i++){                     //循环 10 次
                String msgStr="计数器值: "+i;           //创建内容字符串
                Bundle b=new Bundle();                  //创建消息数据 Bundle
                b.putString("msg", msgStr);             //将内容字符串放进数据 Bundle 中
                Message msg=new Message();              //创建消息对象
                msg.setData(b);                         //设置数据 Bundle 到消息中
                msg.what=Constant.DISPLAY_TOAST;        //设置消息的 what 值
                lt.hd.sendMessage(msg);                 //发送消息
                try {                                   //休息 3000ms 再发送下一条消息
                        Thread.sleep(3000);
                    } catch (InterruptedException e) {e.printStackTrace();}
            }}}.start();                                //启动线程
    }}
```

提示：上述代码主要完成的是创建线程对象，定时向消息循环线程发送消息，设定一共发送 10 条消息，在发送消息时创建字符串内容和 Bundle 对象，使用 Bundle 对象将信息传送过去，休息 3 秒钟后继续发送下一条消息。

接下来要介绍的是 LooperThread 类的设计与实现，该类的代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_4/src/com/bn/chap4/toast 目录下的 LooperThread.class。

```
1   package com.bn.chap4.toast;
2   ……//该处省略了部分类的导入代码，读者可自行查看随书光盘中源代码。
3   public class LooperThread extends Thread{          //开发有消息循环的线程
4       Handler hd;                                    //消息处理器
5       public Context context;                        //上下文
6       public LooperThread(Context context){
7           this.context=context;                      //获取上下文
8       }
9       public void run(){
10          Looper.prepare();                          //进行 prepare，建立消息队列
11          hd=new Handler(){                          //创建消息处理器对象
12          public void handleMessage(Message msg){
13              super.handleMessage(msg);              //调用父类处理
14              switch(msg.what){
15                case Constant.DISPLAY_TOAST:
16                    Bundle b=msg.getData();          //获取消息中的数据
17                    String msgStr=b.getString("msg");    //获取内容字符串
18                    Toast.makeText(                  //显示 Toast
19                        context,
20                        msgStr,
21                        Toast.LENGTH_SHORT           //显示时间
22                    ).show();
23                break;
24              }};
25          Looper.loop();                             //建立消息循环
26      }}
```

其中：

- 第 1～8 行为导入相关类代码和声明 Handler 引用以及上下文，在该类的构造器中获取上下文引用。
- 第 9～26 行为 run 方法，首先进行 prepare，建立消息队列，然后创建消息处理器对象，根据消息 what 编号的不同，执行不同的业务逻辑，在最后建立消息循环。

最后要介绍的是该软件的常量类，该软件常量类十分简单，仅含有一个变量，代码如下。

代码位置：见随书光盘中源代码/第 4 章/Sample4_4/src/com/bn/chap4/toast 目录下的 Constant .class。

```
1   package com.bn.chap4.toast;
2   ……//该处省略了部分类的导入代码，读者可自行查看随书光盘中源代码。
3   public class Constant{
4       public static final int DISPLAY_TOAST=0;          //显示 Toast 的消息编号
5   }
```

提示：上述代码为常量类，定义静态最终类型的 Toast 的消息编号。

实例 5　手机桌面心情

Android 平台下有一类非常实用的组件——wigde（桌面组件），其可以将应用程序的主要功能直接放置于桌面上，使用十分方便。在本节将对其进行介绍。

【实例描述】

运行该软件首先进入的是填写今日心情的界面，在该界面的文本编辑框中填写自己今天的心情，字符数不能超过 12 个，若超过则系统提示心情不能超过 12 个字。单击“确定”按钮后，系统中自动添加一项心情。

退出该软件，在手机屏幕上长按，会弹出选择添加到主屏幕窗口，然后选择窗口小部件，单击时间心情，会在手机屏幕上出现所填写的心情。本实例的运行效果图，如图 4-5 所示。

图 4-5　心情设置效果图

提示：在图 4-5 中依次表示的是运行本程序进入主界面，在主界面的文本框中输入今日心情。之后长按桌面弹出对话框，并在对话框中选择时间心情，在桌面上显示自己输入的心情以及时间的对话框。

【实现过程】

开发一个 widget 主要包含如下几个组成部分。

- widge 的布局 xml 文件，如 wmain.xml，此文件与普通布局的 xml 文件没有本质的不同。
- widge 的描述 xml 文件，如 appwidgetprovder.xml。
- widget 内容需要更新时接收广播 Intent 对象所属的 Java 类，其继承自 AppWidgeProvider，如 MyWidgetProvider。

在这三部分开发完毕后，需要在 AndroidManifest.xml 文件中对自己开发的 MyWidgetProvider 进行注册，将其注册为一个广播接收器。

当一个 widge 组件的状态发生变化，或收到此组件注册的更新广播时，系统会回调 MyWidgeProvider 中的某些方法，具体情况如表 4-1 所示。

表 4-1　回调方法名及回调的时机

回调方法名	回调的时机
onDeleted	当一个或多个 widget 实例被删除时回调
onDisabled	当最后一个 widget 实例被删除时回调
onEnabled	当第一个 widget 实例产生时调用
onUpdate	widget 在桌面上生成时调用，并更新组件 UI

提示：在实际开发中要根据需要重写恰当的方法。

在本软件中使用了三条广播消息（Intent），具体情况如表 4-2 所示。

表 4-2　广播消息及其作用

Action 字符串	作　用
wyf.action.update_xq	由 Activity 发给 widget，通知其更新心情内容
wyf.action.time_upadte	由后台 TimeService 发出，通知 widget 更新时间信息
wyf.action.load_xq	由后台 TimeService 发出，通知 widget 从持久性存储中加载心情信息，并给心情文本 view 注册单击事件监听器

【代码解析】

经过上面的介绍，相信读者对 widget 有了一定的了解，在本部分将对该软件的核心代码进行详细介绍，首先要介绍的是 main.xlm 文件，该文件的代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_5/res/layout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:orientation="vertical"
4     android:layout_width="fill_parent"
5     android:layout_height="fill_parent"
6     >                                          <!--LinearLayout-->
7    <TextView
8     android:text="今日心情"
9     android:id="@+id/TextView01"
10    android:layout_width="wrap_content"
11    android:layout_height="wrap_content"
12    android:textColor="#FFFFFF"
13    android:textSize="20dip"
14    android:clickable="true"
15  >
16  </TextView>                                  <!--EditText 控件-->
17  <EditText
18    android:id="@+id/EditText01"
19    android:singleLine="false"
20    android:layout_width="fill_parent"
21    android:layout_height="60dip"
22  >
23  </EditText>                                  <!--EditText 控件-->
24  <Button
25    android:text="确定"
26    android:id="@+id/Button01"
27    android:layout_width="wrap_content"
28    android:layout_height="wrap_content">
29  </Button>                                    <!--Button 控件-->
```

```
30  </LinearLayout>                                    <!--LinearLayout-->
```

提示：上述代码主要完成的是主界面的布局设置，在该界面中设有 TextView、EditTExt 和一个 Button 按钮，其属性设置可参看代码注释。

接下来要介绍的是 widge 的布局 xml 文件——wmain.xml 文件，该文件的代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_5/res/layout 目录下的 wmain.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="vertical"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent"
6      android:background="@drawable/dialog"
7      >                                                <!--LinearLayout-->
8     <TextView
9      android:text="请单击输入心情"
10     android:id="@+id/TextView01"
11     android:layout_width="fill_parent"
12     android:layout_height="wrap_content"
13     android:textColor="#FFFFFF"
14     android:textSize="24dip"
15     android:clickable="true"
16    >
17    </TextView>                                       <!--TextView 控件-->
18    <TextView
19     android:id="@+id/TextView02"
20     android:layout_width="fill_parent"
21     android:layout_height="wrap_content"
22     android:textColor="#FFFFFF"
23     android:textSize="24dip"
24    >
25    </TextView>                                       <!--TextView 控件-->
26  </LinearLayout>                                     <!--LinearLayout-->
```

提示：上述代码主要完成的是 widget 布局的设置，在该布局中包含一个 TextView 和一个 EditText 控件，其属性设置请参考代码注释。

widge 的描述 xml 文件——appwidgeprovder.xml 的介绍，该文件代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_5/res/xml 目录下的 appwidgeprovder.xml。

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"
3      android:minWidth="280dip"
4      android:minHeight="60dip"
5      android:initialLayout="@layout/wmain">
6   </appwidget-provider>
```

提示：上述代码中第 3、4 行为设定宽度和高度，第 5 行为给出桌面布局文件。

最后一个要介绍的 xml 文件是 AndroidManifest.xml，在该文件中需要完成对 receiver 的声明以及广播消息的注册，该文件代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_5 目录下的 AndroidManifest.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3        package="com.bn.chap4.widget"
```

```
4        android:versionCode="1"
5        android:versionName="1.0">
6       <application android:icon="@drawable/icon" android:label="@string/app_name">
7         <activity android:name=".Sample4_5_Activity"
8                 android:label="@string/app_name">
9            <intent-filter>
10              <action android:name="android.intent.action.MAIN" />
11              <category android:name="android.intent.category.LAUNCHER" />
12           </intent-filter>
13        </activity>
14        <receiver                                    <!--注册 receiver-->
15          android:name=".MyWidgetProvider"
16          android:label="时间心情"
17          android:icon="@drawable/heart"
18        >
19           <meta-data android:name="android.appwidget.provider"
20              android:resource="@xml/appwidgetprovder"></meta-data>
21           <intent-filter>                              <!--注册广播消息-->
22              <action android:name="wyf.action.time_upadte"></action>
23              <action android:name="wyf.action.update_xq"></action>
24              <action android:name="wyf.action.load_xq"></action>
25              <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
26           </intent-filter>
27        </receiver>
28        <service android:name=".TimeService" android:process=":remote"/>
29     </application>
30     <uses-sdk android:minSdkVersion="7" />
31  </manifest>
```

其中：

- 第 1～17 行为该文件的基本属性设置，其中第 14～18 行为注册 Receiver，并设定 label 和图标。
- 第 19～28 行为注册广播消息，其中 wyf.action.time_upadte 用于后台 TimeService 发出，通知 widget 更新时间信息，wyf.action.update_xq 用于 Activity 发给 widget，通知其更新心情内容，"wyf.action.load_xq"用于后台 TimeService 发出，通知 widget 从持久性存储中加载心情信息，并给心情文本 view 注册单击事件监听器。
- 第 29～31 行是声明 minSdkVersion 为 7。

该软件中 xml 文件的介绍到此就结束了，下面要介绍的是该软件中个各类的设计与实现，首先介绍的是主类 Sample4_5_Activity 的设计与实现，该类代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_5/src/com/bn/chap4/widget 目录下的 Sample4_5_Activity.class。

```
1   package com.bn.chap4.widget;
2   ……//该处省略了部分类的导入代码，读者可自行查看随书光盘中源代码。
3   public class Sample4_5_Activity extends Activity{
4       EditText et;
5       public void onCreate(Bundle savedInstanceState) {
6         super.onCreate(savedInstanceState);
7         setContentView(R.layout.main);                    //设置当前界面
8         et=(EditText)this.findViewById(R.id.EditText01);
                                                  /初始化指向输入心情的文本区的引用
9         Button b=(Button)this.findViewById(R.id.Button01);//获取确定按钮
10        b.setOnClickListener(                             //给确定按钮添加监听器
11          new OnClickListener(){
12                  public void onClick(View v){
13                      String msg=et.getText().toString();//获取文本框中输入的心情
14                      if(msg.trim().length()==0){ //若输入的心情为空则提示并返回
```

```
15                          Toast.makeText(              //显示 Toast
16                              Sample4_5_Activity.this,
17                              "心情不能为空!!! ",
18                              Toast.LENGTH_SHORT     //显示时间
19                          ).show();
20                          return;
21                      }
22                      else if(msg.length()>12){//若输入的心情超过长度则提示并返回
23                          Toast.makeText(              //显示 Toast
24                              Sample4_5_Activity.this,
25                              "心情不能大于 12 个字!!! ",
26                              Toast.LENGTH_SHORT     //显示时间
27                          ).show();
28                          return;
29                      }
30                  Intent intent = new Intent("wyf.action.update_xq");
                                                //发送 Intent 修改 widget 中的内容
31                  intent.putExtra("xxq", msg);          //绑定消息
32                  Sample4_5_Activity.this.sendBroadcast(intent);
33                  Sample4_5_Activity.this.finish();
                                                //发送完 Intent 结束 Activity
34  }});}}
```

其中：

- 第 1～9 行为声明成员变量，设置 main.xlm 为当前界面后，创建各控件对象。
- 第 10～29 行为给 Button 按钮添加监听器，在监听器内首先获取用户所输入的心情内容，判断内容字数是否为空或大于 12，若为空或大于 12 则提示已超过规定字数直接返回，否则进入下一步。
- 第 30～34 行为创建 Intent 对象，绑定消息后发送 wyf.action.update_xq 广播，结束该 Activity。

接下来要介绍的是继承自 Server 的 TimeService 类，在该类中主要完成的是定时更新心情的时间，该类代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_5/src/com/bn/chap4/widget 目录下的 TimeService.class。

```
1   package com.bn.chap4.widget;
2   ……//该处省略了部分类的导入代码，读者可自行查看随书光盘中源代码。
3   public class TimeService extends Service{
4       boolean flag=true;                                  //线程循环标志
5       Thread task;                                        //定时刷新时间的任务线程
6       public IBinder onBind(Intent arg0) {
7           return null;                                    //返回 null
8       }
9       public void onCreate(){
10          super.onCreate();
11          task=new Thread(){                              //创建定时更新时间的线程
12              public void run(){
13                  while(flag){
14                      Intent intent = new Intent("wyf.action.time_upadte");
                                                            //定时发送 Intent 更新时间
15                      Date d=new Date();                  //获取日期
16                      StringBuilder sb=new StringBuilder();
                                                            //创建 StringBuilder 对象
17                      sb.append(d.getYear()+1900);        //添加年份
18                      sb.append("年");
19                      sb.append((d.getMonth()+1<10)?"0":""); //添加月份
```

```
20                          sb.append(d.getMonth()+1);
21                          sb.append("月");
22                          sb.append((d.getDate()<10)?"0":"");     //添加日期
23                          sb.append(d.getDate());
24                          sb.append("日 ");
25                          sb.append((d.getHours()<10)?"0":"");    //添加小时
26                          sb.append(d.getHours());
27                          sb.append(":");
28                          sb.append((d.getMinutes()<10)?"0":""); //添加分钟
29                          sb.append(d.getMinutes());
30                          sb.append(":");
31                          sb.append((d.getSeconds()<10)?"0":""); //添加秒数
32                          sb.append(d.getSeconds());
33                          intent.putExtra("time", sb.toString());     //绑定时间
34                          TimeService.this.sendBroadcast(intent);
35                          intent = new Intent("wyf.action.load_xq");
                                                        //定时发送 Intent 更新心情
36                          TimeService.this.sendBroadcast(intent);
37                          try {
38                              Thread.sleep(500);          //线程睡眠 0.5 秒
39                          } catch (InterruptedException e) {e.printStackTrace();}
40                      }}};}
41      public void onStart(Intent intent, int id){
42          task.start();                                   //启动任务线程
43      }
44      public void onDestroy(){
45          flag=false;                                     //关闭定时更新时间的线程
46      }}
```

其中：

- 第 4～8 行为声明线程循环标志和定时刷新时间的任务线程，在 onBind 方法中因为本例不用 Bind 功能，因此直接返回 null。
- 第 9～40 行为重写 onCreate 方法，在该方法中创建线程，创建 Intent 对象，获取当前时间，并重新组装，最后将组装好的时间绑定在 Intent 对象上，并定时发送广播更改心情内容。
- 第 41～46 行为重写 onStart 方法和 onDestroy 方法，在这两个方法内启动任务线程以及关闭任务线程。

接下来要介绍的是 MyWidgetProvider 类的设计与实现，该类代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_5/src/com/bn/chap4/widget 目录下的 MyWidgetProvider.class。

```
1   package com.bn.chap4.widget;
2   ……//该处省略了部分类的导入代码，读者可自行查看随书光盘中源代码。
3   public class MyWidgetProvider extends AppWidgetProvider{
4       RemoteViews rv;
5       public MyWidgetProvider(){
6       }
7       public void onDisabled(Context context) {   //若为最后一个实例
8           context.stopService(new Intent(context,TimeService.class));
                                                    //停止后台定时更新 Widget 时间
9       }
10      public void onEnabled (Context context) {   //若为第一个实例则打开服务
11          context.startService(new Intent(context,TimeService.class));
                                                    //启动后台定时更新时间
12      }
13      public void onUpdate(Context context, AppWidgetManager appWidgetManager,
```

```
    int[] appWidgetIds) {
        rv = new RemoteViews(context.getPackageName(), R.layout.wmain);
                                                        //创建 RemoteViews
      Intent intent = new Intent(context,Sample4_5_Activity.class);
                                                        //创建修改心情的 Intent
      PendingIntent pendingIntent=PendingIntent.getActivity(
                                                        //创建 PendingIntent
            context,                                    //上下文
            0,
            intent,
            PendingIntent.FLAG_UPDATE_CURRENT
      );
      rv.setOnClickPendingIntent(R.id.TextView01, pendingIntent);
                                                  //按下文本框发送此 PendingIntent
      SharedPreferences              sp=context.getSharedPreferences("xqsj",
Context.MODE_PRIVATE);
      String xqStr=sp.getString(                        //读取上次的心情
            "xq",                                       //键值
            null                                        //默认值
      );
      if(xqStr!=null){                                  //若上次心情存在则更新心情
        rv.setTextViewText(R.id.TextView01, xqStr);
      }
      appWidgetManager.updateAppWidget(appWidgetIds, rv);    //更新 Widget
    }
   public void onReceive(Context context, Intent intent){ //接收广播时调用更新 UI
     super.onReceive(context, intent);
      if (rv == null) {                                 //创建 RemoteViews
        rv = new RemoteViews(context.getPackageName(), R.layout.wmain);
      }
      if (intent.getAction().equals("wyf.action.update_xq")) {
                                                        //收到更新心情的 Action
        rv.setTextViewText(R.id.TextView01, intent.getStringExtra("xxq"));
        SharedPreferences sp=context.getSharedPreferences("xqsj", Context.
MODE_PRIVATE);
        SharedPreferences.Editor editor=sp.edit(); //写入心情
        editor.putString("xq",intent.getStringExtra("xxq"));
        editor.commit();
      }
      else if (intent.getAction().equals("wyf.action.time_upadte")){
                                                        //收到更新时间的 Action
        rv.setTextViewText(R.id.TextView02,intent.getStringExtra("time"));
                                                        //更新时间
      }
      else if (intent.getAction().equals("wyf.action.load_xq"))
      {
        Intent intentTemp = new Intent(context,Sample4_5_Activity.class);
                                                        //创建 Intent
        PendingIntent pendingIntent=PendingIntent.getActivity(
                                                        //创建 PendingIntent
            context,
            0,
            intentTemp,
            PendingIntent.FLAG_UPDATE_CURRENT
        );
        rv.setOnClickPendingIntent(R.id.TextView01, pendingIntent);
                                                  //按下文本框发送此 PendingIntent
        SharedPreferences sp=context.getSharedPreferences("xqsj", Context.
MODE_PRIVATE);
        String xqStr=sp.getString(                      //读取上次的心情
```

```
61                      "xq",                                   //键值
62                      null                                    //默认值
63              );
64              if(xqStr!=null){                                //若上次心情存在则更新心情
65                  rv.setTextViewText(R.id.TextView01, xqStr);
66              }}
67               AppWidgetManager  appWidgetManger  =  AppWidgetManager.getInstance
(context);
68               int[] appIds = appWidgetManger.getAppWidgetIds(     //创建数组
69               new ComponentName
70               (
71                  context,                                    //上下文
72                  MyWidgetProvider.class
73              ));
74            appWidgetManger.updateAppWidget(appIds, rv);      //更新心情
75        }}
```

其中：

- 第 4～12 行为创建空构造器，重写 onDisabled 和 onEnabled 方法，分别完成停止后台定时更新 widget 时间和启动后台定时更新时间的任务。
- 第 13～33 行为重写 onUpdate 方法，该方法为组件在桌面上生成时调用，并更新组件 UI；在该方法中创建 SharedPreferences 对象记录上次的心情。
- 第 34～75 行为若 RemoteViews 对象为空，则首先创建 RemoteViews 对象；若收到更新心情的 Action，则更新心情，并向 SharedPreferences 中写入心情；若更新时间的 Action，则更新时间；若收到更新心情并添加监听，则更新心情并添加监听；为切屏（横竖屏切换）服务，可创建 Intent 和 PendingIntent 对象，将心情写入 SharedPreferences，并更新心情。

提示：特别注意此类对象每次收到消息后系统会创建一个新对象，因此此类对象不可用于存储状态。

实例 6　应用选项菜单的综合技巧

想要让 Android 应用程序有更完善的用户体验，除了设计人性化的用户界面以外，添加一些菜单也是必要的。Android 平台所提供的菜单大体可分为三类：选项菜单、上下文菜单和子菜单。在本节将对选项菜单进行介绍。

【实例描述】

本软件设计成为选择自己的性别和爱好，当运行该软件时，进入主界面，单击手机上的菜单按钮，在手机屏幕的下方会弹出三个选项，单击性别时，弹出性别选择对话框，单击选择自己的性别回到主界面，并在主界面上打印所选择的性别。

单击爱好按钮时，弹出爱好对话框，选择自己的爱好后回到主界面。并在主界面上打印所选择的爱好。单击“确定”按钮则打印一遍所选的信息。本实例的运行效果图，如图 4-6 所示。

提示：在图 4-6 中依次表示的是在本程序的主界面单击“menu”键弹出菜单，单击菜单中的性别选项，弹出性别选择对话框，单击菜单中的爱好选项，弹出爱好选择对话框。

图 4-6　运行效果图

【实现过程】

在该软件中主要使用的是 SubMenu 类，通过该类创建菜单选项按钮，创建菜单项目的编号，为其设定级别，随后在重写的 onCreateOptionsMenu 方法中，完成选项菜单的设置。

重写单选或复选菜单项选中状态变化后的回调方法 onOptionsItemSelected，完成所选信息的更新。

【代码解析】

在本部分要介绍的是核心代码，首先介绍的是 main.xml 的设计与实现。

代码位置：见随书光盘中源代码/第 4 章/Sample4_6/res/layout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout
3       android:id="@+id/LinearLayout01"
4       android:layout_width="fill_parent"
5       android:layout_height="fill_parent"
6       android:orientation="vertical"
7       xmlns:android="http://schemas.android.com/apk/res/android">
                                              <!--LinearLayout-->
8      <ScrollView
9       android:id="@+id/ScrollView01"
10      android:layout_width="fill_parent"
11      android:layout_height="fill_parent">
12        <EditText
13          android:id="@+id/EditText01"
14          android:layout_width="fill_parent"
15          android:layout_height="fill_parent"
16          android:singleLine="false"
17          android:text="@string/label">
18        </EditText>                          <!--EditText 控件-->
19     </ScrollView>                           <!--ScrollView 控件-->
20  </LinearLayout>                            <!--LinearLayout-->
```

提示：上述代码主要完成的是该软件主界面的布局设计，在主界面中包含一个 EditText 控件。

Sample4_6_Activity 类的设计与实现，该类的代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_6/src/com/bn/chap4/widget 目录下的 Sample4_6_Activity.class。

```
1   package com.bn.chap4.xxcd;
2   ……//该处省略了部分类的导入代码，读者可自行查看随书光盘中源代码。
3   public class Sample4_6_Activity extends Activity {
4      public void onCreate(Bundle savedInstanceState) {
5         super.onCreate(savedInstanceState);
6         setContentView(R.layout.main);                    //设置主界面
7      }
8      final int MENU_GENDER_MALE=0;                        //每个菜单项目的编号
9      final int MENU_GENDER_FEMALE=1;
10     final int MENU_HOBBY1=2;
11     final int MENU_HOBBY2=3;
12     final int MENU_HOBBY3=4;
13     final int MENU_OK=5;
14     final int MENU_GENDER=6;
15     final int MENU_HOBBY=7;                              //每个菜单项目的编号
16     final int GENDER_GROUP=0;                            //性别子菜单项组的编号
17     final int HOBBY_GROUP=1;                             //爱好子菜单项组的编号
18     final int MAIN_GROUP=2;                              //外层总菜单项组的编号
19     MenuItem[] miaHobby=new MenuItem[3];                 //爱好菜单项组
20     MenuItem male=null;                                  //男性性别菜单项
21     public void appendStateStr(){                        //获取当前选择状态的方法
22      String result="您选择的性别为：";
23      if(male.isChecked()){                               //若为选中状态
24          result=result+"男";
25      }else{                                              //若没有选中
26          result=result+"女";
27      }
28      String hobbyStr="";
29      for(MenuItem mi:miaHobby){                          //循环扫描
30          if(mi.isChecked()){
31              hobbyStr=hobbyStr+mi.getTitle()+"、";   //添加信息
32          }}
33      if(hobbyStr.length()>0){                            //若字符长度大于 0
34          result=result+",您的爱好为："+hobbyStr.substring(0, hobbyStr.length()
-1)+"。\n";
35      }else{
36          result=result+"。\n";                           //否则添加句号
37      }
38      EditText et=(EditText)Sample4_6_Activity.this.findViewById(R.id.Edit
Text01);
39          et.append(result);                              //添加信息
40     }
41     public boolean onCreateOptionsMenu(Menu menu){
42      SubMenu subMenuGender = menu.addSubMenu(    //性别单选菜单项组
43          MAIN_GROUP,MENU_GENDER,0,R.string.gender);
44      subMenuGender.setIcon(R.drawable.gender);   //设置性别图标
45      male=subMenuGender.add(GENDER_GROUP, MENU_GENDER_MALE, 0, R.string.male);
46      male.setChecked(true);
47      subMenuGender.add(GENDER_GROUP, MENU_GENDER_FEMALE, 0, R.string.female);
48      //设置 GENDER_GROUP 组是可选择的，互斥的
49      subMenuGender.setGroupCheckable(GENDER_GROUP, true,true);
50      //爱好复选菜单项组
51      SubMenu subMenuHobby = menu.addSubMenu(MAIN_GROUP,MENU_HOBBY,0,R.string.
hobby);
```

```
52      subMenuHobby.setIcon(R.drawable.hobby);          //设置爱好图标
53      miaHobby[0]=subMenuHobby.add(HOBBY_GROUP, MENU_HOBBY1, 0, R.string.hobby1);
54      miaHobby[1]=subMenuHobby.add(HOBBY_GROUP, MENU_HOBBY2, 0, R.string.hobby2);
55      miaHobby[2]=subMenuHobby.add(HOBBY_GROUP, MENU_HOBBY3, 0, R.string.hobby3);
56      miaHobby[0].setCheckable(true);                  //设置菜单项为复选菜单项
57      miaHobby[1].setCheckable(true);
58      miaHobby[2].setCheckable(true);
59      MenuItem ok=menu.add(GENDER_GROUP+2,MENU_OK,0,R.string.ok);//确定菜单项
60      OnMenuItemClickListener lsn=new OnMenuItemClickListener(){
                                                  //实现菜单项单击事件监听接口
61              public boolean onMenuItemClick(MenuItem item) {
62                  appendStateStr();
63                  return true;
64              }};
65      ok.setOnMenuItemClickListener(lsn);          //给确定菜单项添加监听器
66      ok.setAlphabeticShortcut('o');               //设置字符快捷键
67      return true;
68    }
69    public boolean onOptionsItemSelected(MenuItem mi){
70      switch(mi.getItemId()){
71        case MENU_GENDER_MALE:                     //单选菜单项状态的切换
72        case MENU_GENDER_FEMALE:
73             mi.setChecked(true);
74             appendStateStr();                     //当有效项目变化时记录在文本区中
75        break;
76        case MENU_HOBBY1:                          //复选菜单项状态的切换
77        case MENU_HOBBY2:
78        case MENU_HOBBY3:
79           mi.setChecked(!mi.isChecked());
80           appendStateStr();                       //当有效项目变化时记录在文本区中
81        break;
82      }
83      return true;
84    }}
```

其中：

- 第 1～20 行为设置 main.xml 为当前界面，并创建每个菜单项目的编号、性别子菜单项组的编号、爱好子菜单项组的编号和外层总菜单项组的编号。
- 第 21～40 行为组装字符串的方法，该方法主要完成的是对获取字符的重新组装。
- 第 41～68 行为重写 onCreateOptionsMenu，在该方法中依次设置性别和爱好的图标，设置性别 GENDER_GROUP 组是可选择的，互斥的；爱好选项为复选菜单项，并为菜单添加监听器。
- 第 69～84 行为单选或复选菜单项选中状态变化后的回调方法，单选菜单项的状态切换时，记录在文本区。

实例 7　上下文菜单的应用

上下文与选项菜单不同，选项菜单是为整个 Activit 服务的，而上下文菜单是注册到某个 View 的。在本节将对其进行详细介绍。

【实例描述】

该软件的主界面含有两个文本输入框，默认情况下用户可以通过长按（约 2 秒）相应 View 以呼出上下文菜单。单击弹出对话框选项时，则在相应的文本框中显示所单击的选项。本实例

的运行效果图，如图 4-7 所示。

图 4-7　上下文菜单运行效果图

提示：在图 4-7 中依次表示的是运行本程序进入主界面，在主界面长按文本框弹出编辑文字对话框，选中该文本框中的某一项可以在主界面显示自己的选择。

【实现过程】

软件中当一个 View 注册了上下文菜单后，长按 2 秒钟左右即可打开包含自定义选项的对话框。需要重写菜单项选中状态变化后的回调方法 onCreateContextMenu 和菜单项选中状态变化后的回调方法 onContextItemSelected。

【代码解析】

在本部分首先介绍的是 main.xml 文件，该文件的代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_6/res/layout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout                                    <!--设置 LinearLayout 属性-->
3      android:id="@+id/LinearLayout01"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent"
6      android:orientation="vertical"
7      xmlns:android="http://schemas.android.com/apk/res/android">
                                                     <!--LinearLayout-->
8       <EditText
9         android:text="@string/et1"
10        android:id="@+id/EditText01"
11        android:layout_width="fill_parent"
12        android:layout_height="wrap_content"
13        xmlns:android="http://schemas.android.com/apk/res/android">
14      </EditText>                                  <!--EditText 控件-->
15      <EditText
16        android:text="@string/et2"
17        android:id="@+id/EditText02"
18        android:layout_width="fill_parent"
19        android:layout_height="wrap_content"
20        xmlns:android="http://schemas.android.com/apk/res/android">
```

```
21      </EditText>                                    <!--EditText 控件-->
22  </LinearLayout>                                    <!--LinearLayout-->
```

其中：

- 第 1～7 行为设置 LinearLayout 属性，设置其内部控件摆放顺序为竖直排放。
- 第 8～21 行为向 LinearLayout 中添加两个 EditText，并设置 EditText 的显示文本、id 和长宽，同时指定 xmlns 属性。

接下来要介绍的是 Sample4_7_Activity 类的设计与实现，该类的代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_7/src/com/bn/chap4/sxwcd 目录下的 Sample4_7_Activity.class。

```
1   package com.bn.chap4.sxwcd;
2   ……//该处省略了部分类的导入代码，读者可自行查看随书光盘中源代码。
3   public class Sample4_7_Activity extends Activity {
4      public void onCreate(Bundle savedInstanceState) {
5         super.onCreate(savedInstanceState);
6         setContentView(R.layout.main);              //设置当前界面
7         this.registerForContextMenu(findViewById(R.id.EditText01));
                                                      //为两个文本框注册上下文菜单
8         this.registerForContextMenu(findViewById(R.id.EditText02));
9      }
10     final int MENU1=1;                             //每个菜单项目的编号
11     final int MENU2=2;
12     final int MENU3=3;
13     final int MENU4=4;
14     final int MENU5=5;                             //每个菜单项目的编号
15     public void onCreateContextMenu (ContextMenu menu, View v,
16      ContextMenu.ContextMenuInfo menuInfo) {       //每次调出上下文菜单时被调用
17      if(v==findViewById(R.id.EditText01)){         //若是第一个文本框
18          menu.add(0, MENU1, 0, R.string.mi1);      //添加子菜单
19          menu.add(0, MENU2, 0, R.string.mi2);
20          menu.add(0, MENU3, 0, R.string.mi3);
21      }
22      else if(v==findViewById(R.id.EditText02)){    //若是第二个文本框
23          menu.add(0, MENU4, 0, R.string.mi4);      //添加子菜单
24          menu.add(0, MENU5, 0, R.string.mi5);
25      }
26     }
27     public boolean onContextItemSelected(MenuItem mi){
                                                      //菜单项选中状态变化后回调方法
28          switch(mi.getItemId()){
29            case MENU1:
30            case MENU2:
31            case MENU3:                             //向第一个文本框中添加文本内容
32                EditText et1=(EditText)this.findViewById(R.id.EditText01);
33                et1.append("\n"+mi.getTitle()+" 被按下");
34            break;
35            case MENU4:
36            case MENU5:                             //向第二个文本框中添加内容
37                EditText et2=(EditText)this.findViewById(R.id.EditText02);
38                et2.append("\n"+mi.getTitle()+" 被按下");
39            break;
40          }
41      return true;
42     }}
```

其中：

- 第 4～14 行为设置 main.xml 为当前主界面，并为两个文本框注册上下文菜单，其中第

10～14 行为每个菜单选项的编号。

- 第 15～26 行为重写 onCreateContextMenu 方法，该方法为每次调出上下文菜单时被调用，主要完成的是添加子菜单；重写 onContextItemSelected 方法，在菜单项选中状态变化后回调的方法，主要完成向文本框中添加内容。

实例 8　手机背景颜色的设置

手机的应用程序默认的背景颜色是黑色的，有时是不满足开发需求的，Android 手机中可以自定义应用程序的背景色，在本节将对其进行介绍。

【实例描述】

该软件的主界面包含两个按钮，当单击按钮时更改手机的背景颜色，单击“深灰”按钮时，将手机背景设置为深灰，单击“浅灰”按钮将手机背景色设置为浅灰。本实例的运行效果图，如图 4-8 所示。

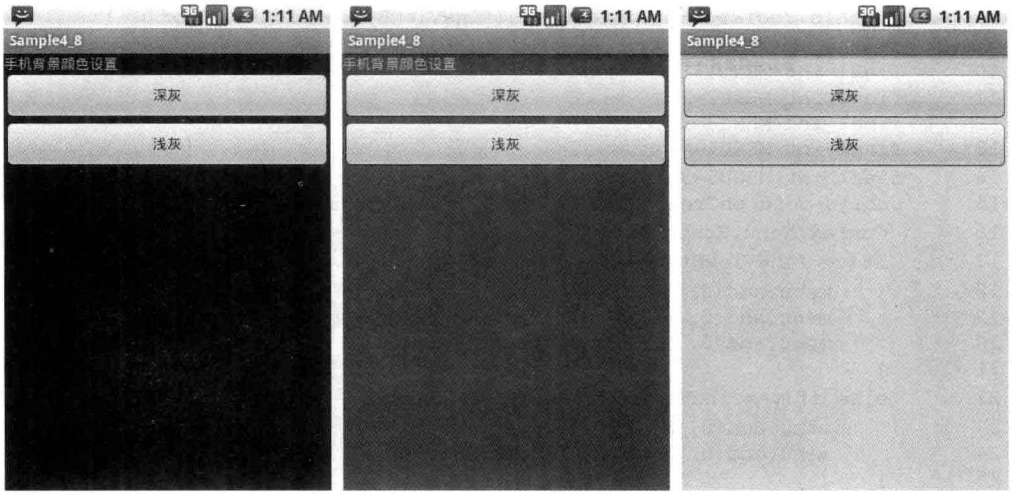


图 4-8　手机背景颜色的设置

提示：在图 4-8 中依次表示的是运行本程序进入主界面，在主界面单击“深灰”按钮改变背景颜色为深灰色，单击“浅灰”按钮改变背景颜色为浅灰色。

【实现过程】

在本实例中主要运用了 LinearLayout 的 setBackgroundColo() 方法和 Button 的 setOnClickListener()方法。

【代码解析】

经过上面的理论介绍，相信读者对此已有所了解，接下来要介绍的是界面布局的设置，代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_8/res/layout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:id="@+id/LinearLayout01" >                    <!--LinearLayout-->
   <TextView
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="手机背景颜色设置"
   />                                                   <!--TextView 控件-->
    <Button
        android:text="深灰"
        android:id="@+id/Button01"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">
    </Button>                                           <!--Button 控件-->
    <Button
        android:text="浅灰"
        android:id="@+id/Button02"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">
    </Button>                                           <!--Button 控件-->
</LinearLayout>                                         <!--LinearLayout-->
```

提示：上述代码主要完成的是软件主界面布局的设置，其中包括 TextView 和两个 Button 的属性设置。

然后需要介绍的是 Sample4_8_Activity 类的设计与实现，该类的代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_8/src/com/bn/chap4/phonebg 目录下的 Sample4_8_Activity.class。

```
package com.bn.chap4.phonebg;
……//该处省略了部分类的导入代码，读者可自行查看随书光盘中源代码。
public class Sample4_8_Activity extends Activity {
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);                                  //设置当前界面
      Button red=(Button)this.findViewById(R.id.Button01);  //深灰按钮
      Button blue=(Button)this.findViewById(R.id.Button02); /浅灰按钮
      final LinearLayout ll=(LinearLayout)this.findViewById(R.id.
LinearLayout01);
      red.setOnClickListener(                                         //添加监听器
        new OnClickListener(){
                public void onClick(View v) {
                    ll.setBackgroundColor(Color.DKGRAY);    //设置为深灰
                }});
      blue.setOnClickListener(                                        //添加监听器
        new OnClickListener(){
                public void onClick(View v) {
                    ll.setBackgroundColor(Color.LTGRAY);    //设置为浅灰
                }});}}
```

其中：

- 第 4～9 行为首先设置 mian.xml 为当前界面，声明深灰和浅灰按钮的引用，创建 LinearLayout 对象。
- 第 10～19 主要完成的是深灰按钮和浅灰按钮的监听器设置，在监听器内完成对手机屏幕背景颜色的设置，颜色设置所调用的方法为 setBackgroundColor()。

实例 9　字体颜色的变换

手机背景颜色可以进行自己设置，同样字体的颜色也可以自己设置。在本节将对其进行详细介绍。

【实例描述】

软件的界面中包含一行文本和两个 Button 按钮，一个按钮实现的功能是将字体的颜色设置为浅灰，另一个按钮实现的功能为将字体的颜色设置为白色，设置字体颜色的效果如图 4-9 所示。

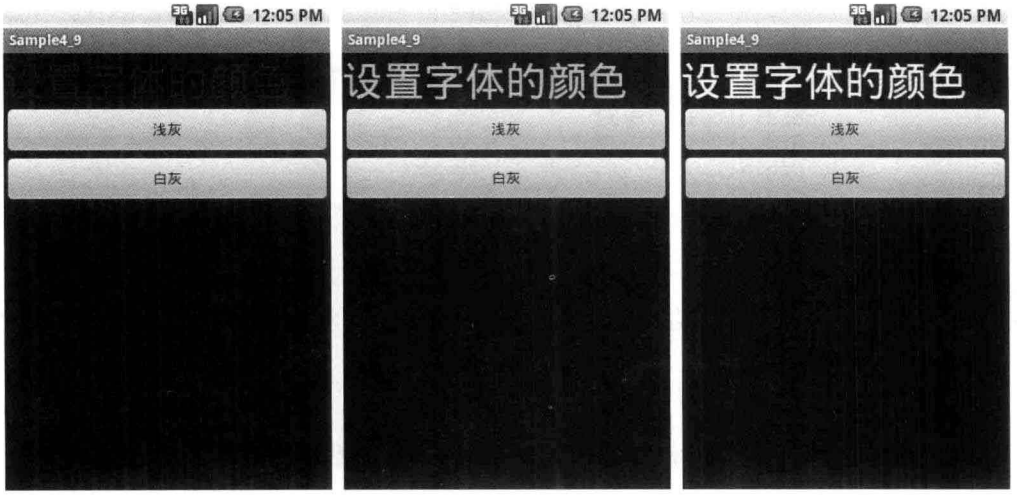


图 4-9　字体颜色的设置

提示：在图 4-9 中依次表示的是运行本程序进入主界面，在主界面单击“浅灰”按钮改变字体的颜色为浅灰色，单击“白色”按钮改变字体的颜色为白色。

【实现过程】

本软件中主要运用了 TextView 的 setTextColor()方法以及 Button 按钮的监听器的设置——setOnClickListener()方法。

【代码解析】

经过前面的理论介绍，接下来要介绍的是主界面的布局设置 main.xml 的实现，该文件的代码如下。

代码位置：见随书光盘中源代码/第 4 章/Sample4_9/res/layout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3       android:orientation="vertical"
4       android:layout_width="fill_parent"
5       android:layout_height="fill_parent"
6       android:background="#000000">                    <!--LinearLayout-->
7       <TextView
8       android:layout_width="fill_parent"
```

```
9       android:layout_height="wrap_content"
10      android:text="设置字体的颜色"      android:textSize="40dip"
11      android:id="@+id/TextView01"
12      android:textColor="#222222"
13      />                                                    <!--TextView 控件-->
14       <Button
15           android:text="浅灰"
16           android:id="@+id/Button01"
17           android:layout_width="fill_parent"
18           android:layout_height="wrap_content">
19       </Button>                                            <!--Button 控件-->
20       <Button
21           android:text="白色"
22           android:id="@+id/Button02"
23           android:layout_width="fill_parent"
24           android:layout_height="wrap_content">
25       </Button>                                            <!--Button 控件-->
26  </LinearLayout>                                           <!--LinearLayout-->
```

其中：

- 第 1～6 行为设置 LinearLayout，设置背景色为黑色，控件摆放顺序为竖直摆放。
- 第 7～25 行为 TextView 和两个 Button 按钮的属性设置，在 TextView 中设置字体颜色；按钮在宽度上充满屏幕，高度上包裹内容。

随后就是对 Sample4_9_Activity 类的设计与实现，该类的源代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_9/src/com/bn/chap4/changecolor 目录下的 Sample4_9_Activity.class。

```
1   package com.bn.chap4.changecolor;
2   ……//该处省略了部分类的导入代码，读者可自行查看随书光盘中源代码。
3   public class Sample4_9_Activity extends Activity {
4      public void onCreate(Bundle savedInstanceState) {
5         super.onCreate(savedInstanceState);
6         setContentView(R.layout.main);                          //设置当前界面
7         Button button0=(Button)this.findViewById(R.id.Button01);//浅灰色按钮
8         Button button0=(Button)this.findViewById(R.id.Button02);   //白色按钮
9         final TextView tv=(TextView)this.findViewById(R.id.TextView01);
10        button0.setOnClickListener(                          //添加监听器
11          new OnClickListener(){
12                  public void onClick(View v) {
13                      tv.setTextColor(Color .LTGRAY);
                                                  //设置 TextView 中字体颜色为浅灰
14                  }});
15        button1.setOnClickListener (
16          new OnClickListener(){                                  //添加监听器
17                  public void onClick(View v) {
18                      tv.setTextColor(Color .WHITE);
                                                  //设置 TextView 中字体颜色为白色
19                  }});}}
```

其中：

- 第 5～9 行设置 main.xml 为当前界面，声明按钮和 TextView 引用。
- 第 10～19 行主要为添加按钮的监听器，在监听器内设置字体的颜色。

实例 10　实现手机界面的置换

在应用程序中，最多的就是界面之间的切换，在本节将对实现界面切换的 setContentView()

方法进行介绍。

【实例描述】

该软件模拟登录界面，若登录成功则切换到下一界面。首先在主界面中需要填写账号和密码，若未填写账号或者密码，则系统提示需要账号和密码不能为空，单击登录按钮，成功进入下一界面，单击清空按钮将账号和密码清空。本实例的运行效果图，如图 4-10 所示。

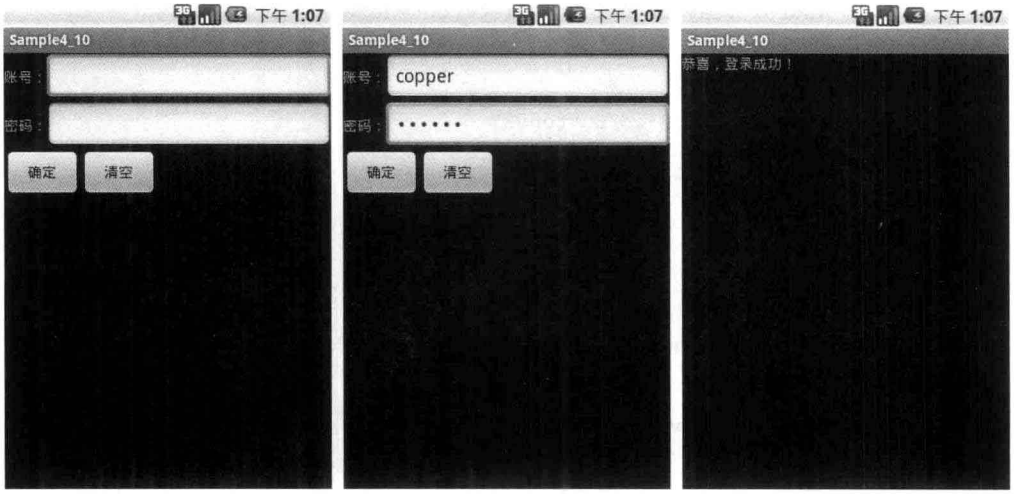


图 4-10　手机界面间的转换

提示：在图 4-10 中依次表示的是运行本程序进入主界面，在主界面的文本框中输入账号及密码，单击“确定”按钮，在主界面显示用户登录是否成功的信息。

【实现过程】

在这里界面的设置是通过 setContentView 实现的，在主程序中首先使用该方法设置 main.xml 为当前界面，然后在 Button 按钮的监听器内设置 other.xml 为成功登录后的界面。

【代码解析】

首先需要介绍的是 main.xml 文件，该文件完成的是登录界面的布局设置，代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_10/res/layout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="vertical"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent"
6      >                                                    <!--LinearLayout--
7      <LinearLayout
8       android:id="@+id/LinearLayout01"
9       android:layout_width="fill_parent"
10      android:layout_height="wrap_content"
11      android:orientation="horizontal">
12      <TextView
13              android:layout_width="wrap_content"
14              android:layout_height="wrap_content"
```

```
15              android:text="账号："
16          />                                          <!--TextView 控件-->
17          <EditText
18              android:id="@+id/EditText01"
19              android:layout_width="fill_parent"
20              android:layout_height="wrap_content">
21          </EditText>                                 <!--TextView 控件-->
22      </LinearLayout>
23      <LinearLayout                                   <!--LinearLayout-->
24       android:id="@+id/LinearLayout02"
25       android:layout_width="fill_parent"
26       android:layout_height="wrap_content"
27       android:orientation="horizontal">
28       <TextView
29              android:layout_width="wrap_content"
30              android:layout_height="wrap_content"
31              android:text="密码："
32          />                                          <!--TextView 控件-->
33          <EditText
34              android:id="@+id/EditText02"
35              android:layout_width="fill_parent"
36              android:layout_height="wrap_content"
37              android:password="true">
38          </EditText>                                 <!--TextView 控件-->
39      </LinearLayout>
40      <LinearLayout
41       android:id="@+id/LinearLayout02"
42       android:layout_width="fill_parent"
43       android:layout_height="wrap_content"
44       android:orientation="horizontal">
45       <Button
46          android:text="确定"
47          android:id="@+id/Button01"
48          android:layout_width="75dip"
49          android:layout_height="wrap_content">
50       </Button>                                      <!--Button 控件-->
51       <Button
52          android:text="清空"
53          android:id="@+id/Button02"
54          android:layout_width="75dip"
55          android:layout_height="wrap_content">
56       </Button>                                      <!--Button 控件-->
57      </LinearLayout>
58  </LinearLayout>
```

其中：

- 第 7～22 行为界面中最上面一排控件的属性设置。
- 第 23～39 行为界面中间一排控件的属性设置。
- 第 40～57 行为界面中最后一排 Button 控件的属性设置。

接下来要介绍的是登录成功后界面的设置，该界面主要完成的是登录成功的提醒，该文件代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_10/res/layout 目录下的 other.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="vertical"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent"
6      >
```

```
7   <TextView
8      android:layout_width="fill_parent"
9      android:layout_height="wrap_content"
10     android:text="恭喜，登录成功！"
11     />                                          <!--TextView 控件-->
12  </LinearLayout>                                <!--LinearLayout-->
```

提示：上述代码中在 LinearLayout 中添加了一个 TextView，并设置其文本、控件的长度和宽度。

最后要介绍的是 Sample4_10_Activity 类的设计与实现，在该类中实现界面之间的转换，代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_10/src/com/bn/chap4/setview 目录下的 Sample4_10_Activity.class。

```
1   package com.bn.chap4.setview;
2   import android.app.Activity;                          //导入相关包
3   import android.os.Bundle;
4   import android.view.View;                             //导入相关包
5   import android.view.View.OnClickListener;
6   import android.widget.Button;                         //导入相关包
7   import android.widget.EditText;
8   import android.widget.Toast;                          //导入相关包
9   public class Sample4_10_Activity extends Activity {
10     public void onCreate(Bundle savedInstanceState) {
11        super.onCreate(savedInstanceState);
12        setContentView(R.layout.main);                  //设置当前界面
13        final EditText user=(EditText)this.findViewById(R.id.EditText01);
                                                          //用户名
14        final EditText pwd=(EditText)this.findViewById(R.id.EditText02);
                                                          //密码
15        Button bOk=(Button)this.findViewById(R.id.Button01);
                                                          //确定按钮
16        Button bClear=(Button)this.findViewById(R.id.Button02);    //清空按钮
17        bOk.setOnClickListener(                         //确定添加监听器
18           new OnClickListener(){
19                   public void onClick(View v) {
20                       String strU=user.getText().toString().trim();
                                                          //获取账号
21                       String strP=pwd.getText().toString().trim();
                                                          //获取密码
22                       if(strU.length()!=0&&strP.length()!=0){
                                                          //若账号密码不为空
23                           setContentView(R.layout.other);    //进入另一个界面
24                       }else{
25                           Toast.makeText(                    //创建 Toast
26                                   Sample4_10_Activity.this,
27                                   "请填写账号和密码！",
28                                   Toast.LENGTH_SHORT).show(); //显示 Toast
29                       }}});
30        bClear.setOnClickListener(                      //清空按钮监听器
31           new OnClickListener(){
32                   public void onClick(View v) {
33                       user.setText("");                  //设置为空
34                       pwd.setText("");                   //设置为空
35  }});}}
```

其中：

- 第 2～16 行为相关包的导入，以及设置 main.xml 文件为当前界面并创建 EditText 和 Button 对象。
- 第 17～29 行为确定按钮的监听器，在该监听器内首先获取账号和密码，若判断用户名或密码不为空，则使用 setContentView 方法将 other.xml 设置为当前界面，否则使用 Toast 进行提示。
- 第 30～35 行为清空按钮的监听器，该监听器主要完成的任务为将账号和密码设置为空。

实例 11　活用信使启动新界面

应用程序间的界面切换有时候仅仅使用 setContentView()方法是不能够满足要求的，在这个时候就需要以另外一种方式实现界面切换这项功能，在本节将对 Intent 的使用进行详细介绍。

【实例描述】

在主界面中包含一个 Button 按钮，当单击“确定”按钮时，则系统进入第二个界面，在第二个界面中同样包含一个按钮，当单击“返回”按钮时，重新回到第一个界面。本实例的运行效果图，如图 4-11 所示。

图 4-11　Activity 间的转换

提示：在图 4-11 中依次为软件运行进入主界面，在主界面单击“确定”按钮，跳转至另一个界面。

【实现过程】

使用 Intent 实现两个 Activity 间的数据通信首先需要创建 Itent 对象，随后使用 setClass()方法设定当前的 Activity 类和需要启动的 Activity 类，使用 startActivity()方法启动另一个 Activity。

想要成功启动另一个 Activity，除了上述步骤外还需要在 AndroidManifest.xml 添加 Activity 的声明，代码为<activity android:name="AnotherActivity"></activity>。

【代码解析】

经过前面的理论介绍，相信读者对此方法已有所了解，下面就对该应用的核心代码进行详细介绍，首先要介绍的是 main.xml 文件的实现，该文件代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_11/res/layout 目录下的 main.xml。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   >                                              <!--LinearLayout-->
    <TextView
       android:layout_width="fill_parent"
       android:layout_height="wrap_content"
       android:text="单击进入下一界面"
   />                                             <!--TextView 控件-->
    <Button
        android:text="确定"
        android:id="@+id/Button01"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">
    </Button>                                     <!--Button 控件-->
</LinearLayout>
```

提示：上述代码主要完成的是在 LinearLayout 中添加一个 TextView 和一个 Button 控件。

随后是第二个 Activity 使用的 anthor.xml 文件，该文件的代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_11/res/layout 目录下的 anthor.xml。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:background="#ffffcc">                    <!--LinearLayout-->
   <TextView
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="第二个 Activity"
   android:textColor="#222222"/>                    <!--TextView 控件-->
   <Button
        android:text="返回"
        android:id="@+id/Button02"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">
    </Button>                                       <!--Button 控件-->
</LinearLayout>
```

提示：上述代码与 main.xml 的基本相似，在 LinearLayout 中设置 TextView 和 Button。

界面布局介绍完毕后，下一步要做的就是在 AndroidManifest.xml 中声明 AnotherActivity，若没有声明，则系统在运行时会抛出异常，终止程序。该文件的源代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_11 目录下的 AndroidManifest.xml。

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.bn.chap4.intent"
    android:versionCode="1"
    android:versionName="1.0">
   <application android:icon="@drawable/icon" android:label="@string/app_name">
     <activity android:name=".Sample4_11_Activity"
            android:label="@string/app_name">
        <intent-filter>
           <action android:name="android.intent.action.MAIN" />
           <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
     </activity>
       <activity android:name="AnotherActivity"></activity>
   </application>
</manifest>
```

提示：上述代码中第 14 行为声明 AnotherActivity 类。

基本配置结束后，接下来要介绍的就是 Sample4_11_Activity 和 AnotherActivity 类的设计与实现了，首先要介绍的是 Sample4_11_Activity 类，该类的代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_11/src/com/bn/chap4/intent 目录下的 Sample4_11_Activity.class。

```
package com.bn.chap4.intent;
import android.app.Activity;                                    //导入相关包
import android.content.Intent;                                  //导入相关包
import android.os.Bundle;                                       //导入相关包
import android.view.View;                                       //导入相关包
import android.view.View.OnClickListener;                       //导入相关包
import android.widget.Button;
public class Sample4_11_Activity extends Activity {
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);                            //设置当前界面
      Button bOK=(Button)this.findViewById(R.id.Button01); //创建 Button 按钮对象
      bOK.setOnClickListener(                                   //添加监听器
        new OnClickListener(){
                public void onClick(View v) {
                    Intent intent=new Intent();                 //创建 Intent 对象
                    intent.setClass(Sample4_11_Activity.this,
AnotherActivity.class);
                    startActivity(intent);                      //启动 Activity
                    Sample4_11_Activity.this.finish(); //关闭当前 Activity
}});}}
```

其中：

- 第 1～12 行为相关包的导入，设置 main.xml 为当前界面，并创建 Button 按钮的对象。
- 第 13～20 行为该界面中“确定”按钮的监听器，在监听器内首先创建 Intent 对象，使用 setClass()方法设定需要启动的 Activity 类，然后使用 startActivity()方法启动 Activity，再然后需要使用 finish()方法结束当前的 Activity。

最后需要介绍的是 AnotherActivity 类的实现，该类的代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_11/src/com/bn/chap4/intent 目录下的 AnotherActivity.class。

```
package com.bn.chap4.intent;
import android.app.Activity;                                    //导入相关包
```

```
3   import android.content.Intent;                                    //导入相关包
4   import android.os.Bundle;                                        //导入相关包
5   import android.view.View;                                        //导入相关包
6   import android.view.View.OnClickListener;                        //导入相关包
7   import android.widget.Button;                                    //导入相关包
8   public class AnotherActivity extends Activity{
9       public void onCreate(Bundle savedInstanceState){
10          super.onCreate(savedInstanceState);
11          setContentView(R.layout.anthor);                         //设置当前界面
12          Button bBack=(Button)this.findViewById(R.id.Button02);
13          bBack.setOnClickListener(                          //为 Button 按钮添加监听器
14          new OnClickListener(){
15                  public void onClick(View v) {
16                      Intent intent=new Intent();                  //创建 Intent 对象
17
intent.setClass(AnotherActivity.this,Sample4_11_Activity.class);
18                      startActivity(intent);                       //启动 Activity
19                      AnotherActivity.this.finish();               //关闭当前 Activity
20  }});}}
```

其中：

- 第 1～12 行为相关包的导入，设置 main.xml 为当前界面，并创建 Button 按钮的对象。
- 第 13～20 行为该界面中返回按钮的监听器，在监听器内首先创建 Intent 对象，使用 setClass()方法设定需要启动的 Activity 类，然后使用 startActivity()方法启动 Activity，最后需要使用 finish()方法结束当前的 Activity。

实例 12　界面间的数据传送技巧

应用程序在界面之间进行切换时，经常需要考虑的是界面之间的数据传输，在 Android 平台下可以通过 Bundle 实现界面之间数据的传输。在本节将对其进行详细介绍。

【实例描述】

该软件设计为填写个人信息，将个人信息发送到另一个界面中。在填写个人信息界面中，需要填写个人的姓名、爱好以及特长。填写完毕后单击“查看”按钮，则跳转到另一个界面，在该界面给出所填写的信息，单击界面中的返回键，返回至个人信息界面。本实例的运行效果图，如图 4-12 所示。

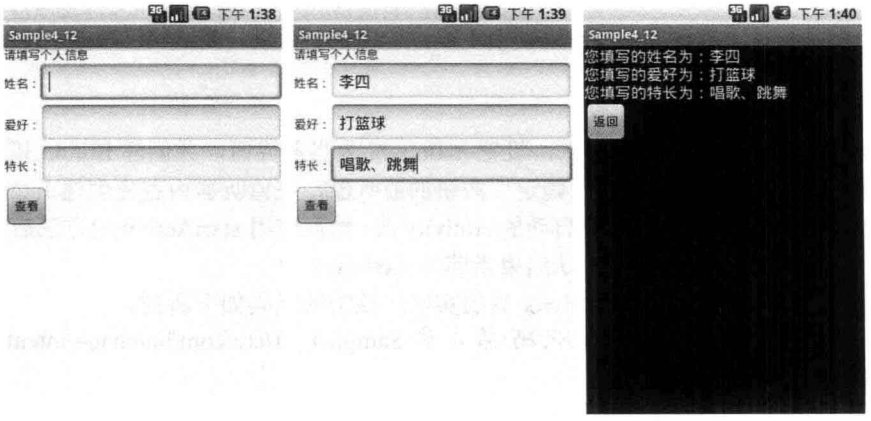


图 4-12　信息的填写和信息的传输

提示：在图 4-12 中依次为程序主界面，在主界面中填写个人信息，单击“查看”按钮，可以查看自己输入的有关自己的信息。

【实现过程】

使用 Intent 实现两个 Activity 间的数据通信首先需要创建 Itent 对象，随后使用 setClass()方法设定当前的 Activity 类和需要启动的 Activity 类，创建 Bundle 对象，将要传输的数据存放到 Bundle 对象中，最后将 Bundle 对象绑定到 Intent 对象上，使用 startActivity()方法启动另一个 Activity。

想要成功启动另一个 Activity，除了上述步骤外，还需要在 AndroidManifest.xml 添加 Activity 的声明，代码为<activity android:name="AnotherActivity"></activity>。

【代码解析】

在本部分将对其核心代码进行详细介绍，首先介绍的是 AndroidManifest.xml 文件的实现，在该文件内需要声明 AnotherActivity，该文件代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_12 目录下的 AndroidManifest.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3       package="com.bn.chap4.bundle"
4       android:versionCode="1"
5       android:versionName="1.0">
6      <application android:icon="@drawable/icon" android:label="@string/app_name">
7         <activity android:name=".Sample4_12_Activity"
8                 android:label="@string/app_name">
9            <intent-filter>
10              <action android:name="android.intent.action.MAIN" />
11              <category android:name="android.intent.category.LAUNCHER" />
12           </intent-filter>
13        </activity>
14           <activity android:name="AnotherActivity"></activity>
15     </application>
16  </manifest>
```

提示：上述代码中第 14 行为 AnotherActivity 在 AndroidManifest.xml 文件中的声明，若没有该行代码，程序运行时会产生异常。

接下来要介绍的是 main.xml 文件的实现，该文件代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_12/res/layout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="vertical"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent"
6      android:background="#ffffcc">                          <!--LinearLayout-->
7      <TextView
8      android:layout_width="fill_parent"
9      android:layout_height="wrap_content"
10     android:text="请填写个人信息"
11     android:textColor="#222222"
12     />                                                     <!--TextView 控件-->
13     <LinearLayout
```

```
14      android:id="@+id/LinearLayout01"
15      android:layout_width="fill_parent"
16      android:layout_height="wrap_content"
17      android:orientation="horizontal">                    <!--LinearLayout-->
18      <TextView
19              android:layout_width="wrap_content"
20              android:layout_height="wrap_content"
21              android:text="姓名："
22              android:textColor="#222222"
23         />                                                <!--TextView 控件-->
24      <EditText
25          android:id="@+id/EditText01"
26          android:layout_width="fill_parent"
27          android:layout_height="wrap_content">
28      </EditText>                                          <!--EditText 控件-->
29     </LinearLayout>
30     <LinearLayout
31      android:id="@+id/LinearLayout01"
32      android:layout_width="fill_parent"
33      android:layout_height="wrap_content"
34      android:orientation="horizontal">
35      <TextView
36              android:layout_width="wrap_content"
37              android:layout_height="wrap_content"
38              android:text="爱好："
39              android:textColor="#222222"
40         />                                                <!--TextView 控件-->
41      <EditText
42          android:id="@+id/EditText02"
43          android:layout_width="fill_parent"
44          android:layout_height="wrap_content">
45      </EditText>                                          <!--EditText 控件-->
46     </LinearLayout>
47     <LinearLayout
48      android:id="@+id/LinearLayout01"
49      android:layout_width="fill_parent"
50      android:layout_height="wrap_content"
51      android:orientation="horizontal">                    <!--LinearLayout-->
52      <TextView
53              android:layout_width="wrap_content"
54              android:layout_height="wrap_content"
55              android:text="特长："
56              android:textColor="#222222"
57         />                                                <!--TextView 控件-->
58      <EditText
59          android:id="@+id/EditText03"
60          android:layout_width="fill_parent"
61          android:layout_height="wrap_content">            <!--TextView 控件-->
62      </EditText>                                          <!--EditText 控件-->
63     </LinearLayout>
64      <Button
65          android:text="查看"
66          android:id="@+id/mainButton"
67          android:layout_width="wrap_content"
68          android:layout_height="wrap_content">
69      </Button>                                            <!--Button 控件-->
70  </LinearLayout>
```

其中：

- 第 1～12 行为最外层 LinearLayout 的属性设置和 TextView 的属性设置。

- 第 13～63 行为内层 LinearLayout 的属性设置，以及在其中的 TextView 和 EditText 的属性设置。
- 第 64～69 行为 Button 按钮的属性设置。

上面介绍的是 Sample4_12_Activity 类中界面 main.xml 文件的实现，下面要介绍的是 AnotherActivity 类中界面 anthor.xml 文件的实现，该文件的代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_12/res/layout 目录下的 anthor.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="vertical"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent"
6      >                                                    <!--LinearLayout-->
7      <TextView
8          android:layout_width="fill_parent"
9          android:layout_height="wrap_content"
10         android:textSize="18dip"
11         android:textColor="#ffffff"
12         android:id="@+id/TextView01"
13     />                                                   <!--TextView 控件-->
14     <Button                                              >
15          android:text="返回"
16          android:id="@+id/anthorButton"
17          android:layout_width="wrap_content"
18          android:layout_height="wrap_content">
19     </Button>                                            <!--Button 控件-->
20  </LinearLayout>
```

提示：上述代码中主要完成的是在 LinearLayout 中添加 TextView 和 Button。

经过界面布局的介绍，接下来要介绍的是 Sample4_12_Activity 和 AnotherActivity 类的设计与实现，首先介绍的是 Sample4_12_Activity 类，该类的代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_12/src/com/bn/chap4/bundle 目录下的 Sample4_12_Activity.class。

```
1   package com.bn.chap4.bundle;
2   import android.app.Activity;                                    //导入相关包
3   import android.content.Intent;
4   import android.os.Bundle;
5   import android.view.View;
6   import android.view.View.OnClickListener;
7   import android.widget.Button;
8   import android.widget.EditText;
9   public class Sample4_12_Activity extends Activity {
10     public void onCreate(Bundle savedInstanceState) {
11        super.onCreate(savedInstanceState);
12        setContentView(R.layout.main);                           //设置主界面
13        final EditText xmEt=(EditText)this.findViewById(R.id.EditText01);
                                                                   //姓名
14        final EditText ahEt=(EditText)this.findViewById(R.id.EditText02);
                                                                   //爱好
15        final EditText tcEt=(EditText)this.findViewById(R.id.EditText03);
                                                                   //特长
16        Button bOK=(Button)this.findViewById(R.id.mainButton); //查看按钮
17        bOK.setOnClickListener(                                  //查看按钮监听器
18          new OnClickListener(){
19                  public void onClick(View v) {
20                      String xm=xmEt.getText().toString().trim(); //获取姓名
```

```
21                          String ah=ahEt.getText().toString().trim();//获取爱好
22                          String tc=tcEt.getText().toString().trim();//获取特长
23                          Intent intent=new Intent();          //创建 Intent 对象
24                          intent.setClass(Sample4_12_Activity.this, Another
Activity.class);
25                          Bundle bundle=new Bundle();          //创建 Bundle 对象
26                          bundle.putString("xm", xm);          //加入信息
27                          bundle.putString("ah", ah);
28                          bundle.putString("tc", tc);
29                          intent.putExtras(bundle);            //绑定信息
30                          startActivity(intent);               //启动 Activity
31                          Sample4_12_Activity.this.finish(); //关闭该 Activity
32  }});}}
```

其中：

- 第 1～16 行为声明相关包的导入，设置 main.xml 为当前界面，并创建姓名、爱好、特长编辑框的对象和“查看”按钮的对象。
- 第 17～32 行为设置“查看”按钮监听器，在监听器内首先获取所填写的姓名、爱好和特长，将获取的信息添加到 Bundle 对象中，随后将 Bundle 对象与 Intent 对象绑定，启动 AnotherActivity 将数据传输到该类。

接下来要介绍的是 AnotherActivity 类的设计与实现，该类的源代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_12/src/com/bn/chap4/bundle 目录下的 AnotherActivity.class。

```
1   package com.bn.chap4.bundle;
2   import android.app.Activity;                                  //导入相关包
3   import android.content.Intent;
4   import android.os.Bundle;
5   import android.view.View;
6   import android.view.View.OnClickListener;
7   import android.widget.Button;
8   import android.widget.TextView;
9   public class AnotherActivity extends Activity{
10      public void onCreate(Bundle savedInstanceState){
11          super.onCreate(savedInstanceState);
12          setContentView(R.layout.anthor);                      //设置当前界面
13          TextView tv=(TextView)this.findViewById(R.id.TextView01);
                                                                  //显示的信息
14          Button button=(Button)this.findViewById(R.id.anthorButton);
                                                                  //返回按钮
15          Bundle bundle=this.getIntent().getExtras();           //获取数据对象
16          String xm=bundle.getString("xm");                     //获取姓名信息
17          String ah=bundle.getString("ah");                     //获取爱好信息
18          String tc=bundle.getString("tc");                     //获取特长信息
19          StringBuilder sb=new StringBuilder();                 //创建对象
20          sb.append("您填写的姓名为："+xm+"\n");
21          sb.append("您填写的爱好为："+ah+"\n");
22          sb.append("您填写的特长为："+tc+"\n");                 //组装字符串
23          tv.setText(sb.toString().trim());
24          button.setOnClickListener(                            //Button 按钮监听器
25              new OnClickListener(){
26                  public void onClick(View v) {
27                      Intent intent=new Intent();               //创建对象
28                      intent.setClass(AnotherActivity.this,
Sample4_12_Activity.class);
29                      startActivity(intent);                    //启动 Activity
```

```
30                          AnotherActivity.this.finish();      //关闭 Activity
31   }});}}
```

其中：

- 第 1～23 行为导入相关包，设置 main.xml 为当前界面，通过 Bundle 对象获取从 Sample4_12_Activity 传送来的数据，并将数据重新组装，显示在屏幕上。
- 第 24～31 行为返回按钮的监听器，在监听器内创建 Intent 对象，用 setClass()方法设定需要启动的 Activity 类，然后使用 startActivity()方法启动 Activity，在最后需要使用 finish()方法结束当前的 Activity。

实例 13　实现数据的返回接收

在应用程序中，界面之间的切换往往伴随着数据间的相互传输，在 Android 平台上，不仅仅可以通过 Bundle 实现数据的回传，也可以使用 startActivityForResult 实现上述功能。

【实例描述】

该软件与前一个相似，但实现的方法有所不同，在主界面填写个人信息，当填写完毕后单击“查看”按钮，进入另一个 Activity 界面，在该界面显示的是上一个界面所填写的个人信息，当单击“返回”按钮时，重新返回至主界面，在主界面不需要再次填写个人信息。本实例的运行效果图，如图 4-13 所示。

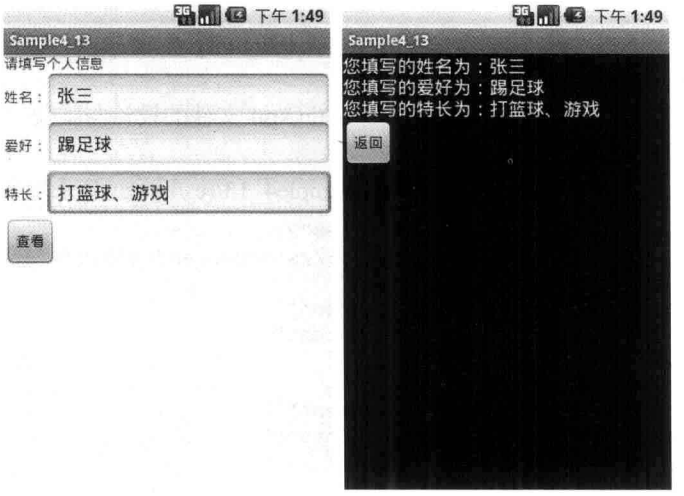


图 4-13　信息的填写和返回

提示：在图 4-13 中依次为填写个人信息界面效果图，单击“确定”按钮，可以查看自己填写的信息。

【实现过程】

在该软件中主要运用的技术为 startActivityForResult()方法，startActivityForResult 与 startActivity 的不同之处在于，后者仅仅是跳转到目标页面，若是想跳回当前页面，则必须再使用一次 startActivity。

而通过使用 startActivityForResult 则可以一次性地完成这项任务，当程序执行到这段代码的时候，界面会跳转到下一个 Activity，而当这个 Activity 被关闭以后(this.finish())，程序会自动跳转回第一个 Activity，并调用前一个 Activity 的 onActivityResult 方法。

【代码解析】

经过上面的介绍，相信读者对此已有所了解，在本部分要介绍的是该软件的核心代码部分，首先要介绍的是 AndroidManifest.xml 文件，该文件的代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_12 目录下的 AndroidManifest.xml。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.bn.chap4.sjfh"
    android:versionCode="1"
    android:versionName="1.0">
   <application android:icon="@drawable/icon" android:label="@string/app_name">
      <activity android:name=".Sample4_13_Activity"
              android:label="@string/app_name">
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>
        <activity android:name="AnotherActivity"></activity>
   </application>
</manifest>
```

提示：该文件主要完成的任务为 AnotherActivity 类的声明。

接下来要介绍的是主界面的布局设置——main.xml 文件的设计与实现，该文件的代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_13/res/layout 目录下的 main.xml。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:background="#ffffcc">                          <!--LinearLayout-->
   <TextView
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="请填写个人信息"
   android:textColor="#222222"
   />                                                     <!--TextView 控件-->
   <LinearLayout
    android:id="@+id/LinearLayout01"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
    <TextView
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:text="姓名："
           android:textColor="#222222"
       />                                                 <!--TextView 控件-->
   <EditText
        android:id="@+id/EditText01"
```

```
26          android:layout_width="fill_parent"
27          android:layout_height="wrap_content">
28       </EditText>                                    <!--EditText 控件-->
29     </LinearLayout>
30     <LinearLayout
31       android:id="@+id/LinearLayout01"
32       android:layout_width="fill_parent"
33       android:layout_height="wrap_content"
34       android:orientation="horizontal">
35       <TextView
36              android:layout_width="wrap_content"
37              android:layout_height="wrap_content"
38              android:text="爱好："
39              android:textColor="#222222"
40          />                                          <!--TextView 控件-->
41       <EditText
42           android:id="@+id/EditText02"
43           android:layout_width="fill_parent"
44           android:layout_height="wrap_content">
45       </EditText>                                    <!--EditText 控件-->
46     </LinearLayout>
47     <LinearLayout
48       android:id="@+id/LinearLayout01"
49       android:layout_width="fill_parent"
50       android:layout_height="wrap_content"
51       android:orientation="horizontal">              <!--LinearLayout-->
52       <TextView
53              android:layout_width="wrap_content"
54              android:layout_height="wrap_content"
55              android:text="特长："
56              android:textColor="#222222"
57          />                                          <!--TextView 控件-->
58       <EditText
59           android:id="@+id/EditText03"
60           android:layout_width="fill_parent"
61           android:layout_height="wrap_content">
62       </EditText>                                    <!--EditText 控件-->
63     </LinearLayout>
64       <Button
65           android:text="查看"
66           android:id="@+id/mainButton"
67           android:layout_width="wrap_content"
68           android:layout_height="wrap_content">
69       </Button>                                      <!--Button 控件-->
70   </LinearLayout>
```

其中：

- 第 1～12 行为最外层 LinearLayout 的属性设置和 TextView 的属性设置。
- 第 13～63 行为内层 LinearLayout 的属性设置，以及在其中的 TextView 和 EditText 的属性设置。
- 第 64～69 行为 Button 按钮的属性设置。

随后要介绍的是 anthor.xml 文件，该文件的代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_12/res/layout 目录下的 anthor.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="vertical"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent"
```

```
    >                                                    <!--LinearLayout-->
      <TextView
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:textSize="18dip"
         android:textColor="#ffffff"
         android:id="@+id/TextView01"
    />                                                   <!--EditText 的属性设置-->
      <Button
          android:text="返回"
          android:id="@+id/anthorButton"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content">
      </Button>                                          <!--Button 的属性设置-->
</LinearLayout>
```

提示：上述代码中主要完成的是在 LinearLayout 中添加 TextView 和 Button。

接下来要介绍的是 Sample4_13_Activity 类的设计与实现，该类的代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_13/src/com/bn/chap4/sjfh 目录下的 Sample4_13_Activity.class。

```
package com.bn.chap4.sjfh;
import android.app.Activity;                              //导入相关包
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
public class Sample4_13_Activity extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);                    //设置当前界面
        final EditText xmEt=(EditText)this.findViewById(R.id.EditText01);
                                                          //姓名
        final EditText ahEt=(EditText)this.findViewById(R.id.EditText02);
                                                          //爱好
        final EditText tcEt=(EditText)this.findViewById(R.id.EditText03);
                                                          //特长
        Button bOK=(Button)this.findViewById(R.id.mainButton);     //查看按钮
        bOK.setOnClickListener(                           //添加监听器
          new OnClickListener(){
                  public void onClick(View v) {
                      String xm=xmEt.getText().toString().trim();//获取姓名
                      String ah=ahEt.getText().toString().trim();//获取爱好
                      String tc=tcEt.getText().toString().trim();//获取特长
                      Intent intent=new Intent();                 //创建对象
                      intent.setClass(Sample4_13_Activity.this,
AnotherActivity.class);
                      Bundle bundle=new Bundle();     //创建 Bundle 对象
                      bundle.putString("xm", xm);     //添加姓名信息
                      bundle.putString("ah", ah);     //添加爱好信息
                      bundle.putString("tc", tc);     //添加特长信息
                      intent.putExtras(bundle);       //绑定信息
                      startActivityForResult(intent,0);       //启动 Activity
                  }});}
    public void onActivityResult(int requestCode,int resultCode,Intent data){
```

```
33      switch(resultCode){
34          case RESULT_OK:                                          //若为 RESULT_OK
35              Bundle bundle=data.getExtras();                      //创建 Bundle 对象
36              String xm=bundle.getString("xm");                    //获取姓名
37              String ah=bundle.getString("ah");                    //获取爱好
38              String tc=bundle.getString("tc");                    //获取特长
39              EditText xmEt=(EditText)this.findViewById(R.id.EditText01);
                                                                     //姓名
40               EditText ahEt=(EditText)this.findViewById(R.id.EditText02);
                                                                     //爱好
41               EditText tcEt=(EditText)this.findViewById(R.id.EditText03);
                                                                     //特长
42               xmEt.setText(xm);                                   //设置文本框文字
43               ahEt.setText(ah);
44               tcEt.setText(tc);
45          break;
46  }}}
```

其中：

- 第 1～16 行为相关包的导入，将 main.xml 文件设置为当前界面以及对象的创建。
- 第 17～31 行为“查看”按钮的监听器，在监听器内获取随后填写的信息，添加到 Bundle 对象中，并使用 Intent 对象绑定信息，最后使用 startActivityForResult()方法启动 Activity。
- 第 32～46 行为重写 onActivityResult()方法，在该方法中判断所收到的 resultCode 参数是否为 RESULT_OK，若满足则创建 Bundle 对象，将读取的信息设置在界面中的文本编辑框中。

最后要介绍的是 AnotherActivity 类的设计与实现，该类的代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_13/src/com/bn/chap4/sjfh 目录下的 AnotherActivity.class。

```
1   package com.bn.chap4.sjfh;
2   import android.app.Activity;                                  //导入相关包
3   import android.os.Bundle;
4   import android.view.View;
5   import android.view.View.OnClickListener;
6   import android.widget.Button;
7   import android.widget.TextView;
8   public class AnotherActivity extends Activity{
9       public void onCreate(Bundle savedInstanceState){
10          super.onCreate(savedInstanceState);
11          setContentView(R.layout.anthor);                      //设置当前界面
12          TextView tv=(TextView)this.findViewById(R.id.TextView01);
                                                                  //显示的信息
13          Button button=(Button)this.findViewById(R.id.anthorButton);
                                                                  //返回按钮
14          Bundle bundle=this.getIntent().getExtras();//创建 Bundle
15          String xm=bundle.getString("xm");                     //读取信息
16          String ah=bundle.getString("ah");
17          String tc=bundle.getString("tc");
18          StringBuilder sb=new StringBuilder();                 //创建 StringBuilder 对象
19          sb.append("您填写的姓名为："+xm+"\n");                 //组装字符串
20          sb.append("您填写的爱好为："+ah+"\n");
21          sb.append("您填写的特长为："+tc+"\n");
22          tv.setText(sb.toString().trim());                     //设置 TextView 文本
23          button.setOnClickListener(                            //返回按钮监听器
24              new OnClickListener(){
25                  public void onClick(View v) {
```

```
    26                      AnotherActivity.this.setResult(RESULT_OK,
AnotherActivity.this.getIntent());
    27                      AnotherActivity.this.finish();      //关闭 Activity
    28  }});}}
```

其中：

- 第 1～22 行为导入相关包，设置 main.xml 为当前界面，通过 Bundle 对象获取从 Sample4_13_Activity 传送来的数据，并将数据重新组装，显示在屏幕上。
- 第 24～28 行为返回按钮的监听器，在监听器内调用 setResult()方法，最后关闭当前 Activity。

实例 14　设置自己的手机显示模式

手机中包含多种显示风格，Android 手机平台为开发人员提供了 style 开发模式，并将 style 的维护工作交给视觉设计人员。在本节将对其进行介绍。

【实例描述】

该软件设置为两种显示模式，一种被设计为红色，另一种被设计为蓝色。本实例的运行效果图，如图 4-14 所示。

图 4-14　手机显示模式的设置

提示：在图 4-14 中依次为将显示字体设置为红色，之后的图片为将显示的字体设置为蓝色。

【实现过程】

在本软件中主要使用了 setTheme 的方法来制定手机当前的模式。

相关代码主要在 main.xml 中进行 TextView 的属性设置，然后在/res/values 目录下对 style.xml 进行显示模式设置。

【代码解析】

首先要介绍的是 main.xml 文件的实现，该文件的代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_14/res/layout 目录下的 main.xml。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   >                                              <!--LinearLayout-->
<TextView
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="@string/hello"
   android:textColor="#969a4f"/>                  <!--TextView 的属性设置-->
</LinearLayout>
```

提示：上述代码主要完成的是 TextView 的属性设置。

下面要介绍的是 style.xml 文件的设计与实现，该文件代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_14/res/values 目录下的 style.xml。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="Theme" parent="android:Theme"></style>
    <style name="Theme.Color_blue">                    <!--Theme 主题名称-->
        <item name="android:windowBackground">
        @drawable/background_blue</item>
        <item name="android:windowNoTitle">false</item>
        <item name="android:colorForeground">#ffffcc</item>
        <item name="android:colorBackground">#ff0000</item>
        <item name="android:textColor">#0000ff</item>
    </style>
    <style name="Theme.Color_red">                     <!--Theme 主题名称-->
        <item name="android:windowBackground">
        @drawable/background_red</item>
        <item name="android:windowNoTitle">false</item>
        <item name="android:colorForeground">#ea5aab</item>
        <item name="android:colorBackground">#FFFF00</item>
        <item name="android:textColor">#ff0000</item>
    </style>
</resources>
```

提示：上述代码中提供了两种模式，一种为 title 为红色，另一种为蓝色。

在这里还需要介绍一下 color.xml 的实现，在该文件主要完成的是颜色的设置，代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_14/res/values 目录下的 color.xml。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <drawable name="background_blue">#0000FF</drawable>
  <drawable name="background_red">#ff0000</drawable>
</resources>
```

提示：上述代码分别设置了蓝色和红色。

最后需要介绍的是 Sample4_14_Activity 类的设计与实现，该类的代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_14/src/com/bn/chap4/theme 目录下的 Sample4_14_Activity.class。

```
1   package com.bn.chap4.theme;
2   import android.app.Activity;                          //导入相关包
3   import android.os.Bundle;
4   public class Sample4_14_Activity extends Activity {
5      public void onCreate(Bundle savedInstanceState) {
6         super.onCreate(savedInstanceState);
7    //      setTheme(R.style.Theme_Color_red);           //设置手机模式为红色
8         setTheme(R.style.Theme_Color_blue);           //设置手机模式为蓝色
9         setContentView(R.layout.main);                //设置当前界面
10     }}
```

提示：上述代码中主要完成的是模式的设置和当前界面的设置。

实例 15　更改手机屏幕显示方向

Android 手机的屏幕可以通过设置更改其显示方向，尤其是在观看影片时，横屏是一个不错的选择，在本节将对上述功能的实现进行介绍。

【实例描述】

该软件设计得十分简单，在主界面包含两个按钮，若手机处于横屏模式，单击“竖屏”按钮时，手机将被设置为竖屏，若手机处于竖屏模式，单击“横屏”按钮时，手机将被设置为横屏。本实例的运行效果图，如图 4-15 所示。

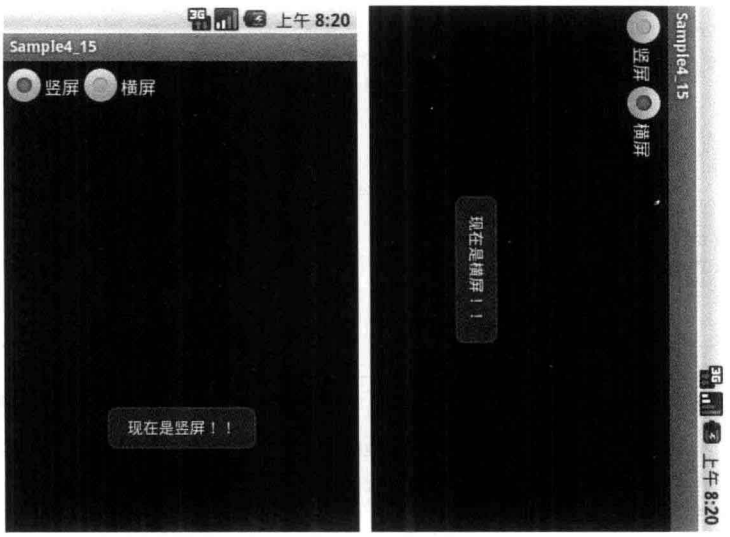


图 4-15　运行效果图

提示：在图 4-15 中依次为竖屏效果图以及横屏效果图。

【实现过程】

通过程序更改手机屏幕的显示方向需要使用 setRequestedOrientation()方法，获取手机当前屏幕的显示方向是通过 getRequestedOrientation()方法实现的，其中在使用 getRequestedOrientation()方法时需要传递 SCREEN_ORIENTATION_LANDSCAPE 和 SCREEN_ORIENTATION_PORTRAIT 两个参数。

SCREEN_ORIENTATION_LANDSCAPE 表示横屏，SCREEN_ORIENTATION_PORTRAIT 表示竖屏。

【代码解析】

在本部分要介绍的是核心代码，首先要介绍的是 main.xm 文件，该文件的代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_15/res/layout 目录下的 main.xml。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   >                                              <!--LinearLayout-->
    <RadioGroup
        android:id="@+id/RadioGroup01"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <RadioButton
            android:text="竖屏"
            android:id="@+id/RadioButton01"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content">
        </RadioButton>                            <!--RadioGroup 的属性设置-->
        <RadioButton
            android:text="横屏"
            android:id="@+id/RadioButton02"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content">
        </RadioButton>                            <!--RadioGroup 的属性设置-->
    </RadioGroup>
</LinearLayout>
```

提示：上述代码为在 LinearLayout 中添加 RadioButton 组，然后在 RadioGroup 中添加两个 RadioButton 按钮，其中一个为竖屏按钮，另一个为横屏按钮。

下面要介绍的是 Sample4_15_Activity 类的设计与实现，该类的代码如下所列。

代码位置：见随书光盘中源代码/第 4 章/Sample4_15/src/com/bn/chap4/changefx 目录下的 Sample4_15_Activity.class。

```
package com.bn.chap4.changefx;
import android.app.Activity;                          //导入相关包
import android.content.pm.ActivityInfo;
import android.os.Bundle;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.Toast;                          //导入相关包
public class Sample4_15_Activity extends Activity {
    public void onCreate(Bundle savedInstanceState) {
```

```
10          super.onCreate(savedInstanceState);
11          setContentView(R.layout.main);                          //设置当前界面
12          RadioGroup rg=(RadioGroup)this.findViewById(R.id.RadioGroup01);
13          final RadioButton rbH=(RadioButton)this.findViewById(R.id.
RadioButton01);//横屏按钮
14          final RadioButton rbV=(RadioButton)this.findViewById(R.id.
RadioButton02);//竖屏按钮
15          if(getRequestedOrientation()==-1){                      //若为-1，表示无法判断
16            Toast.makeText(                                       //创建 Toast
17                    Sample4_15_Activity.this,
18                    "无法区分横竖屏!! ",
19                    Toast.LENGTH_SHORT).show();                   //显示 Toast
20          }
21          RadioGroup.OnCheckedChangeListener mChange=             //创建对象
22            new RadioGroup.OnCheckedChangeListener() {
23            public void onCheckedChanged(RadioGroup group, int checkedId){
24                if(checkedId==rbH.getId()){                       //横屏按钮
25                    if(getRequestedOrientation()==-1){            //若为-1，表示无法判断
26                    Toast.makeText(                               //创建 Toast
27                            Sample4_15_Activity.this,
28                            "无法区分横竖屏!! ",
29                            Toast.LENGTH_SHORT).show();           //显示 Toast
30                  }else
31                    if(getRequestedOrientation()==
32                    ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE){//若为横屏
33                        setRequestedOrientation(
34                            ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
                                                                  //设置为竖屏
35                        Toast.makeText(                           //创建 Toast
36                            Sample4_15_Activity.this,
37                            "现在是竖屏!! ",
38                            Toast.LENGTH_SHORT).show();           //显示 Toast
39                    }
40                }else if(checkedId==rbV.getId()){                 //竖屏按钮
41                    if(getRequestedOrientation()==-1){
42                    Toast.makeText(                               //创建 Toast
43                            Sample4_15_Activity.this,
44                            "无法区分横竖屏!! ",
45                            Toast.LENGTH_SHORT).show();           //显示 Toast
46                  }else
47                    if(getRequestedOrientation()==
48                        ActivityInfo.SCREEN_ORIENTATION_PORTRAIT){ //若为竖屏
49                        setRequestedOrientation(
50                            ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
                                                                  //设置为横屏
51                        Toast.makeText(                           //创建 Toast
52                            Sample4_15_Activity.this,
53                            "现在是横屏!! ",
54                            Toast.LENGTH_SHORT).show();           //显示 Toast
55                    }}}};
56        rg.setOnCheckedChangeListener(mChange);                   //RadioGroup添加监听
57      }}
```

其中：

- 第 1～7 行为该类中相关包的引入。
- 第 8～20 行主要完成的是将 main.xml 设置为当前界面，创建 RadioGroup 和 RadioButton 对象，并使用 getRequestedOrientation()方法判断当前屏幕显示的方向。

- 第 21～57 行为创建 OnCheckedChangeListener 对象，为 RadioGroup 添加监听器，在监听器内判断，若单击的是“竖屏”按钮，则若当前屏幕方向为横屏，将屏幕设置为竖屏；反之，若单击的是“横屏”按钮，则若当前屏幕方向是竖屏，将屏幕的方向设置为横屏显示。

小结

本章中主要介绍 Android 手机用户界面的开发，首先介绍了如何获取屏幕的分辨率及界面的相应事件，然后介绍了几种菜单的实现，利用用户界面的功能实现，用户就可以方便地开发出功能各异的 Android 应用程序。

第 5 章　手机通信服务及手机控制

在本章将要详细介绍手机通信服务及其手机控制，手机的通信服务及手机的控制功能可以说是非常强大，如何巧妙地运用这些功能将是本章介绍的重点。

实例 1　自动调用系统的拨号、上网和发送 E-mail 的功能

在本节将要详细介绍如何调用系统的拨号、上网和发送 E-mail 的功能，以及 Linkify 的使用。

【实例描述】

手机拥有拨号、上网以及发送 E-mail 的功能，自定义一个文本编辑框调用手机的这些功能，充分体验 Android 手机的开放性。

在手机的应用程序中判断所输入的字符串做指定的业务逻辑，比如说，在文本编辑框中输入一个电话号码，借助此文本编辑框来实现拨打电话的功能，或者想要给亲戚朋友发送 E-mail，也可以通过在文本编辑框中输入 E-mail 地址完成该功能，当然，上网也是可以轻松完成的。

本实例的运行效果图，如图 5-1 所示。

图 5-1　程序运行效果图

提示：在图 5-1 中依次为在 EditText 中输入电话号码，单击蓝色字体即可调用手机的拨号系统（如图 5-1 右图所示），以及在 EditText 中输入网址，单击蓝色字体即可进入网站浏览。

【实现过程】

在 Android 手机中，将通过介绍 Linkify 来完成上述需求，Linkify 可以让系统动态获取 EditText 中的信息，并且根据所获取的信息做出相应的判断。在 TextView 上显示所输入的信息，然后实现 EditText 的 setOnKeyListener()方法完成既定目标。

【代码解析】

在本部分会详细介绍该软件的实现过程。首先要介绍的是该软件中 xml 文件的设置，其代码如下所示。

代码位置：见光盘源代码/第 5 章/Sample5_1/ res/layout/main.xml。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"                    <!-设定控件摆放顺序为竖直摆放>
   android:layout_width="fill_parent"
   android:layout_height="fill_parent">              <!--LinearLayout-->
    <TextView
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:text="请您输入电话号码或者网址或者 E-mail: "
    />                                               <!--TextView 控件-->
    <EditText
         android:id="@+id/EditText01"
         android:layout_width="200dip"
         android:layout_height="wrap_content">
    </EditText>                                      <!--EditText 控件-->
    <TextView
         android:id="@+id/TextView01"
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
    />                                               <!--TextView 控件-->
</LinearLayout>
```

提示：上述代码首先设定各控件在 LinearLayout 中的排放顺序，然后依次放置 TextView 和 EditText，分别为其设定宽、高以及显示的文本和 id。

在 xml 文件中搭建界面完成后，随后要做的就是在 Sample5_1_Activity 类中编写应用程序的代码，如下所列。

代码位置：见光盘源代码/第 5 章/Sample5_1/src/com/bn/chap5/tl/Sample5_1_Activity.class。

```
package com.bn.chap5.tl;
……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码
import android.widget.TextView;
public class Sample5_1_Activity extends Activity {
    TextView tv;                                          //声明 TextView 引用
    EditText et;                                          //声明 EditText 引用
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);                  //将 main.xml 设置为当前界面
      tv=(TextView)this.findViewById(R.id.TextView01);  //获取 TextView
      et=(EditText)this.findViewById(R.id.EditText01);  //获取 EditText
      et.setOnKeyListener(                            //为 EditText 添加监听器
            new OnKeyListener(){
                public boolean onKey(View v, int keyCode, KeyEvent event) {
                    tv.setText(et.getText());         //设置 TextView 显示文本
```

```
16                      Linkify.addLinks(
17                              tv,
18                              Linkify.WEB_URLS          //上网功能
19                              |Linkify.EMAIL_ADDRESSES  //发送 E-mail 功能
20                              |Linkify.PHONE_NUMBERS    //拨号功能
21                      );
22                      return false;
23  }});}}
```

提示：上述代码首先获取 TextView 和 EditText 的引用，然后为 EditText 设定监听器，在监听器中设置 Linkify，使其能够完成上网、拨号、发送 E-mail 的功能。

实例 2　电话拨号软件

拨打电话是再平常不过的事情了，拨打电话在 Android 中是如何实现的，在本节将会详细介绍 Intent 在电话拨号时的应用。

【实例描述】

“打电话”功能是每个手机所具有的最基本的功能，在本小软件中，是以 EditText 为输入电话号码的编辑框，当单击“确定”按钮后，实现拨打电话的功能。当然，为了避免用户输入电话号码格式错误，在单击“确定”按钮时，判断用户所输入的是否为正确格式的电话号码，若不正确则提示用户电话号码输入格式错误，并将 EditText 中的文本设置为空字符创建。

本实例的运行效果图，如图 5-2 所示。

图 5-2　输入电话号码图（左），按下“确定”按钮后（中），提示电话号码格式不正确（右）

提示：在图 5-2 中依次表示 EditText 输入的为电话号码，单击“确定”按钮开始拨号，并检测到电话号码格式不正确，使用 Toast 进行提示的效果。

【实现过程】

拨打电话功能的实现首先要做的是在 AndroidMainifest.xml 添加 uses-permission，其代码为：android:name="android.permission.CALL_PHONE"，详见光盘源代码/第 5 章/Sample5_2/AndroidMainifest.xml，然后需要通过自定义 Intent 对象，带入 action.CALL，通过 Uri.parse()方法将用户输入的电话号码带入，最后使用 startActivity()方法（将自定义的 Intent 传入），即可完成拨打电话的功能。

【代码解析】

权限设置后要做的是软件界面的搭建，即 main.xml 文件的设置，其代码如下所示。

代码位置：见光盘源代码/第 5 章/Sample5_2/ res/layout/main.xml。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   >                                                    <!--LinearLayout-->
    <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:te    xt="请您输入电话号码："
   />                                                   <!--TextView 控件-->
   <EditText
    android:id="@+id/EditText01"
    android:layout_width="200dip"
    android:layout_height="wrap_content">
   </EditText>                                          <!--EditText 控件-->
    <Button
     android:text="确定"
     android:id="@+id/Button01"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content">
    </Button>                                           <!--Button 控件-->
</LinearLayout>                                         <!--LinearLayout-->
```

提示：上述代码首先设定各控件在 LinearLayout 中的摆放顺序，然后依次放置 TextView、EditText 和 Button，分别为其设定宽、高以及显示的文本和 id。

最后要介绍的是 Sample5_2_Activity 类的实现，其代码如下所示。

代码位置：见光盘源代码/第 5 章/Sample5_2/src/com/bn/chap5/call/Sample5_2_Activity.class。

```
package com.bn.chap5.call;
……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
public class Sample5_2_Activity extends Activity {
    private EditText et;                                //声明 EditText 引用
    private Button bOk;                                 //声明 Button 引用
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      et=(EditText)this.findViewById(R.id.EditText01);  //创建对象
      bOk=(Button)this.findViewById(R.id.Button01);     //创建对象
      bOk.setOnClickListener(
            new OnClickListener(){
                public void onClick(View v) {
```

```
14                          String number=et.getText().toString().trim();
                                                        //获取输入的手机号码
15                          boolean flag=phoneNumber(number);
16                          if(flag){
17                              Intent intent=//创建对象，运行 action.CALL 的常数与通过
Uri 将字符创建传入
18                              new Intent("android.intent.action.CALL",Uri.parse
("tel:"+number));
19                              startActivity(intent);      //调用 startActivity 方法
20                              et.setText("");             //将 EditText 字符设为空
21                          }else{
22                              Toast.makeText(             //设定 Toast
23                                  Sample5_2_Activity.this,
24                                  "您输入的电话号码格式不正确",
25                                  Toast.LENGTH_SHORT ).show();
26                              et.setText("");             //将 EditText 字符设为空
27  }}});}
28      public boolean phoneNumber(String number){
29        boolean flag=false;
30        String pare="\\d{11}";                            //11 个整数的手机号码正则式
31        String pare2="\\d{12}";                           //12 个整数的座机号码正则式
32        CharSequence num=number;                          //获取电话号码
33        Pattern pattern=Pattern.compile(pare);            //判断是否为手机号码
34        Matcher matcher=pattern.matcher(num);
35        Pattern pattern2=Pattern.compile(pare2);          //判断是否为座机号码
36        Matcher matcher2=pattern2.matcher(num);
37        if(matcher.matches()||matcher2.matches()){        //如果符合格式
38            flag=true;                                    //标志位设为 true
39        }
40        return flag;                                      //返回标志位
41  }}
```

- 第 4～5 行为声明 EditText 和 Button 控件的引用。
- 第 7～27 行为设置 main.xml 为当前界面，并创建 EditText 和 Button 对象，然后为 Button 对象添加监听器，在按钮单击事件中，首先获取 EditText 中输入的文本，然后判断其是否符合电话号码的规则，若符合则创建 Intent 对象，并向系统发送该 intent 开始拨号，最后将 EditText 字符设为空。
- 第 28～41 行为判断所输入的电话号码是否符合规则的方法，在该方法中判断 11 位和 12 位电话号码，若符合电话号码格式则返回 true，否则返回 false。

实例 3　自制电话拨号系统

Android 手机自带一套拨号系统，在 Android 中是允许对其进行更改的，在本节将要为读者介绍如何替换系统的拨号系统，使用自定义的个性拨号系统。

【实例描述】

对于没有键盘的手机，每次拨打电话总会使用系统自带的拨号按钮，若是觉得手机的拨号按钮不好看，那么也可以自己制作一个个性的拨号系统。为每一个按钮添加监听器，当单击按钮时，在 EditText 中显示所单击的数字，最后按拨号键即可完成电话拨号功能。

本实例的运行效果图，如图 5-3 所示。

图 5-3　电话拨号器运行效果图

提示：在图 5-3 中依次表示的是按下手机上拨号键后选择是使用系统拨号还是自定义拨号，选择自定义拨号系统的界面，以及模拟成功播出电话的界面。

【实现过程】

与上一节一样，需要首先为拨打电话功能设置权限，即在 AndroidMainifest.xml 添加 uses-permission，其代码为：android:name="android.permission.CALL_PHONE"，其次是为按钮添加监听器，在 EditText 中显示所单击的数字。

最后同样需要自定义 Intent 对象，带入 action.CALL，通过 Uri.parse()方法将用户输入的电话号码带入，然后使用 startActivity()方法（将自定义的 Intent 传入），完成拨号功能。

【代码解析】

首先要介绍的是 main.xlm，在 main.xml 中完成的是拨号系统按钮属性的设置，其代码如下所列。

代码位置：见光盘源代码/第 5 章/Sample5_3/ res/layout/main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="vertical"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent"
6      >                                          <!--LinearLayout-->
7     <LinearLayout
8       android:id="@+id/LinearLayout06"
9       android:orientation="horizontal"
10      android:layout_width="fill_parent"
11      android:layout_height="wrap_content">      <!--LinearLayout-->
12        <EditText
13          android:text="@string/default_number"
14          android:id="@+id/EditText01"
15          android:layout_width="260dip"
16          android:textSize="24dip"
17          android:editable="false"
18          android:enabled="false"
```

```
19          android:singleLine="true"
20          android:background="@color/white"
21          android:textColor="@color/black"
22          android:layout_marginRight="6dip"
23          android:layout_marginLeft="10dip"
24          android:layout_height="wrap_content">
25        </EditText>                                    <!--EditText 控件-->
26        <Button
27           android:text=" "
28           android:id="@+id/Button_del"
29           android:textSize="24dip"
30           android:layout_width="wrap_content"
31           android:layout_height="wrap_content"
32           android:background="@drawable/myselector_del">
33        </Button>                                      <!--Button 控件-->
34    </LinearLayout>                                    <!--LinearLayout-->
35    <LinearLayout
36     android:id="@+id/LinearLayout01"
37     android:orientation="vertical"
38     android:layout_width="fill_parent"
39     android:layout_height="wrap_content">            <!--LinearLayout-->
40     <LinearLayout
41       android:id="@+id/LinearLayout02"
42       android:orientation="horizontal"
43       android:gravity="center_horizontal"
44       android:layout_width="fill_parent"
45       android:layout_height="wrap_content">
46          <Button
47           android:text="1"
48           android:id="@+id/Button01"
49           android:textSize="54dip"
50           android:textStyle="bold"
51           android:typeface="serif"
52           android:layout_width="wrap_content"
53           android:layout_height="wrap_content"
54           android:background="@drawable/myselector_num">
55          </Button>                                    <!--Button 控件-->
56          ……//该处省略了部分类似代码，读者可自行查看随书光盘中源代码。
57     </LinearLayout>
58      ……//该处省略了部分类似代码，读者可自行查看随书光盘中源代码。
59  </LinearLayout>                                     <!--LinearLayout-->
```

- 第 2～6 行为设置界面中控件总体的摆放顺序。
- 第 7～34 行为设置一个 LinearLayout 中控件摆放顺序为水平摆放，然后依次放置一个 EditText 和一个 Button，并为这两个控件设置属性。
- 第 35～57 行为再次设置 LinearLayout，并将控件摆放顺序设置为水平，然后放置 1～33 个 Button 按钮，并设置 Button 按钮属性。

控件摆放好后，下一步要做的就是为各个控件添加监听器，使其能够完成拨号功能，该功能实现的代码如下所列。

代码位置：见光盘源代码/第 5 章/Sample5_3/src/com/bn/chap5/selfcall/Sample5_3_Activity.class。

```
1   package com.bn.chap5.selfcall;
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3   public class Sample5_3_Activity extends Activity {
4       int[] numButtonIds= {                          //数字按钮的 id 数组
5       R.id.Button00,R.id.Button01,R.id.Button02,
6       R.id.Button03,R.id.Button04,R.id.Button05,
7       R.id.Button06,R.id.Button07,R.id.Button08,
8       R.id.Button09
```

```
9       };
10      public void onCreate(Bundle savedInstanceState) {
11          super.onCreate(savedInstanceState);
12          setContentView(R.layout.main);                    //设置主界面
13          Button bDel=(Button)this.findViewById(R.id.Button_del);
                                                              //为删除按钮添加监听器
14          bDel.setOnClickListener(
15                  new View.OnClickListener()  {
16                   public void onClick(View v) {
17                   EditText et=(EditText)findViewById(R.id.EditText01);
                                                              //获取 EditText 对象
18                   String num=et.getText().toString();
19                   num=(num.length()>1)?num.substring(0,num.length()-1):"";
20                   et.setText(num);                         //设置 EditText 显示文字
21  } });
22          Button bDial=(Button)this.findViewById(R.id.Button_dial);
23          bDial.setOnClickListener(                         //为拨号按钮添加监听器
24                  new View.OnClickListener()  {
25                   public void onClick(View v){
26                   EditText et=(EditText)findViewById(R.id.EditText01);
                                                              //获取输入的电话号码
27                   String num=et.getText().toString();
28                   Intent dial = new Intent();              //创建 Intent 拨号
29                      dial.setAction("android.intent.action.CALL");
30                      dial.setData(Uri.parse("tel://"+num));
31                      startActivity(dial);                  //调用 startActivity 方法
32   }});
33          Button bCancel=(Button)this.findViewById(R.id.Button_cancel);
34          bCancel.setOnClickListener(                       //为退出按钮添加监听器
35                  new View.OnClickListener(){
36                   public void onClick(View v){
37                   Sample5_3_Activity.this.finish();        //退出程序
38  }});
39          View.OnClickListener numListener=new  View.OnClickListener(){
                                                              //为 0～9 数字按钮创建监听器
40            public void onClick(View v){
41              Button tempb=(Button)v;
42              EditText et=(EditText)findViewById(R.id.EditText01);
43              et.append(tempb.getText());                   //添加数字
44  }};
45
46          for(int id:numButtonIds){                         //为所有数字按钮添加监听器
47            Button tempb=(Button)this.findViewById(id);
48            tempb.setOnClickListener(numListener);
49  }}}
```

- 第 4～9 行为声明 Button 数组。
- 第 10～38 行主要为删除按钮、拨号按钮以及退出按钮添加监听器，当按下“删除”按钮时，删除一个数字，按下“拨号”按钮时，创建 Intent 对象，并向系统发送该 intent 开始拨号，按下“退出”按钮，则退出该程序。
- 第 39～49 行主要为设置 1～9 按钮的监听器，每次按下数字键后，在 EditText 中添加相应的数字。

实例 4　手机发送短信

发送短信功能也是手机的一个重要环节，在本节将要详细介绍如何自定义一个短信编辑系

统，然后使用 sendTextMessage()方法发送已编辑的短信。

【实例描述】

手机除了拥有打电话的功能外，还有另外一个十分重要的功能，即发送短信的功能，当然在 Android 平台下，也能够自己设计一个个性的短信发送软件。

该软件自定义两个 EditText 用来获取用户填写的目标号码和短信内容，单击“调用系统程序发送短信”按钮后，首先检验所输入的电话号码是否符合格式要求，符合格式要求后向目标用户发送短信，短信发送成功后，提示用户短信发送成功，若电话号码格式不正确，则提示用户电话号码不符合格式。

本实例的运行效果图，如图 5-4 所示。

图 5-4　软件运行效果图

提示：在图 5-4 中依次为填写目标号码以及短信内容，单击按钮判断目标号码格式正确后弹出 Toast 提示发送成功，以及检测到目标号码格式不正确，弹出 Toast 提示电话号码不正确的界面。

【实现过程】

本软件通过 SmsManager 对象的 sendTestMessage()方法来实现发送短信的功能，在该方法中的 5 个参数分别为收件人、发送人、正文、发送服务、送达服务，其中收件人和正文不可为空。

发送短信功能的实现同样需要增加权限，其权限代码为：<uses-permission android:name="android.permission.SEND_SMS"/>，详见光盘源代码/第 5 章/Sample5_4/AndroidMainifest.xml。

【代码解析】

下面要介绍的是该软件的界面 main.xml，该文件的代码如下所列。

代码位置：见光盘源代码/第 5 章/Sample5_4/ res/layout/main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3       android:orientation="vertical"
```

```
4       android:layout_width="fill_parent"
5       android:layout_height="fill_parent"
6       android:background="@drawable/bg"
7       android:gravity="bottom"
8       >                                              <!--LinearLayout-->
9       <TextView
10        android:text="@string/tel"
11        android:id="@+id/TextView02"
12        android:textSize="20dip"
13        android:textStyle="bold"
14        android:textColor="@color/black"
15        android:layout_width="wrap_content"
16        android:layout_height="wrap_content"
17        android:paddingLeft="5dip">
18      </TextView>                                    <!--TextView 控件-->
19      <EditText
20        android:text="@string/telno"
21        android:id="@+id/EditText02"
22        android:layout_width="fill_parent"
23        android:layout_height="wrap_content">
24      </EditText>                                    <!--EditText 控件-->
25      <TextView
26        android:text="@string/sms"
27        android:id="@+id/TextView01"
28        android:layout_width="wrap_content"
29        android:textSize="20dip"
30        android:textStyle="bold"
31        android:textColor="@color/black"
32        android:paddingLeft="5dip"
33        android:layout_height="wrap_content">
34      </TextView>                                    <!--TextView 控件-->
35      <EditText
36        android:text="@string/smsnr"
37        android:id="@+id/EditText01"
38        android:layout_width="fill_parent"
39        android:singleLine="false"
40        android:gravity="top|left"
41        android:layout_height="100dip">
42      </EditText>                                    <!--EditText 控件-->
43      <Button
44        android:text="@string/dial"
45        android:id="@+id/Button01"
46        android:textSize="20dip"
47        android:layout_width="fill_parent"
48        android:layout_height="wrap_content">
49      </Button>                                      <!--Button 控件-->
50  </LinearLayout>                                    <!--LinearLayout-->
```

提示：上述代码主要完成的是发送短信软件界面的搭建，第 9～23 行为设置目标号码控件，第 25～42 行为设置短信内容控件，第 43～49 行为 Button 按钮的设置。

控件搭建完成后，下一步要做的就是为各个空间添加监听，使其能够完成既定任务，其代码如下。

代码位置：见光盘源代码/第 5 章/Sample5_4/src/com/bn/chap5/sms/Sample5_4_Activity.class。

```
1   package com.bn.chap5.sms;
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3   @SuppressWarnings("deprecation")
4   public class Sample5_4_Activity extends Activity {
```

```
5       public void onCreate(Bundle savedInstanceState) {
6          super.onCreate(savedInstanceState);
7          setContentView(R.layout.main);                          //设置当前界面
8          Button bdial=(Button)this.findViewById(R.id.Button01);//获取 Button 对象
9          bdial.setOnClickListener(                               //为拨号按钮添加监听器
10               new View.OnClickListener(){
11                public void onClick(View v){
12                EditText etTel=(EditText)findViewById(R.id.EditText02);
                                                                   //获取 EditText 对象
13                String telStr=etTel.getText().toString();  //获取输入的电话号码
14                EditText etSms=(EditText)findViewById(R.id.EditText01);
15                String smsStr=etSms.getText().toString();  //获取输入的短信内容
16                if(PhoneNumberUtils.isGlobalPhoneNumber(telStr))
                                                          //判断号码字符串是否合法
17                {                                       //合法则发送短信
18                    v.setEnabled(false);                //将发送按钮设置为不可用
19                    sendSMS(telStr,smsStr,v);           //发送短信的方法
20                }
21                else{                                   //不合法则提示
22                    Toast.makeText(
23                            Sample5_4_Activity.this, //上下文
24                            "电话号码不符合格式!!! ",     //提示内容
25                            5000                        //信息显示时间
26                            ).show();
27   }}});}
28     private void sendSMS(String telNo,String smsStr,View v){//直接发送短信的方法
29        PendingIntent pi=                                    //创建 PendingIntent 对象
30        PendingIntent.getActivity(this, 0, new Intent(this,Sample5_4_Activity.
class), 0);
31        SmsManager sms=SmsManager.getDefault();
32        sms.sendTextMessage
33            (
34                telNo,                                  //收件人
35                null,                                   //发送人
36                smsStr,                                 //短信内容
37                pi,                                     //发送服务
38                null                                    //送达服务
39            );
40        Toast.makeText(//短信发送成功给予提示
41                    Sample5_4_Activity.this,            //上下文
42                    "恭喜你，短信发送成功！",             //提示内容
43                    5000                                //信息显示时间
44                    ).show();
45        v.setEnabled(true);                             //发送按钮设置为可用
46   }}
```

其中：

- 第 6～26 行为获取 Button 对象，并为 Button 对象添加监听器，当单击“Button”按钮时，检测所输入的电话号码是否符合格式要求，若符合要求则发送短信，并提示发送成功，若不符合则提示电话号码不符合格式。
- 第 28～46 行为发送短信的方法，通过获取输入的收件人号码、短信内容和 View，然后调用 sendTextMessage()方法，向收件人发送短信。其中该方法中的 5 个参数分别为收件人、发送人、正文、发送服务、送达服务，其中收件人和正文不可为空。

实例 5　简易电子邮件

在本节将要详细介绍 Android 手机的发送电子邮件的功能。

【实例描述】

前面提到了手机的两个最基本的功能——打电话和发送短信，在这里将要介绍的是手机的另外一个功能——发送 E-mail。该软件通过利用 Android 强大的网络支持能力，向目标用户发送 E-mail。

在该软件的界面中需要填写的是收件人地址、发件人地址、主题以及邮件的内容，单击发送时，软件会自动检测收件人地址和发件人地址格式填写是否正确，若不正确则使用 Toast 提示用户填写错误，若填写正确，则正常发送 E-mail。

本实例的运行效果图，如图 5-5 所示。

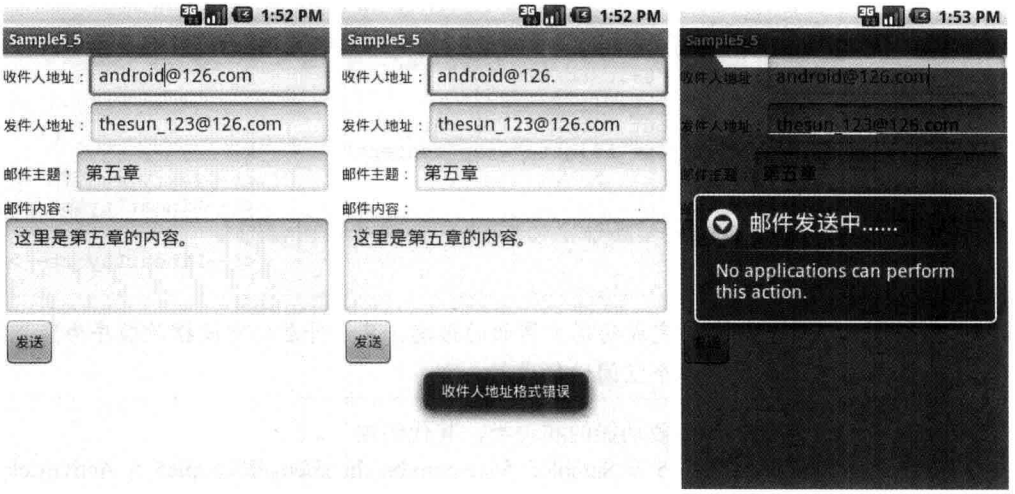


图 5-5　发送 E-mail 软件运行效果图

提示：在图 5-5 中依次为填写邮件的信息，单击“发送”按钮后检测到收件人地址格式错误，使用 Toast 进行提示；单击“发送”按钮，验证收件人和发件人地址格式正确后对邮件进行模拟发送。

【实现过程】

该软件通过自定义 Intent 对象，使用 Android.content.Intent.ACTION_SEND 的参数来实现通过手机发送 E-mail 的服务。在发送过程中需要检测所输入的 E-mail 地址是否符合格式要求，否则不可能发送成功，然而实际上，在手机发送或接收 E-mail 时，是通过 Android 内置的 Gmail 程序处理的。

由于模拟器中并没有内置的 Gmail Client 端程序，所以会出现如图 5-5 右图所示的效果。

【代码解析】

首先介绍的是 main.xml 文件，该文件代码如下所列。

代码位置：见光盘源代码/第 5 章/Sample5_5/ res/layout/main.xml。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:background="#ffffcc"
   >                                                    <!--LinearLayout-->
    <LinearLayout
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:orientation="horizontal">
        <TextView
         android:text="收件人地址："
         android:id="@+id/TextView01"
         android:textColor="#222222"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content">
        </TextView>                                     <!--TextView 控件-->
        <EditText
         android:text=""
         android:id="@+id/EditText01"
         android:textColor="#222222"
         android:layout_width="fill_parent"
         android:layout_height="wrap_content">
        </EditText>                                     <!--TextView 控件-->
    </LinearLayout>                                     <!--LinearLayout-->
    ……//该处省略了部分类似代码，读者可自行查看随书光盘中源代码。
</LinearLayout>                                         <!--LinearLayout-->
```

提示：上述代码主要完成的是主界面的搭建，其最外层的空间摆放顺序为竖直摆放，然后再按照要求对各个空间进行设置。

接下来要介绍的是该软件主要功能的实现类，其代码如下。

代码位置：见光盘源代码/第 5 章/Sample5_5/src/com/bn/chap5/email/Sample5_5_Activity.class。

```
package com.bn.chap5.email;
……//该出省略了部分类的导入，读者可自行查看随书光盘中源代码。
public class Sample5_5_Activity extends Activity {
    EditText etReceiver;                                    //收件人
    EditText etSender;                                      //发件人
    EditText etTheme;                                       //主题
    EditText etMessage;                                     //内容
    Button bSend;                                           //发送按钮
    String strReceiver;                                     //收件人信息
    String strSender;                                       //发件人信息
    String strTheme;                                        //主题信息
    String strMessage;                                      //内容信息
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      etReceiver=(EditText)this.findViewById(R.id.EditText01); //获取收件人对象
      etSender=(EditText)this.findViewById(R.id.EditText04);//获取发件人对象
      etTheme=(EditText)this.findViewById(R.id.EditText02); //获取主题对象
      etMessage=(EditText)this.findViewById(R.id.EditText03);//获取内容对象
      bSend=(Button)this.findViewById(R.id.Button01);       //发送按钮对象
      bSend.setOnClickListener(                             //添加监听器
```

```
22            new OnClickListener(){
23                public void onClick(View v) {
24                    strReceiver=etReceiver.getText().toString().trim();
                                                                    //获取收件人
25                    strSender=etSender.getText().toString().trim();
                                                                    //获取发件人
26                    strTheme=etTheme.getText().toString().trim();   //获取主题
27                    strMessage=etMessage.getText().toString().trim();
                                                                    //获取内容
28                    String  parent="^[a-zA-Z][\\w\\.-]*[a-zA-Z0-9]@[a-zA-Z0-9]
[\\w\\.-]*s[a-zA-Z0-9]"+
29                                "\\.[a-zA-Z][a-zA-Z\\.]*[a-zA-Z]$"; //匹配正则式
30                    if(!strReceiver.matches(parent)){ //查看收件人地址是否符合格式
31                        Toast.makeText(
32                            Sample5_5_Activity.this,
33                            "收件人地址格式错误",
34                            Toast.LENGTH_SHORT).show();
35                    }else if(!strSender.matches(parent)){
                                                    //查看发件人地址是否符合格式
36                        Toast.makeText(
37                            Sample5_5_Activity.this,
38                            "发件人地址格式错误",
39                            Toast.LENGTH_SHORT).show();
40                    }else{                          //若都符合格式，则发送邮件
41                        Intent intent=new Intent(android.content.Intent.ACTION_
SEND);//发送邮件功能
42                        intent.setType("plain/text");       //设置邮件格式
43                        intent.putExtra(android.content.Intent.EXTRA_EMAIL,
strReceiver);
44                        intent.putExtra(android.content.Intent.EXTRA_CC,
strSender);
45                        intent.putExtra(android.content.Intent.EXTRA_SUBJECT,
strTheme);
46                        intent.putExtra(android.content.Intent.EXTRA_TEXT,
strMessage);
47                        startActivity(Intent.createChooser(intent,
48                            getResources().getString(R.string.start)));
49   }}} );}}
```

- 第 14～20 为获取各控件的对象。
- 第 21～49 行是为 Button 添加监听器，当单击“Button”按钮时，首先判断获取的收件人地址是否符合 E-mail 格式要求，然后判断发件人的地址是否符合要求，若不符合要求则提示格式错误，若满足格式要求，则创建 Intent 对象，并将获取的信息放入 Intent 对象中，最后调用 startActicity()方法。

实例 6　自制手机通讯录搜索

在本节将要详细介绍如何对通讯录中联系人进行搜索，主要是对 ContentResolver 的应用。

【实例描述】

手机中的通讯录是一个不可或缺的部分，手机用户的所有联系人均在里面，本软件实现的就是通过自制的手机通讯录软件对联系人进行简单搜索。

在 EditText 中输入所要搜索联系人的名字的首字母，在自定义的 ContentResolver 中会显示出首字母相同的所有联系人，当输入的是“*”时，则会显示出所有的联系人。当单击其中一个

联系人时，会在主界面中显示该联系人的姓名和电话。若该用户没有电话，则会显示该联系人尚未有电话号码。

本实例的运行效果图，如图 5-6 所示。

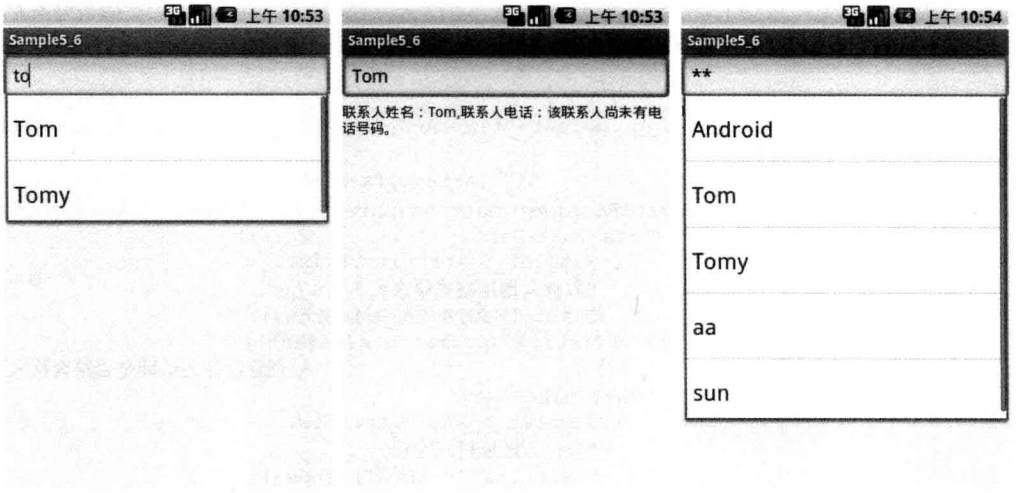


图 5-6　搜索运行效果图

提示：在图 5-6 中依次为搜索 to 开头的联系人，选中搜索结果中的某一项后的界面，该联系人的信息，以及搜索通讯录中所有联系人的界面。

【实现过程】

在本软件中，主要应用了 AutoCompleteView Widget，以及如何使用 ContentResolver 访问通讯录里的联系人，并将搜索到的联系人添加到 CursorAdapter 中，当单击其中一位联系人时，为 TextView 重新设置显示文本，将选中的联系人的姓名和电话显示在界面中。

需要注意的是，在进行软件测试之前需要在模拟器的通讯录中添加联系人，否则将无法正常测试。

【代码解析】

经过上面的理论介绍，接下来将要介绍的是该软件中的代码部分，首先要介绍的是在 AndroidManifest.xml 中权限的添加，代码如下。

代码位置：见光盘源代码/第 5 章/Sample5_6/AndroidMainifest.xml。

```
1    <?xml version="1.0" encoding="utf-8"?>
2    ……//该处省略了部分不重要代码，读者可自行查看随书光盘中源代码
3        </application><!--添加查看通讯录的权限-->
4         <uses-permission android:name="android.permission.READ_CONTACTS">
</uses-permission>
5    </manifest>
```

提示：上述代码中第 4 行为添加查看通讯录的权限。

下面要介绍实现界面搭建的 main.xml 文件，其代码如下。

代码位置：见光盘源代码/第 5 章/Sample5_6/ res/layout/main.xml。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:background="#ffffcc"
   >                                              <!--LinearLayout-->
    <AutoCompleteTextView
     android:id="@+id/AutoCompleteTextView01"
     android:layout_width="fill_parent"
     android:layout_height="wrap_content">
    </AutoCompleteTextView>
    <TextView
     android:id="@+id/TextView01"
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:textColor="#222222">
    </TextView>                                    <!--TextView 控件-->
</LinearLayout>                                    <!--LinearLayout-->
```

提示：上述代码中第 8～12 行为设置 AutoCompleteTextView，第 13～18 行为设置 TextView。

接下来要介绍的是该软件中 Activity 类的实现，其代码如下。

代码位置：见光盘源代码/第 5 章/Sample5_6/src/com/bn/chap5/txl/Sample5_6_Activity.class。

```
package com.bn.chap5.txl;
……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
public class Sample5_6_Activity extends Activity {
    private AutoCompleteTextView myAclt;                    //声明引用
    private TextView tv;
    private Cursor cursor;                                  //声明 Cursor 引用
    private ContactsAdapter myCa;
    static final String[] PEOPLE_PROJECTION=
    {                                                       //需要查询的通讯录字段
        Contacts.People._ID,
        Contacts.People.PRIMARY_PHONE_ID,
        Contacts.People.TYPE,
        Contacts.People.NUMBER,                             //手机号码
        Contacts.People.LABEL,
        Contacts.People.NAME                                //联系人姓名
    };
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);                        //设置当前界面
      myAclt=(AutoCompleteTextView)this.findViewById(R.id.AutoComplete
TextView01);//获取对象
      tv=(TextView)this.findViewById(R.id.TextView01);  //获取对象
      ContentResolver content=getContentResolver();//获取 ContentResolver 对象
      cursor=content.query(                             //取得通讯录的 Cursor
            Contacts.People.CONTENT_URI,
            PEOPLE_PROJECTION,
            null,
            null,
            Contacts.People.DEFAULT_SORT_ORDER
      );
      myCa=new ContactsAdapter(this,cursor);            //创建对象
```

```
31          myAclt.setAdapter(myCa);
32          myAclt.setOnItemClickListener(
33            new AdapterView.OnItemClickListener(){
34                    public void onItemClick(AdapterView<?> arg0, View arg1,
35                            int arg2, long arg3) {
36                        Cursor c=myCa.getCursor();          //取得 Cursor 对象
37                        c.moveToPosition(arg2);             //移动到单击位置
38                        String number=c.getString(          //获取电话号码
39                        c.getColumnIndexOrThrow(Contacts.People.NUMBER));
40                        if(number==null){                   //若号码为空
41                            number="该联系人尚未有电话号码";
42                        }
43                        String name=c.getString(c.getColumnIndexOrThrow
(Contacts.People.NAME));
44                        tv.setText("联系人姓名:"+name+",联系人电话:"+number+"。");
45  }});}}
```

- 第 4～16 行为声明各控件的引用，以及声明需要查询的通讯录字段名。
- 第 17～31 行为创建各控件对象，以及从通讯录中获取 Cursor，并创建 ContactsAdapter 类的对象。
- 第 32～45 行为为 ContactsAdapter 对象添加监听器，当单击事件发生时，首先取得 Cursor 对象，然后移动至单击位置，并获取该联系人的电话和姓名，最后将该联系人的电话和姓名设置到 TextView 中。

最后为读者介绍的是 ContactsAdapter 类的实现，该类代码如下。

代码位置：见光盘源代码/第 5 章/Sample5_6/src/com/bn/chap5/txl/ContactsAdapter.class。

```
1   package com.bn.chap5.txl;
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码
3   public class ContactsAdapter extends CursorAdapter{
4       ContentResolver myCr;                   //声明引用
5       public ContactsAdapter(Context context, Cursor c) {
6           super(context, c);
7           myCr=context.getContentResolver(); //通过上下文取得 ContentResolver
8       }
9       public void bindView(View arg0, Context arg1, Cursor arg2) {
10          ((TextView) arg0).setText(          //获取通讯录中的姓名
11          arg2.getString(arg2.getColumnIndexOrThrow(Contacts.People.NAME)));
12      }
13      @Override
14      public View newView(Context arg0, Cursor arg1, ViewGroup arg2) {
15          final LayoutInflater myLi=LayoutInflater.from(arg0);
                                                //声明 LayoutInflater 对象
16          final TextView tv=(TextView)myLi.inflate(
17                  android.R.layout.simple_dropdown_item_1line,
18                  arg2,
19                  false
20          );
21          tv.setText(                         //设置显示文字
22              arg1.getString(arg1.getColumnIndexOrThrow(Contacts.People.NAME))
23          );
24          return tv;
25      }
26      public String convertToString(Cursor cursor){   //获取联系人姓名
27          String  str=cursor.getString(cursor.getColumnIndexOrThrow(Contacts.
People.NAME));
28          return str;                                 //返回姓名字符串
29      }
30      public Cursor runQueryOnBackgroundThread(CharSequence cs){
```

```
31          if(getFilterQueryProvider()!=null){
32              return getFilterQueryProvider().runQuery(cs);
33          }
34          StringBuilder sb=new StringBuilder();          //声明 StringBuilder 对象
35          String[] str=null;
36          if(cs!=null){                                  //若 cs 为空
37              sb.append("UPPER(");                       //组装字符串
38              sb.append(Contacts.People.NAME);
39              sb.append(") GLOB ?");
40              str=new String[]{                          //创建 String 对象，并赋值
41              cs.toString().toUpperCase()+"*"
42              };
43          }
44          return myCr.query(                             //返回搜索结果
45                  Contacts.People.CONTENT_URI,
46                  Sample5_6_Activity.PEOPLE_PROJECTION,  //字段值
47                  sb==null?null:sb.toString(),
48                  str,
49                  Contacts.People.DEFAULT_SORT_ORDER
50  );}}
```

- 第 4～8 为该类的构造器，在构造器中获取 ContentResolver。
- 第 9～29 行为获取通讯录中联系人的姓名，并在 TextView 中设置。
- 第 30～50 行为获取搜索字符为*时的结果集，并将结果集返回。

实例 7　一键查询联系人资料

本节要介绍的是对通讯录里的联系人进行全部查询，并将单击的联系人资料用 Toast 显示出来。

【实例描述】

本软件构造十分简单，通过设计一个搜索按钮，来完成对整个通讯录的搜索，当单击搜索按钮时，则会进入手机的通讯录，在通讯录中可以查看联系人的资料，当单击通讯录中联系人名单时，重新回到软件界面，并使用 Toast 提示所选联系人的姓名和电话号码，若没有电话号码则将其号码设为 null。

本实例的运行效果图，如图 5-7 所示。

图 5-7　程序运行效果图

提示：在图 5-7 中依次表示的是软件运行后的界面，单击界面中的搜索，则进入通讯录界面（图 5-7 的中图），当单击联系人时，重新返回至主界面（图 5-7 的右图）。

【实现过程】

在该程序中首先声明一个静态的常量 COUNT，将其作为数据返回给主 Activity 的判断依据，然后自定义 Button 的 OnClick 事件，并使用 Uri.parse()方法将联系人的位置通过参数“content://contact/people”传入，最后通过使用 startActivityForResult()方法打开新的 Activity。

另外，还需要重写 onActivityResult()方法，在其中实现联系人数据的访问，通过 Cursor 对象访问联系人姓名和电话，最后返回至 Activity，并在 Toast 中显示。

在该软件中同样需要设置权限，权限为：<uses-permission android:name="android.permission.READ_CONTACTS"></uses-permission>，详见光盘源代码/第 5 章/Sample5_7/AndroidMainifest.xml。

【代码解析】

本部分将要详细介绍该软件中的核心代码，该类的代码如下所列。

代码位置：见光盘源代码/第 5 章/Sample5_7/src/com/bn/chap5/lxr/Sample5_7_Activity.class。

```
1   package com.bn.chap5.lxr;
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3   public class Sample5_7_Activity extends Activity {
4       private Button bSearch;                                    //搜索按钮
5       private static final int COUNT=0;                          //静态常量
6      public void onCreate(Bundle savedInstanceState) {
7         super.onCreate(savedInstanceState);
8         setContentView(R.layout.main);                           //设置主界面
9         bSearch=(Button)this.findViewById(R.id.Button01);        //按钮对象
10        bSearch.setOnClickListener(
11           new OnClickListener(){
12               public void onClick(View v) {
13                Uri uri=Uri.parse("content://contacts/people");
                                                  //创建 Uri 对象取得联系人的资源位置
14                Intent intent=new Intent(Intent.ACTION_PICK,uri);
                                                  //创建 Intent 对象，并将取得的值返回
15                startActivityForResult(intent,COUNT);
                                                  //打开新的 Activity，并获取返回值
16  }});}
17     protected void onActivityResult(int requestCode,int resultCode,Intent data){
18      switch(requestCode){
19          case 0:                                                //若为 0
20              final Uri uri=data.getData();
21              if(uri!=null){
22                  try{
23                   Cursor cursor=managedQuery(uri,null,null,null,null);
                                                  //获取联系人信息
24                   cursor.moveToFirst();        //移动指针到最前端
25                   String lxrName=cursor.getString(
26                   cursor.getColumnIndexOrThrow(Contacts.People.NAME));
                                                  //取得联系人姓名
27                   String lxrPhoneNumber=cursor.getString(
28                   cursor.getColumnIndexOrThrow(Contacts.People.NUMBER));
                                                  //取得联系人电话
```

```
29                      Toast.makeText(
30                          Sample5_7_Activity.this,
31                          "你所查询的联系人姓名为："+lxrName+",电话为："+lxrPhoneNumber,
32                          Toast.LENGTH_SHORT
33                          ).show();
34                      }catch(Exception e){e.printStackTrace();}   //打印异常
35                  }
36              break;
37          }
38          super.onActivityResult(requestCode, resultCode, data);   //调用父类方法
39      }}
```

- 第 4～16 行主要完成的是声明以及创建 Button 对象，声明静态常量作为判断数据返回给主 Activity 的判断依据，并重写 Button 的 OnClick()方法，当单击 Button 时创建 Uri 对象，并通过 Intent 对象将数据传回，最后打开新的 Activity。
- 第 17～39 行为重写 onActivityResult()方法，在该方法中若传入的数据为 0，则获取联系人信息，将指针移动到最前端，然后获取所选联系人的姓名和电话号码，并将该信息通过 Toast 显示出来。

实例 8　有图标的爱好选择系统

在本节要介绍的是 BaseAdapter 的应用，在适配器中添加合适的图片和文字，美化其外观的同时，增加可读性。

【实例描述】

该软件构造简单，模拟的爱好放入 Spinner 内，当单击下拉列表时，弹出爱好选项，选项中包含爱好的图标和文字，当单击其中一个爱好时，在界面的文字更改为所选爱好。

本实例的运行效果图，如图 5-8 所示。

图 5-8　系统运行效果图

提示：在图 5-8 中依次表示的是开始运行时的界面效果，单击下拉列表后显示的列表信息，以及选中列表中的某一项的界面效果。

【实现过程】

首先在界面中添加 TextView，并将其设置为显示当前 Spinner 中的内容，当单击 Spinner 时，弹出已设定好的爱好，每个爱好前面有相应的图标提示。然后为 Spinner 添加适配器 BaseAdapter，并为 BaseAdapter 添加监听器，当单击其中一个选项时，则更改 TextView 中的内容。

【代码解析】

在本部分将要详细介绍该软件的实现类，其代码如下。

代码位置：见光盘源代码/第 5 章/Sample5_8/src/com/bn/chap5/adp/Sample5_8_Activity.class。

```
1   package com.bn.chap5.adp;
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3   public class Sample5_8_Activity extends Activity {
4       final static int WRAP_CONTENT=-2;                    //表示 WRAP_CONTENT 的常量
5       //所有资源图片（足球、篮球、排球）id 的数组
6     int[] drawableIds={R.drawable.football,R.drawable.basketball,R.drawable.
volleyball};
7       //所有资源字符串（足球、篮球、排球）id 的数组
8       int[] msgIds={R.string.zq,R.string.lq,R.string.pq};
9       public void onCreate(Bundle savedInstanceState) {
10        super.onCreate(savedInstanceState);
11        setContentView(R.layout.main);                     //设置当前界面
12        Spinner sp=(Spinner)this.findViewById(R.id.Spinner01);//初始化 Spinner
13        BaseAdapter ba=new BaseAdapter(){                  //为 Spinner 准备内容适配器
14              public int getCount() {
15                  return 3;                                //总共三个选项
16              }
17              public Object getItem(int arg0) { return null; }
18              public long getItemId(int arg0) { return 0; }
19              public View getView(int arg0, View arg1, ViewGroup arg2) {
20                  LinearLayout ll=new LinearLayout(Sample5_8_Activity.this);
                                                           //初始化 LinearLayout
21                  ll.setOrientation(LinearLayout.HORIZONTAL);     //设置朝向
22                  ImageView  ii=new ImageView(Sample5_8_Activity.this);
                                                           //初始化 ImageView
23                  ii.setImageDrawable(getResources().getDrawable
(drawableIds[arg0]));//设置图片
24                  ll.addView(ii);                          //添加到 LinearLayout 中
25                  TextView tv=new TextView(Sample5_8_Activity.this);
                                                           //初始化 TextView
26                  tv.setText(" "+getResources().getText(msgIds[arg0]));
                                                           //设置内容
27                  tv.setTextSize(24);                      //设置字体大小
28                  tv.setTextColor(R.color.black);          //设置字体颜色
29                  ll.addView(tv);                          //添加到 LinearLayout 中
30                  return ll;
31              }};
32        sp.setAdapter(ba);                                 //为 Spinner 设置内容适配器
33        sp.setOnItemSelectedListener(                      //设置选项选中的监听器
34          new OnItemSelectedListener(){
35              public void onItemSelected(AdapterView<?> arg0, View arg1,
36                  int arg2, long arg3) {          //重写选项被选中事件的处理方法
37                  TextView tv=(TextView)findViewById(R.id.TextView01);
                                                           //获取主界面 TextView
38                  LinearLayout ll=(LinearLayout)arg1;
                                                  //获取当前选中选项对应的 LinearLayout
```

```
39                  TextView tvn=(TextView)ll.getChildAt(1);//获取其中的 TextView
40                  StringBuilder sb=new StringBuilder();
                                                    //用 StringBuilder 动态生成信息
41                  sb.append(getResources().getText(R.string.ys)); //组装字符串
42                  sb.append(":");
43                  sb.append(tvn.getText());
44                  tv.setText(sb.toString());      //信息设置进主界面 TextView
45              }
46              public void onNothingSelected(AdapterView<?> arg0) { }
47  });}}
```

其中：

- 第 4～8 行为声明图片资源和文字资源的数组，在数组内存放其对应的 id。
- 第 10～32 行为初始化 Spinner 以及为 Spinner 准备适配器，并在适配器内添加一个 LinearLayout，将 LinearLayout 中控件摆放顺序设定为水平排放，最后一次摆放 ImageView 和 TextView，用来显示图片和文字。
- 第 33～47 行为设置选项选中时的监听器，其主要实现的功能为将在 TextView 中显示所选中的文字。

实例 9　界面切换时的震动提醒

在本节要介绍的是手机震动效果的设置，在该程序中设定当界面进行成功转换时，手机震动提醒转换成功。

【实例描述】

在重要场合不适宜开启声音，那么震动就发挥其重要的作用了，可以说手机震动提醒是经常用到的功能，在本实例中，模拟登录系统，当登录成功时，界面进行切换，界面切换成功时，手机震动提醒界面切换完毕，用户可以进行下一步操作。

当然，手机的震动也可以设置在其他功能中，比如编写游戏时，赛车碰撞到障碍物进行短时间的震动，可以大大增加游戏的可玩性。

本实例的运行效果图，如图 5-9 所示。

图 5-9　本实例运行效果图

提示：在图 5-9 中依次为填写用户名和登录密码，单击“登录”按钮验证用户信息通过，切换界面，同时手机震动，由于在模拟器中无法实现真正的震动，在这里使用 Toast 进行提示，等震动 2 秒钟后，震动效果自动结束，并同样使用 Toast 进行提示。

【实现过程】

在登录界面中包含了本地验证，若用户名或密码填写错误会提示用户填写错误，然后需要让自己的手机在指定的时间进行震动提醒，需要创建 Vibrator 对象，通过调用 vibrate 方法来实现震动效果。

在 Vibrate 构造器中有四个参数，前三个值为设定震动的大小，将数值改成一大一小就可以感觉到震动的差异，最后一个值为震动的时间。当 repeat 值为 0 时表示震动会一直持续下去，当 repeat 值为–1 时，表示震动只会出现一轮，运行完毕后不会再有震动出现。

【代码解析】

经过上面的原理介绍，接下来要介绍的是该实例的实现类，该类代码如下所示。

代码位置：见光盘源代码/第 5 章/Sample5_9/src/com/bn/chap5/zd/Sample5_9_Activity.class。

```
1   package com.bn.chap5.zd;
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3   public class Sample5_9_Activity extends Activity {
4      Button myButton;                                    //声明 Button
5      EditText etUser;                                    //用户名
6      EditText etPwd;                                     //密码
7      String user;                                        //用户名字符串
8      String pwd;                                         //密码字符串
9      private Vibrator vb;                                //声明 Vibrate
10     Handler hd=new Handler(){                           //创建 Handler 对象
11      public void handleMessage(Message msg){
12          switch(msg.what){
13              case 0:                                    //若接收信息的编号为 0
14                  Toast.makeText(
15                          Sample5_9_Activity.this,
16                          "手机震动已经取消",      //显示的文本
17                          Toast.LENGTH_SHORT     //设定显示时间
18                          ).show();
19              break;
20          }}};
21     public void onCreate(Bundle savedInstanceState) {
22        super.onCreate(savedInstanceState);
23        setContentView(R.layout.main);                   //设定当前界面
24        vb=(Vibrator)getApplication().getSystemService(Service.VIBRATOR_SERVICE);
25        myButton=(Button)this.findViewById(R.id.Button01);     //获取 Button
26        etUser=(EditText)this.findViewById(R.id.EditText01);
27        etPwd=(EditText)this.findViewById(R.id.EditText02);
28        myButton.setOnClickListener(                     //为 Button 添加监听器
29          new OnClickListener(){
30                  public void onClick(View v) {       //重写 OnClick 方法
31                      user=etUser.getText().toString().trim();//获取用户名
32                    pwd=etPwd.getText().toString().trim();    //获取密码
33                      if(user.equals("123")&&pwd.equals("123")){
34                          gotoChange();                    //进入另一个界面
35                          vb.vibrate(new long[]{100,10,100,10000}, -1);
```

```
                                                          //设置手机震动
36                          Toast.makeText(
37                                  Sample5_9_Activity.this,
38                                  "手机在震动",
39                                  Toast.LENGTH_SHORT    //Toast 显示时间
40                                  ).show();
41                      }else{
42                          Toast.makeText(
43                                  Sample5_9_Activity.this,
44                                  "用户名或密码错误",      //显示的文本
45                                  Toast.LENGTH_SHORT    //Toast 显示时间
46                                  ).show();
47  }}}};}
48      public void gotoChange(){                         //定义另一个界面
49       setContentView(R.layout.change);                 //设置为当前界面
50       new Thread(){
51           public void run(){
52               try{Thread.sleep(2000);                  //使线程睡眠 2 秒钟
53               }catch(Exception e){e.printStackTrace();}
54               vb.cancel();                             //取消震动
55               hd.sendEmptyMessage(0);                  //发送消息启动 Toast
56  }}.start();                                           //启动线程
57  }}
```

- 第 21～47 行为获取引用对象，并为 Button 添加监听器，当单击登录按钮时，首先进行本地验证，若满足要求则进行界面的切换，同时设置 vibrate 方法，使手机进行震动。
- 第 48～56 行为设置 change 界面，首先设置 change 为当前界面，然后开启线程等待手机震动结束，最后取消震动。

实例 10　带图片的小提醒

在上一节介绍的 Toast 中，显示的仅仅是文字，而在 Toast 中是可以加入图片的，本节将介绍如何在 Toast 中加入图片。

【实例描述】

本实例基于 5.9 中的实例，在 Toast 中添加了图片，美化界面，达到图文可视化提醒。本实例的运行效果图，如图 5-10 所示。

图 5-10　本实例运行效果图

提示：在图 5-10 中依次为登录界面，在登录界面单击“登录”按钮验证用户信息符合，登录成功后，弹出带图片的 Toast 提示界面。

【实现过程】

在本实例中，将自定义一个 Layout 放入 Toast 中，使用于提示的 Toast 对象不仅仅局限于文字提示。在 Toast 中放入图片的步骤为，首先自定义 LinearLayout，然后将 ImageView 对象放入 LinearLayout 中，最后再将 LinearLayout 对象放入 Toast 中即可。

【代码解析】

接下来是该实例核心代码的讲解，其代码如下所示。

代码位置：见光盘源代码/第 5 章/Sample5_10/src/com/bn/chap5/toast/Sample5_10_Activity.class。

```
1   package com.bn.chap5.toast;
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3   public class Sample5_10_Activity extends Activity {
4      Button myButton;                                          //声明 Button
5      EditText etUser;                                          //用户名
6      EditText etPwd;                                           //密码
7      String user;                                              //用户名字符串
8      String pwd;                                               //密码字符串
9      private Vibrator vb;                                      //声明 Vibrator 引用
10     Handler hd=new Handler(){                                 //创建 Handler 对象
11      public void handleMessage(Message msg){
12          switch(msg.what){
13              case 0:
14                  Toast.makeText(                           //声明 Toast
15                          Sample5_10_Activity.this,
16                          "手机震动已经取消",
17                          Toast.LENGTH_SHORT                //Toast 显示时间
18                          ).show();
19              break;
20  }} };
21     public void onCreate(Bundle savedInstanceState) {
22        super.onCreate(savedInstanceState);
23        setContentView(R.layout.main);                         //设置当前界面
24
vb=(Vibrator)getApplication().getSystemService(Service.VIBRATOR_SERVICE);
25        myButton=(Button)this.findViewById(R.id.Button01);     //获取 Button
26        etUser=(EditText)this.findViewById(R.id.EditText01);
27        etPwd=(EditText)this.findViewById(R.id.EditText02);
28        myButton.setOnClickListener(                           //为 Button 创建监听器
29          new OnClickListener(){
30                  public void onClick(View v) {
31                      user=etUser.getText().toString().trim();//获取用户名
32                      pwd=etPwd.getText().toString().trim(); //获取密码
33                      LinearLayout ll=new LinearLayout(Sample5_10_Activity.
this);
34                      ImageView iv=new ImageView(Sample5_10_Activity.this);
                                                               //创建 ImageView 对象
35                      TextView tv=new TextView(Sample5_10_Activity.this);
                                                               //创建 TextView 对象
36                      tv.setText("手机在震动");
37                      if(user.equals("123")&&pwd.equals("123")){
```

```
38                          gotoChange();                    //进入另一个界面
39                          vb.vibrate(new long[]{100,10,100,10000}, -1);
40                          Toast toast=Toast.makeText(     //创建 Toast 对象
41                                  Sample5_10_Activity.this,
42                                  tv.getText().toString().trim(),
43                                  Toast.LENGTH_SHORT      //指定显示时间
44                                  );
45                          View textView=toast.getView();
46                          ll.setOrientation(LinearLayout.HORIZONTAL);
                                                            //设置摆放顺序
47                          iv.setImageResource(R.drawable.icon);  //设置图片
48                          ll.addView(iv);                  //将 ImagerView 放入
49                          ll.addView(textView);            //将 TextView 放入
50                          toast.setView(ll);     //将 LinearLayout 放入 Toast 中
51                          toast.show();                    //显示 Toast
52                      }else{
53                          Toast.makeText(
54                                  Sample5_10_Activity.this,
55                                  "用户名或密码错误",
56                                  Toast.LENGTH_SHORT      //设置时间
57                                  ).show();               //显示 Toast
58  }}});}
59      public void gotoChange(){
60      ……//该处省略了部分不重要代码，读者可自行查看随书光盘中源代码。
61  }}
```

- 第 4～20 行声明控件引用以及创建 Handler 对象。
- 第 22～27 行为获取 Button 和 EditText。
- 第 28～58 行为 Button 添加监听器，在监听器中首先本地验证，判断用户名和密码是否符合规定，若符合规定则创建 Toast 对象，并将创建的 ImageView 和 TextView 放入 Toast 中，最后将其显示出来，若不符合，则直接提示用户名或密码错误。

实例 11　音乐播放器在状态栏上图标提示

在 Android 手机界面的最上方有一条提供显示时间、信号强度、电量等信息的空间，这就是手机的状态栏。在本节将要介绍的是如何利用手机状态栏。

【实例描述】

本软件是一个简易的音乐播放器，每当打开音乐播放器后，就会在手机的状态栏显示一个音乐播放的图标，尤其是关闭前台后在后台运行音乐播放器时，这个图标成为音乐播放器正在运行的标志。当需要打开前台时，可以拉开状态栏，单击图标完成打开前台的工作。

提示：关于音乐播放器的情况将会在后面进行详细介绍。

本实例的运行效果图，如图 5-11 所示。

提示：在图 5-11 中，第一幅图为音乐播放器刚开始运行的效果，第二幅图为音乐播放器退出前台播放的效果图，可以在手机的状态栏上看到音乐播放器的提示图标，第三幅图为拉下手机的状态栏后的效果图，在该界面可以单击图标重新回到音乐播放器界面，还可以单击清除，清除状态栏中的图标提示。

图 5-11　音乐播放器运行效果图

【实现过程】

实现上述功能是通过使用 Notification 技术，当应用程序向 Android 系统发出一个 Notification 后，其会在状态栏显示小图标。

在本实例中，当指定的 Notification 发出 200 毫秒后震动 300 毫秒，当然也可以使用系统自带的一些动作，调用系统自带动作的代码为：myNotification.defaults=Notification.DEFAULT_VIBRATE。

系统提供的动作如表 5-1 所列。

表 5-1　系统默认动作

Notification 常量	对应的动作
Notification.DEFAULT_SOUND	发出系统默认的声音
Notification.DEFAULT_LIGHTS	屏幕发亮
Notification.DEFAULT_VIBRATE	手机震动
Notification.DEFAULT_ALL	包含以上三种情况

【代码解析】

经过上面的理论介绍，接下来要介绍该功能的实现类，其代码如下。

代码位置：见光盘源代码/第 5 章/Sample5_11/src/com/bn/chap5/no/Sample5_11_Activity.class。

```
1   package com.bn.chap5.no;
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3   public class Sample5_11_Activity extends Activity { //迷你播放器的前台 Activity
4      String currentPlay="/sdcard/zdan.mp3";           //播放路径
5      UIUpdateReceiver uiur;                           //界面更新 Intent 的接收者
6      public void onCreate(Bundle savedInstanceState) {
7         super.onCreate(savedInstanceState);
8         setContentView(R.layout.main);                //设置当前界面
9         ImageButton ibPlayPause=(ImageButton)findViewById(R.id.ImageButton
PlayPause); //获取引用
10        ImageButton ibStop=(ImageButton)findViewById(R.id.ImageButtonStop);
                                                        //获取停止按钮的引用
```

```
        uiur=new UIUpdateReceiver(this);          //创建界面更新 Intent 接收者
        ibStop.setOnClickListener(                //为停止按钮添加监听器
          new OnClickListener(){
                  public void onClick(View v) {//按下停止按钮后广播出去停止命令的
Intent
                      Intent intent=new Intent(Constant.MUSIC_CONTROL);
                      intent.putExtra("cmd", Constant.COMMAND_STOP);
                      Sample5_11_Activity.this.sendBroadcast(intent);
  }});
        ibPlayPause.setOnClickListener(           //为播放/暂停按钮添加监听器
              new OnClickListener(){
                  public void onClick(View v) {
                      if(uiur.status==Constant.STATUS_PLAY){//若当前为播放状态
则发出暂停命令
                          Intent intent=new Intent(Constant.MUSIC_CONTROL);
                          intent.putExtra("cmd", Constant.COMMAND_PAUSE);
                          Sample5_11_Activity.this.sendBroadcast(intent);
                          //若当前为停止状态则发出播放命令，同时发出播放路径
                      }else if(uiur.status==Constant.STATUS_STOP){
                          Intent intent=new Intent(Constant.MUSIC_CONTROL);
                          intent.putExtra("cmd", Constant.COMMAND_PLAY);
                          intent.putExtra("path", currentPlay);  //放入路径
                          Sample5_11_Activity.this.sendBroadcast(intent);
                      }else{                      //若当前为暂停则发出继续播放命令
                          Intent intent=new Intent(Constant.MUSIC_CONTROL);
                          intent.putExtra("cmd", Constant.COMMAND_PLAY);
                          Sample5_11_Activity.this.sendBroadcast(intent);
  }}});
        IntentFilter filter=new IntentFilter();
                          //动态注册接收状态更新 Intent 的 UIUpdateReceiver
          filter.addAction(Constant.UPDATE_STATUS);
          this.registerReceiver(uiur, filter);
          this.startService(new Intent(this,MyMusicPlayerService.class));
                          //发出启动后台 Service 的 Intent
    }
    public void onStart(){
     super.onStart();
     try{
         NotificationManager nm =
         (NotificationManager)getSystemService(NOTIFICATION_SERVICE);
                          //获取通知管理器
         Intent intent = new Intent(this,Sample5_11_Activity.class);
                          //创建 Intent
         PendingIntent pi = PendingIntent.getActivity(this,0, intent, 0);
                          //将 Intent 封装为 PendingIntent
         Notification myNotification = new Notification(); //创建 Notification
         myNotification.icon = R.drawable.notilogo; //设置 Notification 的图标
         myNotification.vibrate = new long[] { 200, 300 };
                          //200 毫秒后震动 300 毫秒
         //设置 Notification 的信息与单击发出的 PendingIntent
         myNotification.setLatestEventInfo(this, "音乐播放器运行中", "单击查看",
pi);
         nm.notify(0,myNotification);                  //发出 Notification
        }
        catch(Exception e){e.printStackTrace();}      //打印异常
    }
    public void onDestroy(){
     super.onDestroy();
     this.unregisterReceiver(uiur);                   //注销 Receiver
  }}
```

- 第 4～41 行为首先声明播放路径以及界面更新 Intent 的接收者，然后依次为停止、播放/暂停按钮添加监听器，并动态注册接收状态更新 Intent 的 UIUpdateReceiver，;//发出启动后台 Service 的 Intent。
- 第 42～57 为设置 Notification，首先是获取通知管理器，创建 Intent 对象后将其封装成 PendingIntent，然后创建 Notification 对象，设置显示的图标，添加系统默认的震动特效，最后发出 Notification。
- 第 58～61 行为重写 onDistroy()方法，在该方法中注销 Receiver。

实例 12　自制打开或关闭 Wi-Fi

在 Android 手机中可以设置 Wi-Fi，在无线控制目录下可以对其进行打开或关闭的操作，在本节将要介绍的是自制按钮实现 Wi-Fi 的打开或关闭。

【实例描述】

本实例通过定义一个 CheckBox 对 Wi-Fi 的开关进行设置，但单击 CheckBox 时，会自动检测当前 Wi-Fi 处于何种状态，若 Wi-Fi 正处于打开过程中或已经打开，假如想要打开 Wi-Fi，则系统提示 Wi-Fi 已经打开或正在打开过程中，再一次打开 Wi-Fi 失败。

若 Wi-Fi 处于关闭或正在关闭的过程中，假如想要关闭 Wi-Fi，则系统提示 Wi-Fi 正在关闭过程中或已经关闭，再一次关闭 Wi-Fi 失败。若处于系统不明状态，则提示再一次打开或关闭失败。

本实例的运行效果图，如图 5-12 所示。

图 5-12　运行效果图

提示：在运行效果图中，第一幅图为刚运行时的界面，此时检测到 Wi-Fi 处于未知状态，这是由于模拟器的 Wi-Fi 功能无法启动，第二幅图为打开 Wi-Fi 失败，第三幅图为关闭 Wi-Fi 失败。

【实现过程】

为了能够掌握 Wi-Fi 的当前状态，在每次打开或关闭 Wi-Fi 时均会检测当前 Wi-Fi 的状态，其状态常数如表 5-2 所列。

表 5-2　Wi-Fi 状态常量与其所对应的 Wi-Fi 状态

Wi-Fi 状态常量	表示的状态
WifiManager.WIFI_STATE_ENABLED	Wi-Fi 已经打开
WifiManager.WIFI_STATE_ENABLING	Wi-Fi 正在打开的过程中
WifiManager.WIFI_STATE_DISABLED	Wi-Fi 已经关闭
WifiManager.WIFI_STATE_DISABLING	Wi-Fi 正在关闭的过程中
WifiManager.WIFI_STATE_UNKNOWN	Wi-Fi 处于未知状态

根据以上 Wi-Fi 状态和 CheckBox 的开关就可以得出是否能够打开或关闭 Wi-Fi 了。

【代码解析】

在这里将要介绍的是核心代码部分，其代码如下所示。

代码位置：见光盘源代码/第 5 章/Sample5_12/src/com/bn/chap5/wifi/Sample5_12_Activity.class。

```
1   package com.bn.chap5.wifi;
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码
3   public class Sample5_12_Activity extends Activity {
4       ……//该处省略了部分不重要代码，读者可自行查看随书光盘源代码
5     public void onCreate(Bundle savedInstanceState) {
6       ……//该处省略了部分不重要代码，读者可自行查看随书光盘源代码
7       mWifi=(WifiManager)this.getSystemService(Context.WIFI_SERVICE);
                                                  //创建对象
8       if(mWifi.isWifiEnabled()){                //判断运行程序后，Wi-Fi 是否已被打开
9         if(mWifi.getWifiState()==WifiManager.WIFI_STATE_ENABLED){
                                                  //判断 Wi-Fi 是否已打开
10            cb.setChecked(true);                //若已经打开，则将 CheckBox 设为 true
11            cb.setText("关闭 WIFI");            //设置文字为关闭 WIFI
12        }else{
13            cb.setChecked(false);               //若没有打开，则将 CheckBox 设为 false
14            cb.setText("打开 WIFI");            //设置文字为打开 Wi-Fi
15        }
16      }else{
17        cb.setChecked(false);                   //若没有打开，则将 CheckBox 设为 false
18        cb.setText("打开 WIFI");                //设置文字为打开 Wi-Fi
19      }
20      cb.setOnClickListener(                    //判断 CheckBox 的单击事件
21            new OnClickListener(){
22                public void onClick(View v) {
23        if(cb.isChecked()==false){              //当选项为取消状态时----false
24            try{                                //尝试关闭 Wi-Fi
25            if(mWifi.isWifiEnabled()){          //若 Wi-Fi 状态为已打开
26                        if(mWifi.setWifiEnabled(false)){   //关闭 Wi-Fi
27                        tv.setText("已经成功关闭 Wi-Fi");//设置 TextView 的文字
28                        }else{tv.setText("关闭 Wi-Fi 失败");
                                                  //设置 TextView 的文字}
29            }else{                              //若 Wi-Fi 状态不为打开时
30                switch(mWifi.getWifiState()){
31                case WifiManager.WIFI_STATE_ENABLING:
                                                  //Wi-Fi 在启动过程中，将无法关闭
32                        tv.setText("Wi-Fi 在启动过程中，关闭 Wi-Fi 失败!!!!!!");
33                break;
34                case WifiManager.WIFI_STATE_DISABLING: //Wi-Fi 在关闭过程中，
将无法关闭
35                        tv.setText("WIFI 在关闭过程中，关闭 Wi-Fi 失败!!!!!! ");
36                break;
37                case WifiManager.WIFI_STATE_DISABLED:  //Wi-Fi 已经关闭
38                        tv.setText("Wi-Fi 已经关闭，关闭 Wi-Fi 失败!!!!!!!! ");
39                break;
40                case WifiManager.WIFI_STATE_UNKNOWN:   //Wi-Fi 处于未知状态
41                        tv.setText("Wi-Fi 处于未知状态，关闭 Wi-Fi 失败!!!!!!");
42                break;
43                }
44                cb.setText("打开 Wi-Fi");
45                }
```

```
46                }catch(Exception e){e.printStackTrace();}   //打印异常
47            }else if(cb.isChecked()==true){
48                try{                                                //尝试打开 Wi-Fi
49                if(!mWifi.isWifiEnabled()&&mWifi.getWifiState()!=WifiManager.
WIFI_STATE_ENABLING)
50                {                                    //当 Wi-Fi 处于关闭，且处于非打开过程
51                    if(mWifi.setWifiEnabled(true)){          //打开 Wi-Fi
52                        switch(mWifi.getWifiState()){
53                        case WifiManager.WIFI_STATE_ENABLING://若 Wi-Fi 在打开过程中
54                            tv.setText("Wi-Fi 正在启动过程中，打开 Wi-Fi 失败!!!!");
55                        break;
56                        case WifiManager.WIFI_STATE_ENABLED://若 Wi-Fi 已经打开
57                            tv.setText("Wi-Fi 已经打开，打开 Wi-Fi 失败!!!!!!! ");
58                        break;
59                        default:                                    //其他未知错误
60                            tv.setText("由于不明原因，打开 Wi-Fi 失败!!!!!!! ");
61                        break;
62                        }
63                    }else{tv.setText("打开 Wi-Fi 失败");}
64                }else{
65                    switch(mWifi.getWifiState()){
66                        case WifiManager.WIFI_STATE_ENABLING://在打开过程中，打开
失败
67                            tv.setText("WIFI 在打开过程中，打开 Wi-Fi 失败");
68                        break;
69                        case WifiManager.WIFI_STATE_DISABLING://在关闭过程中，打
开失败
70                            tv.setText("Wi-Fi 在关闭过程中，打开 Wi-Fi 失败");
71                        break;
72                        case WifiManager.WIFI_STATE_DISABLED://已经关闭，打开失败
73                            tv.setText("Wi-Fi 已经关闭，打开 Wi-Fi 失败");
74                        break;
75                        case WifiManager.WIFI_STATE_UNKNOWN://不明原因，打开失败
76                            tv.setText("由于不明原因，打开 Wi-Fi 失败");
77                        break;
78                        }}
79                    cb.setText("关闭 WIFI");
80                }catch(Exception e){e.printStackTrace();}   //打印异常
81    }}});}
82        protected void onResume(){
83          try{                                          //取得打开程序当下 Wi-Fi 的状态
84              switch(mWifi.getWifiState()){
85                  ……//该处省略了部分类似代码，读者可自行查看随书光盘中源代码。
86              }
87          }catch(Exception e){e.printStackTrace();}        //打印异常
88          super.onResume();
89        }
90        protected void onPause(){                          //重写 onPause 方法
91          super.onPause();
92    }}
```

- 第 7～19 行为创建 WIFI 对象，并首先判断程序运行后，Wi-Fi 是否已经打开，若处于打开状态，则将 CheckBox 设置为 true 以及将文字设置为关闭 Wi-Fi，否则将 CheckBox 设置为 false，并将文字设置为打开 Wi-Fi。
- 第 20～46 行为设置 CheckBox 的监听器，当单击 CheckBox 时，首先判断 CheckBox 的状态是 true 还是 false，若为 false，则为尝试关闭 Wi-Fi，若 Wi-Fi 处于已打开状态，则成功地将 Wi-Fi 关闭，若 Wi-Fi 处于其他状态，则关闭失败。

- 第 47～81 行为 CheckBox 的状态为 true 时的处理方法，首先判断当 Wi-Fi 处于关闭，且处于非打开过程，然后根据当前 Wi-Fi 的状态判断能否打开 Wi-Fi。
- 第 82～92 行为重写 onResume 方法和 onPause 方法。

实例 13　还原手机桌面背景

手机的桌面背景可以进行设置，同样可以将其还原，在这里将要介绍如何将手机背景还原为最初背景。

【实例描述】

本软件实现简单，在界面中仅仅一个按钮，当单击按钮时，将屏幕的背景置换为手机桌面默认的背景。本实例的运行效果图，如图 5-13 所示。

图 5-13　还原为默认背景实例运行效果图

提示：在图 5-13 中，第一幅图为将手机桌面背景置换，然后运行程序，单击按钮后，Toast 提示已还原，第三幅图为还原后的桌面背景。

【实现过程】

在本实例中所用到的技术为使用 clearWallpaper()方法，当调用该方法时，自动将已更改的手机桌面背景还原为默认背景，由于在调用该方法时可能有一场抛出，所以需要捕获异常。并且，在该实例中需要添加权限，其权限代码为：<uses-permission android:name="android.permission.SET_WALLPAPER"></uses-permission>，详见光盘源代码/第 5 章/Sample5_13/AndroidMainifest.xml。

【代码解析】

首先要介绍的是 Sample5_13_Activity 实现类，其代码如下。

代码位置：见光盘源代码/第 5 章/Sample5_13/src/com/bn/chap5/clear/Sample5_13_Activity.class。

```
1    package com.bn.chap5.clear;
2    ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
```

```
3   public class Sample5_13_Activity extends Activity {
4     Button button;                                          //声明引用
5      public void onCreate(Bundle savedInstanceState) {
6        super.onCreate(savedInstanceState);
7        setContentView(R.layout.main);                       //设置当前界面
8        button=(Button)this.findViewById(R.id.Button01);     //获取 Button 对象
9        button.setOnClickListener(                           //添加监听器
10             new OnClickListener(){
11                   public void onClick(View v) {
12                       try{
13                           Sample5_13_Activity.this.clearWallpaper();
                                                    //调用 clearWallpaper
14                          Toast.makeText(          //设置 Toast
15                                  Sample5_13_Activity.this,
16                                  "手机桌面背景已经还原为默认设置!! ",
17                                  Toast.LENGTH_SHORT).show();//显示 Toast
18                      }catch(Exception e){e.printStackTrace();}//打印异常
19  }});}}
```

提示：上述代码中第 9～19 行为实现 button 的监听器，在重写的 onClick 方法中调用 clearWallpaper 方法，并设置 Toast 提醒用户，界面已经还原为默认设置。

实例 14　设置手机桌面背景

每个人都会有自己喜欢的相片，将自己喜欢的相片设为手机的背景基本上已成为每个手机用户的习惯，在本部分将介绍如何运用程序设置自己喜欢的手机桌面背景。

【实例描述】

本软件结构简单，界面由一个 TextView 和两个 Button 构成，当单击更换桌面背景按钮时，系统自动将已设定好的图片设置为手机的桌面背景，同时出现 Toast 提示用户，桌面背景已经置换成功。

当单击退出查看时，即为退出程序，观看已置换的手机桌面背景。本实例的运行效果图，如图 5-14 所示。

图 5-14　置换桌面效果图

提示：在图 5-14 中，第一幅图为实例运行前的背景图片，第二幅图为实例运行后选择单击更换桌面背景按钮后，Toast 提示已更换桌面背景，第三幅图为更换后的桌面背景。

【实现过程】

在本实例中使用了系统中 setWallpaper()方法，当调用该方法时，将手机桌面背景设置为已选定的图片，该图片首先要经过 InputStream 处理成数据流。在该实例中同样需要为其声明权限，其权限为：<uses-permission android:name="android.permission.SET_WALLPAPER"></uses-permission>。

【代码解析】

在这里将要详细介绍该实例功能的实现，代码如下。

代码位置：见光盘源代码/第 5 章/Sample5_14/src/com/bn/chap5/zdybj/Sample5_14_Activity.class。

```
1   package com.bn.chap5.zdybj;
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3   public class Sample5_14_Activity extends Activity {
4       InputStream is;                                          //声明输入流
5       Button button;                                           //声明 Button 引用
6       Button exit;
7      public void onCreate(Bundle savedInstanceState) {
8         super.onCreate(savedInstanceState);
9         setContentView(R.layout.main);                         //设置当前界面
10        button=(Button)this.findViewById(R.id.Button01);       //获取对象
11        exit=(Button)this.findViewById(R.id.Button02);
12        button.setOnClickListener(                             //添加监听器
13              new OnClickListener(){
14                    public void onClick(View v) {
15                        Resources resource=getBaseContext().
getResources();                                                  //获取 resources
16                        is=resource.openRawResource(R.drawable.wall2);
                                                                 //加载图片
17                        try{
18                            Sample5_14_Activity.this.setWallpaper(is);
                                                                 //设置桌面背景
19                        }catch(Exception e){e.printStackTrace();}
                                                                 //打印异常
20                        Toast.makeText(
21                        Sample5_14_Activity.this,
22                        "已经成功置换桌面背景!! ",               //设置显示文字
23                        Toast.LENGTH_SHORT).show();            //显示 Toast
24  }});
25        exit.setOnClickListener(                               //添加监听器
26          new OnClickListener(){
27                  public void onClick(View v) {
28                      System.exit(0);                          //退出程序
29  }});}}
```

- 第 8～24 行为设置 main.xml 为当前界面，并获取 Button 对象，同时为“确定”按钮添加监听器，当单击该按钮时，为手机桌面重新设置背景。
- 第 25～28 行为给退出按钮添加监听器，该按钮实现的功能为退出该程序。

实例 15　轻松获取手机桌面背景

上面的两个实例介绍了如何设置桌面背景以及还原桌面背景，在本节将介绍如何获取当前手机桌面背景。

【实例描述】

本软件通过设置一组 RadioButton 来完成以上功能，其中一个按钮为单击时获取当前手机桌面的背景，另一个按钮为单击后设置为默认图片。本实例的运行效果图，如图 5-15 所示。

图 5-15　实例运行效果图

提示：在图 5-15 中，第一幅图为实例开始运行的效果，第二幅图为单击获取桌面背景按钮时，在 ImageView 中显示桌面背景，第三幅图为单击默认图片按钮时，在 ImageView 中显示默认图片。

【实现过程】

在本软件中，控件部分运用了 RadioGroup 和 ImageView，当单击其中一个 RadioButton 时，设置 ImageView 显示的图片，其中获取桌面背景图片的方法为 getWallpaper()方法。在该实例中同样需要为其声明权限，其权限为：

```
<uses-permission android:name="android.permission.SET_WALLPAPER"></uses-permission>。
```

【代码解析】

经过上面的理论介绍，接下来要介绍该实例的核心代码，如下所示。

代码位置：见光盘源代码/第 5 章/Sample5_15/src/com/bn/chap5/get/Sample5_15_Activity.class。

```
1   package com.bn.chap5.get;
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3   public class Sample5_15_Activity extends Activity {
4       RadioGroup rg;
5       RadioButton rbGet;                                    //获取桌面背景按钮
6       RadioButton rbReset;                                  //默认背景按钮
7       ImageView iv;
8      public void onCreate(Bundle savedInstanceState) {
```

```
    9          super.onCreate(savedInstanceState);
    10         setContentView(R.layout.main);                          //设置当前界面
    11         rg=(RadioGroup)this.findViewById(R.id.RadioGroup01);   //获取对象
    12         rbGet=(RadioButton)this.findViewById(R.id.RadioButton01);
    13         rbReset=(RadioButton)this.findViewById(R.id.RadioButton02);
    14         iv=(ImageView)this.findViewById(R.id.ImageView01);//获取 ImageView 对象
    15         RadioGroup.OnCheckedChangeListener  change=new  RadioGroup.OnChecked
ChangeListener() {
    16               public void onCheckedChanged(RadioGroup group, int checkedId) {
    17                   if(checkedId==rbGet.getId()){           //若单击的是获取按钮
    18                       iv.setImageDrawable(getWallpaper()); //获取手机桌面背景并设置
    19                   }
    20                   if(checkedId==rbReset.getId()){         //若单击的是还原按钮
    21                       iv.setImageDrawable(getResources().getDrawable(R.
drawable.icon));
    22                   }}};
    23           rg.setOnCheckedChangeListener(change);      //为 RadioGroup 添加监听器
    24   }}
```

提示：上述代码中通过为 RadioGroup 设置监听器，若单击的是获取手机桌面背景图片按钮，将 ImageView 显示的图片设置为当前手机桌面背景；若单击的是默认图片按钮，则将 ImageView 显示的图片设置为默认图片。

实例 16　轻松查看手机的相关信息

每一部手机都有其详细的信息，这些信息包括了手机号码、电信网络国别等等，在本节后面部分将详细介绍如何获取这些信息。

【实例描述】

本软件设计简易，当软件运行后，在界面上有一个按钮，当单击按钮时，即可获取手机上的相关信息，信息包括手机号码、电信网络国别、电信公司代码、电信公司名称、手机 SIM 码、手机通信类型、手机网络类型、是否漫游以及蓝牙和 WIFI 的状态等。

当单击其中的一条信息时，会有 Toast 弹出，给出所单击的信息。本实例的运行效果图，如图 5-16 所示。

图 5-16　手机信息获取效果图

提示：在手机信息获取运行效果图中，第一幅图为软件开始运行的界面，当单击查看手机信息按钮时，在界面中显示手机的详细信息，当单击其中的一条信息时，Toast 提示所单击的信息。

【实现过程】

该界面继承自 Activity，通过使用 TelephonyManager 获取手机的电信网络信息，以及通过运用 Android.provider.Settings.System 取得手机相关信息。将获取的信息首先存放在 List 中，然后将 List 中的信息放到 ListView 中，当单击 Button 按钮时，将其显示在界面中。

使用 TelephonyManager 获取手机 SIM 卡的信息需要为手机添加权限声明，权限代码为：<uses-permission android:name="android.permission.READ_PHONE_STATE"></uses-permission>。

【代码解析】

在本部分要讲解的是该软件的实现类，该类代码如下。

代码位置：见光盘源代码/第 5 章/Sample5_16/src/com/bn/chap5/telephon/Sample5_16_Activity.class。

```
1   package com.bn.chap5.telephon;
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3   public class Sample5_16_Activity extends Activity {
4       ……//该处省略了部分代码，读者可自行查看随书光盘中源代码。
5       public void onCreate(Bundle savedInstanceState) {
6           super.onCreate(savedInstanceState);
7           setContentView(R.layout.main);                          //设置当前界面
8           lv=(ListView)this.findViewById(R.id.ListView01);  //获取 ListView 对象
9           tm=(TelephonyManager)getSystemService(TELEPHONY_SERVICE);  //创建对象
10          cr=Sample5_16_Activity.this.getContentResolver();
                                                    //创建 ContentResolver 对象
11          bCheck=(Button)this.findViewById(R.id.Button01);  //创建 Button 对象
12          String str=null;                                        //记录 cr 获取的信息
13          name.add("手机号码：");                                  //添加信息
14          ……//该处省略了部分代码，读者可自行查看随书光盘中源代码。
15          name.add("WIFI 状态：");
16          if(tm.getLine1Number()!=null){                          //判断是否有手机号码
17            list.add(tm.getLine1Number());                        //若有，加入列表。
18          }else{list.add("无法取得您的电话号码");}                  //若没有添加信息提示
19          if(!tm.getNetworkCountryIso().equals("")){              //判断电信网络国别
20            list.add(tm.getNetworkCountryIso());                  //若有，加入列表
21          }else {list.add("无法取得您的电信网络国别");}             //若没有添加信息提示
22          if(!tm.getNetworkOperator().equals("")){                //电信公司代码
23            list.add(tm.getNetworkOperator());
24          }else{list.add("无法获取电信公司代码");}
25          if(!tm.getNetworkOperatorName().equals("")){            //电信公司名称
26            list.add(tm.getNetworkOperatorName());
27          }else{list.add("无法获取电信公司名称");}
28          if(tm.getSimSerialNumber()!=null){                      //手机 SIM 码
29            list.add(tm.getSimSerialNumber());
30          }else{list.add("无法获取手机 SIM 码");}
31          if(tm.getPhoneType()==TelephonyManager.PHONE_TYPE_GSM){ //手机通信类型
32            list.add("GSM");
33          }else if(tm.getPhoneType()==TelephonyManager.PHONE_TYPE_CDMA){
```

```
34            list.add("CDMA");                                    //添加信息
35          }else{list.add("无法获取手机通信类型");}
36          if(tm.getNetworkType()==TelephonyManager.NETWORK_TYPE_EDGE){
                                                                   //获取手机网络类型
37            list.add("EDGE");
38          }else if(tm.getNetworkType()==TelephonyManager.NETWORK_TYPE_GPRS){
39            list.add("GPRS");                                    //添加 GPRS
40          }else if(tm.getNetworkType()==TelephonyManager.NETWORK_TYPE_UMTS){
41            list.add("UMTS");                                    //添加 UMTS
42          }else if(tm.getNetworkType()==TelephonyManager.NETWORK_TYPE_HSDPA){
43            list.add("HSDPA");
44          }else{list.add("无法获取手机网络类型");}
45          if(tm.isNetworkRoaming()){                           //手机是否漫游
46            list.add("手机漫游中");
47          }else{list.add("手机无漫游");}
48          str=android.provider.Settings.System.getString(     //获取字符串
49                cr,android.provider.Settings.System.BLUETOOTH_ON
50                );
51          if(str.equals("1")){                                 //若为 1
52            list.add("蓝牙已打开");                            //添加蓝牙已打开
53          }else{list.add("蓝牙未打开");}
54          str=android.provider.Settings.System.getString(cr,  android.provider.
Settings.System.WIFI_ON);
55          if(str.equals("1")){
56            list.add("WIFI 已打开");                           //添加 WIFI 已打开
57          }else{
58            list.add("WIFI 未打开");
59          }
```

- 第 6～15 行为创建对象，并向 name 列表中添加信息。
- 第 16～44 行为通过 TelephonyManager 对象分别获取手机号码、电信网络国别、电信公司代码、手机 SIM 码、手机通信类型、手机网络类型，在获取这些信息时，需要判断是否获取成功，若没有成功则向信息列表中添加无法获取该信息提示。
- 第 45～59 行为通过 ContentResolver 对象分别获取手机是否处于漫游状态，蓝牙是否已经打开。

接下来介绍的是 Button 按钮监听器的实现，其代码如下。

代码位置：见光盘源代码/第 5 章/Sample5_16/src/com/bn/chap5/telephon/Sample5_16_Activity.class。

```
1   bCheck.setOnClickListener(
2       new OnClickListener(){
3           public void onClick(View v) {
4               BaseAdapter ba=new BaseAdapter(){            //创建适配器
5                   public int getCount() {
6                       return list.size();                  //获取 list 数量
7                   }
8                   public Object getItem(int position) {
9                       return null;
10                  }
11                  public long getItemId(int position) {
12                      return 0;
13                  }
14                  public View getView(int arg0, View arg1, ViewGroup arg2) {
15                      LinearLayout ll=new LinearLayout(Sample5_16_Activity.
this);                                                       //创建对象
16                      ll.setOrientation(LinearLayout.HORIZONTAL);
                                                             //设置空间摆放顺序
17                      ll.setPadding(5, 5, 5, 5);           //设置留白
```

```
18                      TextView tv=new TextView(Sample5_16_Activity.this);
                                                        //初始化 TextView
19                      tv.setTextColor(Color.BLACK);   //设置字体颜色
20                      tv.setPadding(5,5,5,5);         //设置留白
21                      tv.setText(name.get(arg0));     //添加任务名字
22                      tv.setGravity(Gravity.LEFT);    //左对齐
23                      tv.setTextSize(18);             //字体大小
24                      ll.addView(tv);                 //添加 TextView
25                      TextView tvv=new TextView(Sample5_16_Activity.this);
                                                        //初始化 TextView
26                      tvv.setTextColor(Color.BLACK);  //设置字体颜色
27                      tvv.setPadding(5,5,5,5);
28                      tvv.setText(list.get(arg0));    /添加任务名字
29                      tvv.setGravity(Gravity.LEFT);   //左对齐
30                      tvv.setTextSize(18);            //字体大小
31                      ll.addView(tvv);                //添加 TextView
32                      return ll;
33                      }
34                  };
35                  lv.setAdapter(ba);                  //设置适配器
36                  lv.setOnItemClickListener(          //设置选中菜单的监听器
37                      new OnItemClickListener(){
38                          publicvoidonItemClick(AdapterView<?>arg0,Viewarg1,
39                              int arg2, long arg3) {
40                          Toast.makeText(             //创建 Toast
41                                  Sample5_16_Activity.this,
42                                  name.get(arg2)+""+list.get(arg2),
                                                        //显示的信息
43                                  Toast.LENGTH_SHORT).show(); //显示 Toast
44  }});});}}
```

- 第 4～34 行为创建适配器，首先获取列表的数量，创建 LinearLayout 对象，并以此向 ListView 中添加列表中的信息。
- 第 35～44 行配置适配器，并为 ListView 添加监听器，实现单击相应信息，弹出 Toast 提示

实例 17　查看 SIM 卡的详细信息

手机的 SIM 卡是手机的一个重要的组成部分，没有 SIM 卡也就不能正常拨打电话，在本节将介绍如何获取 SIM 卡的信息。

【实例描述】

本软件设计模式与上面的例子相似，通过在主界面中设置一个按钮，当单击按钮时，显示获取的 SIM 卡的信息，从而实现一键查询的功能。本实例的运行效果图，如图 5-17 所示。

提示：在图 5-17 中，第一幅图为软件开始运行时的主界面，当单击查看 SIM 卡信息按钮时，显示 SIM 卡的详细信息，单击其中一条信息时，弹出 Toast 提示。

【实现过程】

该界面继承自 Activity，通过使用 TelephonyManager 获取手机 SIM 卡的相关信息，并将获得

的 SIM 卡的状态、卡号、SIM 卡供应商、SIM 卡供应商名称、SIM 卡国别以 ListView 形式呈现在界面上。使用 TelephonyManager 获取手机 SIM 卡的信息需要为手机添加权限声明，权限代码为：<uses-permission android:name="android.permission.READ_PHONE_STATE"></uses-permission>，详见光盘源代码/第 5 章/Sample5_17/AndroidMainifest.xml。

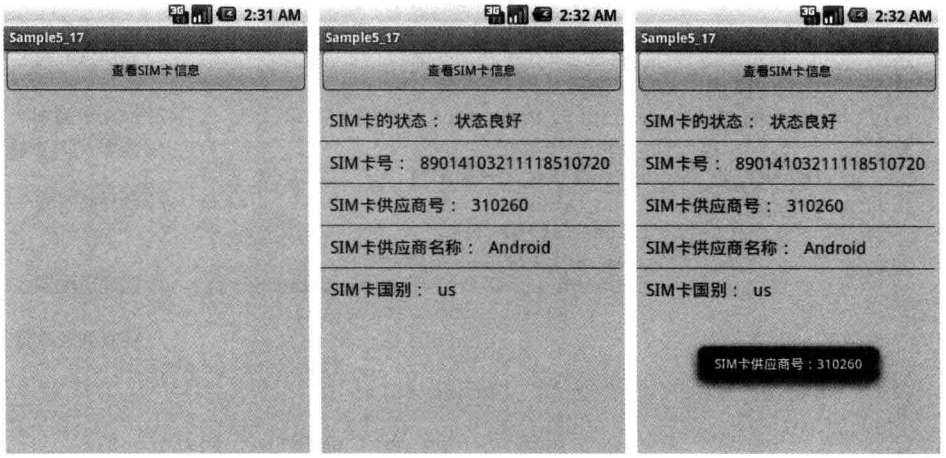


图 5-17　实例运行效果图

【代码解析】

下面要介绍的是 Sample5_17_Activity 的实现类，其代码如下。

代码位置：见光盘源代码/第 5 章/Sample5_17/src/com/bn/chap5/sim/Sample5_17_Activity.class。

```
1   package com.bn.chap5.sim;
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3   public class Sample5_17_Activity extends Activity {
4       private ListView lv;                                    //声明 ListView
5       private TelephonyManager tm;                            //声明 TelephonyManager
6       private List<String> list=new ArrayList<String>();      //声明信息列表
7       private List<String> name=new ArrayList<String>();      //声明名称列表
8       private Button bCheck;                                  //声明 Button
9      public void onCreate(Bundle savedInstanceState) {
10        super.onCreate(savedInstanceState);
11        setContentView(R.layout.main);                        //设置主界面
12        lv=(ListView)this.findViewById(R.id.ListView01);      //获取对象
13        tm=(TelephonyManager)getSystemService(TELEPHONY_SERVICE);//创建对象
14        bCheck=(Button)this.findViewById(R.id.Button01);
15        name.add("SIM 卡的状态：");                            //添加名称信息
16        name.add("SIM 卡号：");
17        name.add("SIM 卡供应商号：");
18        name.add("SIM 卡供应商名称：");                        //添加名称信息
19        name.add("SIM 卡国别：");
20        if(tm.getSimState()==TelephonyManager.SIM_STATE_READY){ //SIM 卡状态
21          list.add("状态良好");
22        }else if(tm.getSimState()==TelephonyManager.SIM_STATE_ABSENT){//无 SIM 卡
23          list.add("您目前没有 SIM 卡");
24        }else  if(tm.getSimState()==TelephonyManager.SIM_STATE_UNKNOWN){//SIM
卡处于未知状态
25          list.add("SIM 卡处于未知状态");
26        }
```

```
        if(tm.getSimSerialNumber()!=null){                    //SIM 卡卡号
          list.add(tm.getSimSerialNumber());                  //添加 SIM 卡卡号
        }else{list.add("没有 SIM 卡卡号");}
        if(!tm.getSimOperator().equals("")){                  //SIM 卡供应商代号
          list.add(tm.getSimOperator());                      //添加供应商代号
        }else{list.add("没有 SIM 卡供应商代号");}
        if(!tm.getSimOperatorName().equals("")){              //SIM 卡供应商名称
          list.add(tm.getSimOperatorName());                  //添加供应商名称
        }else{list.add("没有 SIM 卡供应商名称");}
        if(!tm.getSimCountryIso().equals("")){                //SIM 卡国别
          list.add(tm.getSimCountryIso());                    //添加 SIM 卡国别
        }else{list.add("无法获取 SIM 国别");}
        bCheck.setOnClickListener(                            //添加监听器
              new OnClickListener(){
                  public void onClick(View v) {
                      BaseAdapter ba=new BaseAdapter(){       //创建适配器
                          public int getCount() {
                              return list.size();}            //获取列表数量
                          public Object getItem(int position) {
                              return null;
                          }
                          public long getItemId(int position) {
                              return 0;
                          }
                          public View getView(int arg0, View arg1, ViewGroup
arg2) {
                              LinearLayout ll=new LinearLayout(Sample5_17_
Activity.this);
ll.setOrientation(LinearLayout.HORIZONTAL);//设置布局
                              ll.setPadding(5, 5, 5, 5);      //设置留白
                              TextView tv=new TextView(Sample5_17_Activity.
this);
                              tv.setTextColor(Color.BLACK);   //设置字体颜色
                              tv.setPadding(5,5,5,5);
                              tv.setText(name.get(arg0));     //设置显示名称
                              tv.setGravity(Gravity.LEFT);    //左对齐
                              tv.setTextSize(18);             //字体大小
                              ll.addView(tv);                 //添加 TextView
                              TextView tvv=new TextView(Sample5_17_Activity.
this);
                              tvv.setTextColor(Color.BLACK);//设置字体颜色
                              tvv.setPadding(5,5,5,5);
                              tvv.setText(list.get(arg0));    //设置显示信息
                              tvv.setGravity(Gravity.LEFT);   //左对齐
                              tvv.setTextSize(18);            //字体大小
                              ll.addView(tvv);                //添加 View
                              return ll;
                          }};
                  lv.setAdapter(ba);                          //设置适配器
                  lv.setOnItemClickListener(                  //设置选中菜单的监听器
                          new OnItemClickListener(){
                                  public  void  onItemClick(AdapterView<?>
arg0, View arg1,
                                          int arg2, long arg3) {
                                      Toast.makeText(   //创建 Toast
                                          Sample5_17_Activity.this,
                                          name.get(arg2)+""+list.get(arg2),
```

```
79                                          Toast.LENGTH_SHORT).show();
                                                        //显示 Toast
80   }});}});}}
```

- 第 4～19 行为声明引用和创建对象，并向 name 列表中添加名称。
- 第 20～38 行为通过 TelephonyManager 对象判断 SIM 卡的状态、SIM 卡卡号、SIM 卡供应商代号、供应商名称以及 SIM 卡国别，最后将获取的信息添加到 list 列表中。
- 第 39～71 行为 Button 按钮的监听器，当单击 Button 按钮时，首先获取列表中信息的数量，然后按照顺序依次向 ListView 中添加信息。
- 第 72～80 行为 ListView 监听器，当单击 ListView 中某个选项时，弹出 Toast 提示所单击选项的信息。

实例 18　按键移动图片——方向键的应用

在本部分，将通过使用键盘中方向键控制图片移动位置，来详细介绍方向键的应用。

【实例描述】

在该软件的主界面中有一个 Button 按钮和一个 ImageView，通过使用手机上的方向键对 ImageView 的位置进行控制，同样可以使用触屏方式移动图片，当按下 Button 时，将图片还原到图片的最初位置。

本实例的运行效果图，如图 5-18 所示。

图 5-18　运行效果图

提示：在图 5-18 中，第一幅图为图片的初始位置，后两幅图表示通过键盘将图片移动到此位置。

【实现过程】

在该实例中，当单击键盘中上下左右时，图片在 Layout 中的相对位置就会发生变化，这是通过重写 onKeyDown()方法实现的。

本实例中同样通过重写 onTouchEvent()方法来更改图片的相对位置。

【代码解析】

接下来要介绍的是该软件的核心代码部分，首先介绍的是该类中 onKeyDown 方法的实现，其代码如下所示。

代码位置：见光盘源代码/第 5 章/Sample5_18/src/com/bn/chap5/dpad/Sample5_18_Activity.class。

```
1   package com.bn.chap5.dpad;
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3   public class Sample5_18_Activity extends Activity {
4       ……//该处省略了部分代码，读者可自行查看随书光盘中源代码。
5      public void onCreate(Bundle savedInstanceState) {
6           super.onCreate(savedInstanceState);
7           setContentView(R.layout.main);                          //设置主界面
8           DisplayMetrics dm=new DisplayMetrics();                 //获取手机的分辨率
9           getWindowManager().getDefaultDisplay().getMetrics(dm);
10          screenHeight=dm.heightPixels;                           //获取屏幕高度
11          screenWidth=dm.widthPixels;                             //获取屏幕宽度
12          button=(Button)this.findViewById(R.id.Button01);//获取 Button 对象
13          iv=(ImageView)this.findViewById(R.id.ImageView01); //获取 ImageView 对象
14          ivHeight=50;                                            //图片大小
15          ivWidth=50;
16          resetPicter();                                          //将图片设置到屏幕中间
17          button.setOnClickListener(                              //给按钮添加监听器
18              new OnClickListener(){
19                  public void onClick(View v) {
20                      resetPicter();                              //将图片设置到屏幕中间
21                  }});}
22      public void resetPicter(){
23      ivX=(screenWidth-ivWidth)/2;                                //图片的 X 坐标
24      ivY=(screenHeight-ivHeight)/2;                              //图片的 Y 坐标
25      iv.setLayoutParams(                                         //设置图片位置
26          new AbsoluteLayout.LayoutParams(ivWidth, ivHeight, ivX, ivY)
27      );}
28      public boolean onKeyDown(int keyCode,KeyEvent event){//重写 onKeyDown 方法
29       switch(keyCode){
30          case 19:                                                //向上键
31              ivY=ivY-span;                                       //计算 Y 坐标新位置
32              if(ivY<0){ivY=0;}                                   //当小于 0 时，不再减少
33              iv.setLayoutParams(                                 //设置图片位置
34                  new AbsoluteLayout.LayoutParams(ivWidth, ivHeight, ivX, ivY)
35                  );
36          break;
37          case 20:                                                //向下键
38              ivY=ivY+span;                                       //计算 Y 坐标新位置
39              if(ivY>screenHeight-2*ivHeight){                    //若大于阈值
40                  ivY=screenHeight-2*ivHeight;                    //保持值不变
41              }
42              iv.setLayoutParams(                                 //设置图片位置
43                  new AbsoluteLayout.LayoutParams(ivWidth, ivHeight, ivX, ivY)
44                  );
45          break;
46          case 21:                                                //向左键
47              ivX=ivX-span;                                       //计算 X 坐标新位置
48              if(ivX<0){ivX=0;}                                   //若小于阈值
49              iv.setLayoutParams(                                 //设置图片位置
```

```
50                  new AbsoluteLayout.LayoutParams(ivWidth, ivHeight, ivX, ivY)
51                  );
52          break;
53          case 22:                                          //向右
54              ivX=ivX+span;                                 //设置 X 坐标新位置
55              if(ivX>screenWidth-ivWidth){                  //若大于阈值
56                  ivX=screenWidth-ivWidth;                  //保持值不变
57              }
58              iv.setLayoutParams(                           //设置图片位置
59                  new AbsoluteLayout.LayoutParams(ivWidth, ivHeight, ivX, ivY)
60                  );
61          break;
62      }
63      return true;                                          //返回 true
64  }
```

- 第 5～21 行为重写 onCreate()方法，在该方法中设置主界面，获取手机的分辨率，获取 Button 和 ImageView 对象，设置图片的大小，并为 Button 添加监听器。
- 第 22～27 行为创建 resetPicter()方法，在该方法中设置图片在手机 XY 轴上的坐标为手机屏幕的中心位置，并设置图片在 Layout 中的位置。
- 第 28～64 行为重写 onKeyDown()对象，在该方法中检测上下左右键的触发，当单击向上键时，图片上移，单击向下键时，图片下移，单击向左键时，图片左移，单击向右键时，图片右移。

接下来要介绍的是 onTouchEvent()方法的实现类，其代码如下。

代码位置：见光盘源代码/第 5 章/Sample5_18/src/com/bn/chap5/dpad/Sample5_18_Activity.class。

```
1       public boolean onTouchEvent(MotionEvent e){
2           switch(e.getAction()){
3               case MotionEvent.ACTION_DOWN:
4                   x=e.getX();y=e.getY();                    //获取 XY 坐标
5               break;
6               case MotionEvent.ACTION_UP:
7                   float tempX=e.getX();float tempY=e.getY(); //获取 XY 坐标
8                   float dx=tempX-x;float dy=tempY-y;         //两次坐标差值
9                   if(dx>0&&dy>0){                 //若均大于零
10                      ivX=ivX+span;ivY=ivY+span;
11                      iv.setLayoutParams(         //设置图片位置
12                      new AbsoluteLayout.LayoutParams(ivWidth, ivHeight, ivX,
ivY)
13                      );}
14                  if(dx<0&&dy>0){                 //若 x 轴上小于零，y 轴上大于零
15                      ivX=ivX-span;ivY=ivY+span;
16                      iv.setLayoutParams(         //设置图片位置
17                      new AbsoluteLayout.LayoutParams(ivWidth, ivHeight, ivX,
ivY)
18                      );}
19                  if(dx>0&&dy<0){                 //若 X 轴上大于零，y 轴上小于零
20                      ivX=ivX+span;ivY=ivY-span;
21                      iv.setLayoutParams(         //设置图片位置
22                      new AbsoluteLayout.LayoutParams(ivWidth, ivHeight, ivX,
ivY)
23                      );}
24                  if(dx<0&&dy<0){                 //若均小于零
25                      ivX=ivX-span;ivY=ivY-span;
26                      iv.setLayoutParams(         //设置图片位置
27                      new AbsoluteLayout.LayoutParams(ivWidth, ivHeight, ivX,
ivY)
```

```
28                          );}
29                  break;
30                  }
31          return true;
32  }}
```

提示：上述代码中为重写 onTouchEvent()方法，在该方法中首先记录触摸屏幕时的 XY 轴上的坐标，然后记录抬起时 XY 轴上的坐标，根据两次坐标的差值，判断图片运动的方向，设置图片的位置。

实例 19　查看正在运行的程序

如何查看手机中正在运行的程序，在本部分将对其进行详细介绍。

【实例描述】

在该软件中有两个 Button 按钮，一个按钮实现的功能为显示手机中正在运行的程序，另一个按钮为清空记录程序名称和 ID 的列表。当单击查看正在运行的程序时，在按钮下方显示正在运行的程序，单击清空程序列表按钮，清空记录。

该软件可以轻松地查看手机中正在运行的程序，能够实时对自己的手机有所了解。本实例的运行效果图，如图 5-19 所示。

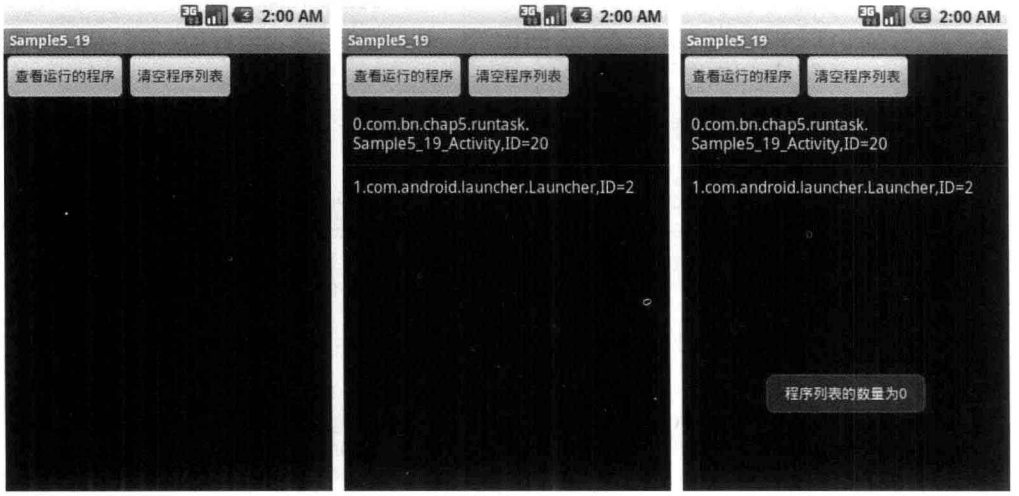


图 5-19　查看正在运行的程序

提示：在图 5-19 中，第一幅图为程序开始运行时的界面，单击查看运行的程序按钮后，效果如图 5-19 中第二幅图所示，单击清空程序列表如图 5-19 中第三幅图所示。

【实现过程】

在本实例中，通过 ActivityManager.getRunningTasks 方法来获取正在运行中的应用程序，最后用 ListView 将获取的信息显示在手机屏幕上，程序的运行需要为其声明权限，其权限设置的代码为：<uses-permission android:name="android.permission.GET_TASKS"></uses-permission>。

设置两个 Button 和一个 ListView，通过按钮的单击事件控制 ListView 中显示的信息。

【代码解析】

在本部分将要介绍的是该实例的核心代码，如下所列。

代码位置：见光盘源代码/第 5 章/Sample5_19/src/com/bn/chap5/runtask/Sample5_19_Activity.class。

```
1   package com.bn.chap5.runtask;
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3   public class Sample5_19_Activity extends Activity {
4       ActivityManager am;                                    //获取运行程序的引用
5       Button button;                                         //Button 引用
6       Button bClear;                                         //清空按钮引用
7       ListView lv;                                           //ListView 引用
8       List<String> list=new ArrayList<String>();             //创建 list 对象
9       public void onCreate(Bundle savedInstanceState) {
10        super.onCreate(savedInstanceState);
11        setContentView(R.layout.main);                       //设置当前界面
12        button=(Button)this.findViewById(R.id.Button01);     //获取 Button 对象
13        bClear=(Button)this.findViewById(R.id.Button02);     //获取清除按钮对象
14        lv=(ListView)this.findViewById(R.id.ListView01);     //获取 ListView 对象
15        bClear.setOnClickListener(                           //添加监听器
16          new OnClickListener(){
17                  public void onClick(View v) {
18                      list.clear();                          //清空列表
19                      Toast.makeText(                        //显示 Toast 提示
20                              Sample5_19_Activity.this,
21                              "程序列表的数量为"+list.size(),   //显示的信息
22                              Toast.LENGTH_SHORT             //显示的时间
23                              ).show();
24  }});
25        button.setOnClickListener(                           //添加监听器
26          new OnClickListener(){
27                  public void onClick(View v) {
28                      try{
29                          am=(ActivityManager)Sample5_19_Activity.this.
getSystemService
30                              (ACTIVITY_SERVICE); //创建 ActivityManager 对象
31                          List<ActivityManager.RunningTaskInfo> l=
32                              am.getRunningTasks(20);    //获取正在运行的程序
33                          if(l.size()==0){               //若没有正在运行的程序
34                              Toast.makeText(            //弹出 Toast 提示
35                                      Sample5_19_Activity.this,
36                                      "目前没有正在运行的程序！", //显示的信息
37                                      Toast.LENGTH_SHORT     //显示的时间
38                                      ).show();
39                              return;
40                          }
41                          for(int i=0;i<l.size();i++){
42
list.add(i+"."+l.get(i).baseActivity.getClassName()+",ID="+l.get(i).id);
43                          }
44                          BaseAdapter ba=new BaseAdapter(){  //创建适配器
45                              public int getCount() {
46                                  return list.size();   //list 的数量
47                              }
48                              public Object getItem(int position) {
49                                  return null;
```

```
50                                  }
51                                  public long getItemId(int position) {
52                                      return 0;
53                                  }
54                                  public  View  getView(int  arg0,  View  arg1,
ViewGroup arg2) {
55                                      LinearLayout ll=new LinearLayout(Sample5_
19_Activity.this);
56
ll.setOrientation(LinearLayout.HORIZONTAL);
57                                      ll.setPadding(5, 5, 5, 5);     //设置留白
58                                      TextView   tv=new   TextView(Sample5_19_
Activity.this);
59                                      tv.setTextColor(Color.WHITE);
                                                                  //设置字体颜色
60                                      tv.setPadding(5,5,5,5);
61                                      tv.setText(list.get(arg0)); //添加任务名字
62                                      tv.setGravity(Gravity.LEFT);   //左对齐
63                                      tv.setTextSize(16);            //字体大小
64                                      ll.addView(tv);            //添加 TextView
65                                      return ll;
66                                  }};
67                          lv.setAdapter(ba);                  //设置适配器
68                          lv.setOnItemClickListener(          //设置选中菜单的监听器
69                                  new OnItemClickListener(){
70                                          public void onItemClick(Adapter
View<?> arg0, View arg1,
71                                                  int arg2, long arg3) {
72                                              Toast.makeText(   //Toast 提示
73
Sample5_19_Activity.this,
74                                              list.get(arg2)+"", //显示的信息
75                                              Toast.LENGTH_SHORT//显示的时间
76                                                      ).show();
77                                  }});
78                          }catch(Exception e){e.printStackTrace();}   //打印异常
79  }});}}
```

其中：

- 第 15～24 行为清空程序列表按钮的监听器，单击按钮后清空程序列表，并使用 Toast 显示列表的数量。
- 第 25～79 行为查看正在运行程序按钮的监听器，在该监听器中首先创建 ActivityManager 对象，并获取正在运行的程序，若没有正在运行的程序，则提示目前没有正在运行的程序，若有则创建适配器，并在 ListView 中显示出来，当单击相应的信息时，使用 Toast 显示所单击的信息。

实例 20　手机屏幕更改时信息的捕捉和提醒

Android 手机能够根据手机的姿态自动变换为横屏或者竖屏，在本节将会介绍如何捕捉以及控制手机屏幕的更改信息。

【实例描述】

在该软件中，首先提示的是当前手机状态的分辨率，然后是两个按钮，当单击更换为横屏按钮时，屏幕将切换为横屏，并更改屏幕的分辨率。此时将按钮更换为横屏，设置为不可单击，

更换为竖屏按钮可以单击。

当单击更换为竖屏按钮时，手机屏幕切换为竖屏，并更换屏幕的分辨率。本实例的运行效果图，如图 5-20 所示。

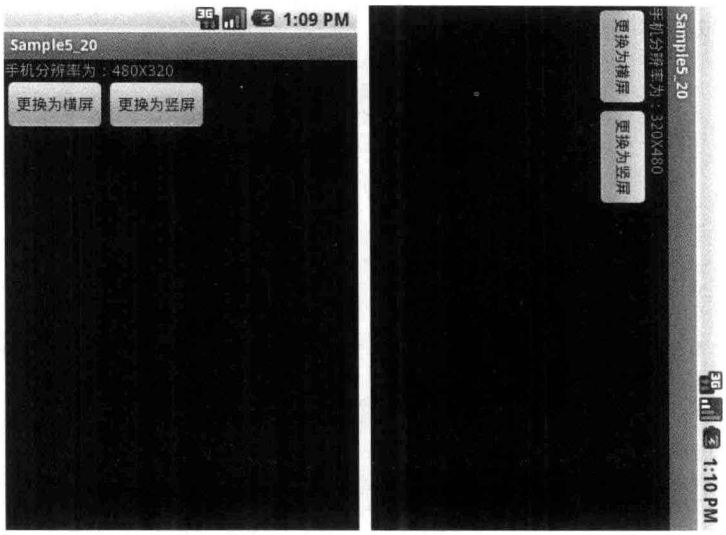


图 5-20　竖屏效果和横屏效果图

提示：在图 5-20 中，为竖屏与横屏时的状态。

【实现过程】

在本实例中，重写 Actibvity 中的 onConfigurationChanged 方法，并捕捉画面方向更改事件，同时改变 TextView 中的显示。onConfigurationChanged 方法是系统设置改变之后所触发的事件，其传入的参数为 Configuration 对象。

在 AndroidManifest 的 Activity 中需要设置 Configuration 属性以及添加权限，其权限代码为：<uses-permission android:name="android.permission.CHANGE_CONFIGURATION"> </uses-permission>，详见光盘源代码/第 5 章/Sample5_20/AndroidMainifest.xml。

【代码解析】

下面要介绍的是 Sample5_20_Activity 类的实现，该类代码如下。

代码位置：见光盘源代码/第 5 章/Sample5_20/src/com/bn/chap5/configchange/Sample5_20_Activity.class。

```
1   package com.bn.chap5.configchange;
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3   public class Sample5_20_Activity extends Activity {
4       TextView tv;                                          //声明 TextView
5       Button bLandscape;                                    //横向
6       Button bPortrait;                                     //纵向
7       int screenHeight;                                     //手机屏幕高度
8       int screenWidth;                                      //手机屏幕宽度
9       public void onCreate(Bundle savedInstanceState) {
10        super.onCreate(savedInstanceState);
```

```
11          setContentView(R.layout.main);                              //设置当前界面
12          tv=(TextView)this.findViewById(R.id.TextView01);
13          bLandscape=(Button)this.findViewById(R.id.Button01);   //横向按钮
14          bPortrait=(Button)this.findViewById(R.id.Button02);    //纵向按钮
15          final Display defaultDisplay=getWindow().getWindowManager().getDefault
Display();//获取分辨率
16          screenHeight=defaultDisplay.getHeight();          //获取手机分辨率高度
17          screenWidth=defaultDisplay.getWidth();            //获取手机分辨率宽度
18          tv.setText(screenHeight+"X"+screenWidth);         //设置 TextView 显示文本
19          if(screenHeight>screenWidth){                     //纵向
20            bPortrait.setClickable(false);                  //不可单击
21            bLandscape.setClickable(true);                  //可单击
22          }else{                                            //横向
23            bPortrait.setClickable(true);                   //可单击
24            bLandscape.setClickable(false);                 //不可单击
25          }
26          bLandscape.setOnClickListener(                    //设置为横屏
27            new OnClickListener(){
28            public void onClick(View arg0) {
29            setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_
LANDSCAPE);//设置为横屏
30                      screenHeight=defaultDisplay.getHeight();//获取手机分辨率高度
31                      screenWidth=defaultDisplay.getWidth(); //获取手机分辨率宽度
32                     tv.setText(screenWidth+"X"+screenHeight);
                                                                  //设置 TextView 显示文本
33    }});
34          bPortrait.setOnClickListener(                     //设置为竖屏
35            new OnClickListener(){
36            public void onClick(View arg0) {
37
setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);//设置为竖屏
38                      screenHeight=defaultDisplay.getHeight();//获取手机分辨率高度
39                      screenWidth=defaultDisplay.getWidth(); //获取手机分辨率宽度
40                      tv.setText(screenWidth+"X"+screenHeight);
                                                                  //设置 TextView 显示文本
41    }});}
42      public void onConfigurationChanged(Configuration config){
43       if(config.orientation==Configuration.ORIENTATION_LANDSCAPE){ //横屏
44            bPortrait.setClickable(true);                         //可单击
45            bLandscape.setClickable(false);                       //不可单击
46       }else if(config.orientation==Configuration.ORIENTATION_PORTRAIT){//竖屏
47            bPortrait.setClickable(false);                        //不可单击
48            bLandscape.setClickable(true);                        //可单击
49       }
50       super.onConfigurationChanged(config);
51    }}
```

其中：

- 第 15～25 行为获取手机屏幕的分辨率，并根据手机分辨率设置按钮是否可单击。
- 第 26～41 行为设置按钮的监听器，当单击设置为横屏按钮时，将手机屏幕设置为横屏，同时改变 TextView 中显示的文本，当单击设置为 shp 按钮时，将手机屏幕设置为竖屏，同时改变 TextView 中显示的文本。
- 第 42～51 行为重写 onConfigurationChanged 方法，每当手机屏幕改变时系统自动调用该方法，同时设置按钮是否可单击。

小结

本章详细介绍了手机通信服务及手机的控制，在手机通信服务中自动调用系统拨号、上网和发送 E-mail 功能的实现，首先为主界面中的 EditText 设置监听器，单击此控件为 Linkify.addLinks()方法设置内容，实现系统拨号、上网和发送 E-mail 功能。

手机控制中的还原桌面背景，进入主界面显示单击按钮触动监听事件，并执行 Sample5_13_Activity.this.clearWallpaper()方法还原桌面的背景。通过本章的详细学习，读者对手机通信以及手机控制会有更深入的理解。

第 6 章　手机的自动服务功能

在本章将会介绍 Android 手机强大的自动服务功能，其自动服务功能包含应用程序的后台运行，时时监控是否收到短信等功能，下面将为读者逐一介绍以上手机的自动服务功能。

实例 1　自动服务的主要功能

用户在使用手机时，总是希望在听音乐的同时可以使用手机上网或者发短信。这时就需要歌曲播放程序具有后台运行的能力，在本节将会详细介绍 Service 的应用。

【实例描述】

在本实例主界面中有四个按钮，分别为开始 Service、销毁 Service、绑定 Service 以及取消绑定 Service。当单击开始 Service 时，系统调用 onCreate 方法创建 Service。当单击销毁 Service 按钮时，系统调用 onSextroy 方法销毁 Service。

单击绑定 Service 按钮，系统调用 onCreate 和 onBind 方法，创建并绑定 Service；单击取消绑定 Service 方法时，系统调用 onUnbind 方法和 onDestroy 方法，取消绑定并销毁 Service。

本实例的运行效果如图 6-1 所示。

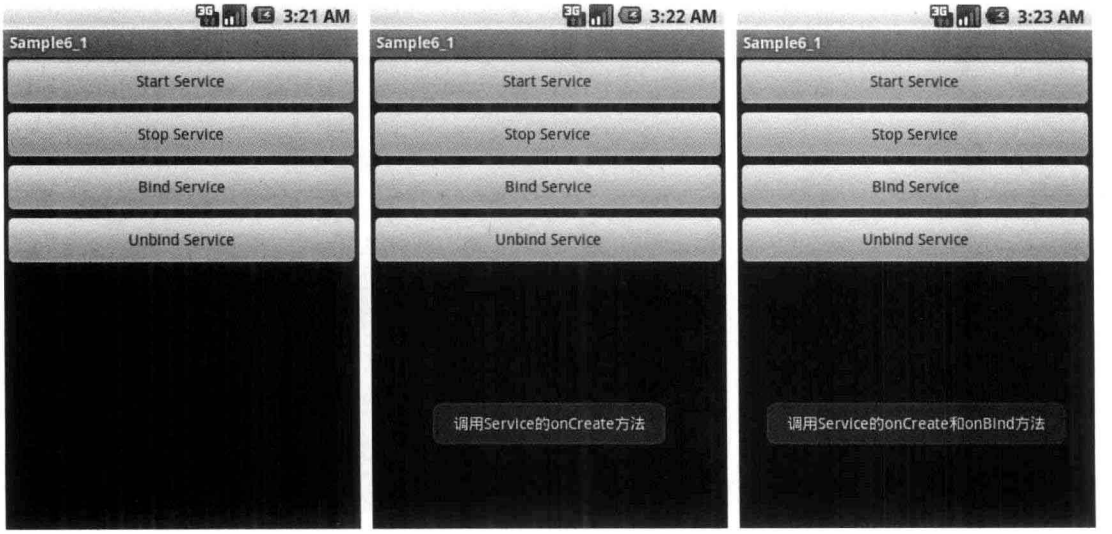


图 6-1　本实例运行效果图

提示：在图 6-1 中，第一幅图为实例开始运行时界面，第二幅图为单击 Start Service 按钮的界面，第三幅图为单击 Bind Service 按钮的界面。

【实现过程】

本实例为自己创建 Service，首先需要继承 android.app.Service 类，然后重写 Service 声明周期状态迁移过程中系统要回调的各个方法。Service 的声明周期回调方法主要有六个，分别是 onCreate、onStart、onDestroy、onBind、onUnbind 与 onRebind。

提示：这六个声明周期回调方法根据 Service 启动方式的不同，有不同的组合方式，后面会详细介绍。在这里需要特别注意的是，Service 自己是不能启动的，必须通过 Context 对象调用 startService 或 bindService 方法来启动，用这两种方法启动的 Service 声明周期是不相同的。

调用 startService 方法时，若 Service 没有启动，则首先调用该 Service 的 onCreate 方法，然后再调用其 onStart 方法。若 Service 已经启动，则会直接调用 onStart 方法。通过 startService 启动后，Service 可以通过 Context 对象调用 stopService 来关闭，也可以通过 Service 自身调用 stopSelf 或 stopSelfResult 来关闭，关闭之前将调用 onDestroy 方法。

若是通过某个 Context 调用 bindService 方法启动 Service 则有所不同，其功能为当前的 Context 对象通过一个 Service-Service，首先调用 Service 的 onCreate 方法，来初始化启动 Service，然后调用 Service 的 onBind 方法初始化绑定。如果绑定 Service 的 Context 对象被销毁时，被绑定的 Service 也会被调用 onUnbind 和 onDestroy 方法并停止运行。

【代码解析】

首先要介绍的是该实例的核心代码部分，主要介绍 AndroidManifest.xml 文件，代码如下。

代码位置：见随书光盘中源代码/第 6 章/Sample6_1/目录下的 AndroidManifest.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>                    <!--版本号和编码方式-->
2   <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3                         package="com.bn.chap6.server"
4                         android:versionCode="1"
5                 android:versionName="1.0">                  <!--版本名称-->
6       <application android:icon="@drawable/icon" android:label="@string/app_name">
7                 <activity android:name=".Sample6_1_Activity"
8                   android:label="@string/app_name">         <!--显示的程序名称-->
9             <intent-filter>
10                <action android:name="android.intent.action.MAIN" />
                                                              <!--标识程序开始-->
11                <category android:name="android.intent.category.LAUNCHER" />
12            </intent-filter>
13        </activity>
14            <service android:name=".MyService"/>            <!--注册 Service-->
15      </application>
16        <uses-sdk android:minSdkVersion="7" />              <!--设置 SDK 的值-->
17  </manifest>
```

提示：上述代码中第 14 行为注册 Service。

上述代码已经介绍了本实例的核心部分，接下来要介绍的是继承 Activity 类的开发，代码如下。

代码位置：见随书光盘中源代码/第 6 章/Sample6_1/src/com/bn/chap6/server 目录下的 Sample6_1_Activity.java。

```
1   package com.bn.chap6.server;                          //声明包
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3   public class Sample6_1_Activity extends Activity {
4       ServiceConnection sc;                             //声明 ServiceConnection 引用
5       OnClickListener listener;                         //声明 OnClickListener 引用
6       Handler hd=new Handler(){                         //创建 Handler 对象
7           public void handleMessage(Message msg){
8               switch(msg.what){
9                   case 0:
10                      Toast.makeText(                   //创建 Toast
11                          Sample6_1_Activity.this,
12                          "调用 Service 的 onCreate 和 onBind 方法",
13                           Toast.LENGTH_SHORT).show();   //显示时间
14                  break;
15                  case 1:
16                      Toast.makeText(                   //创建 Toast
17                          Sample6_1_Activity.this,
18                          "调用 Service 的 onUnbind 和 onDistroy 方法",
19                          Toast.LENGTH_SHORT).show();   //显示时间
20                  break;
21                  case 2:
22                      Toast.makeText(                   //创建 Toast
23                          Sample6_1_Activity.this,
24                          "调用 Service 的 onDistroy 方法", //显示的信息
25                          Toast.LENGTH_SHORT).show();   //显示时间
26                  break;
27                  case 3:
28                      Toast.makeText(                   //创建 Toast
29                          Sample6_1_Activity.this,
30                          "调用 Service 的 onCreate 方法",  //显示的信息
31                           Toast.LENGTH_SHORT).show();  //显示时间
32                  break;
33              }}};
34      public void onCreate(Bundle savedInstanceState) {
35          super.onCreate(savedInstanceState);
36          setContentView(R.layout.main);                //设置当前界面
37          sc=new ServiceConnection(){                   //创建对象
38              public void onServiceConnected(ComponentName name, IBinder service)
{
39              }
40              public void onServiceDisconnected(ComponentName name) {
41              }};
42          listener=new OnClickListener(){               //创建监听器对象
43              public void onClick(View v) {
44                  Intent intent=new Intent(Sample6_1_Activity.this,MyService.
class);
45                  switch(v.getId()){
46                    case R.id.Button01:                 //按下 Start Service 按钮
47                      startService(intent);             //启动 Service
48                      hd.sendEmptyMessage(3);           //发送消息
49                    break;
50                    case R.id.Button02:                 //按下 Stop Service 按钮
51                       stopService(intent);             //销毁 Service
52                       hd.sendEmptyMessage(2);          //发送消息
53                   break;
54                    case R.id.Button03:                 //按下 Bind Service 按钮
55                     bindService(intent,sc,BIND_AUTO_CREATE);//绑定 Service
56                     hd.sendEmptyMessage(0);            //发送消息
57                    break;
```

```
58                      case R.id.Button04:                    //按下 Unbind Service 按钮
59                        unbindService(sc);                   //取消绑定 Service
60                        hd.sendEmptyMessage(1);              //发送消息
61                      break;
62                    }}};
63          this.findViewById(R.id.Button01).setOnClickListener(listener);
                                                               //添加监听器
64          this.findViewById(R.id.Button02).setOnClickListener(listener);
                                                               //添加监听器
65          this.findViewById(R.id.Button03).setOnClickListener(listener);
                                                               //添加监听器
66          this.findViewById(R.id.Button04).setOnClickListener(listener);
                                                               //添加监听器
67    }}
```

其中：

- 第 4～33 行首先声明 ServiceConnection、OnClickListener 应用，并创建 Handler 对象，在重写的 handleMessage 方法中，根据所接收消息的编号，弹出相应的 Toast 提示。
- 第 34～62 行为创建 ServiceConnection、OnClickListene 对象，在 onClick 方法中为每一个 Button 按钮添加单击事件，依次为启动、销毁、绑定和取消绑定 Service。
- 第 63～66 行为给每一个按钮注册监听器。

上述代码介绍的是本实例的主控制类 Sample6_1_Activity 的开发，接下来要介绍的是创建并继承 Service 类的开发，代码如下。

代码位置：见随书光盘中源代码/第 6 章/Sample6_1/src/com/bn/chap6/server 目录下的 MyService.java。

```
1   package com.bn.chap6.server;
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3   public class MyService extends Service{
4       public IBinder onBind(Intent arg0) {                   //重写 onBind 方法
5           Log.d("MyService", "=========onBind=========");     //打印信息
6       ~   return null;
7       }
8       public boolean onUnbind(Intent arg0){                  //重写 onUnbind 方法
9           Log.d("MyService", "=========onUnbind=========");   //打印信息
10          return super.onUnbind(arg0);
11      }
12      public void onRebind(Intent arg0){                     //重写 onRebind 方法
13          super.onRebind(arg0);
14          Log.d("MyService", "=========onRebind=========");   //打印信息
15      }
16      public void onCreate(){                                //重写 onCreate 方法
17          super.onCreate();
18          Log.d("MyService", "=========onCreate=========");   //打印信息
19      }
20      public void onDestroy(){                               //重写 onDestroy 方法
21          super.onDestroy();
22          Log.d("MyService", "=========onDestroy=========");  //打印信息
23  }}
```

提示：上述代码为继承 Service 类，在该类中必须重写 onBind 方法。其他方法可根据需要进行重写。在本实例中重写了 onBind、onUnbind、onRebind、onCreate、onDestroy 方法，在每个方法中增加打印，以方便观察。

实例 2　系统服务的开始与停止

经过上面小节对系统服务的一些理论知识介绍，读者对系统服务应该有了大概的了解。接下来在本节将要详细介绍系统服务的开始与停止。

【实例描述】

在该软件主界面中有两个按钮，分别为启动 Service 和关闭 Service，当单击启动 Service 按钮时，开启 Service，同时启动后台一个线程，每过 1 秒钟增加 1，开始记录服务运行的时间；当单击关闭 Service 按钮时，关闭 Service，关闭线程。

本实例的运行效果图如图 6-2 所示。

图 6-2　启动和关闭 Service

提示：在图 6-2 中依次表示的是单击启动 Service 按钮的界面，以及单击关闭 Service 按钮的界面。

【实现过程】

主程序中存在两个按钮，当单击启动 Service 按钮时，系统首先调用 startService 方法，通过发送 Intent 来完成 Service 的启动，后者为关闭 Service。在 Intent 对象中，第一个参数为当前的 Activity(Sample6_2_Activity.java)，第二个参数为服务类（MyService. Java）。

当打开 Service 时，为了证明服务仍在运行过程中，开启线程记录 Service 运行的时间，当单击关闭按钮时，终止线程。

【代码解析】

本部分首先介绍的是 AndroidManifest.xml 的开发，在该 xml 文件中需要声明 Service 的名称。同时设定 exported 的属性为 ture，从而确定该服务可以被其他程序访问，代码如下。

代码位置：见随书光盘中源代码/第 6 章/Sample6_2 目录下的 AndroidManifest.xml。

```
<?xml version="1.0" encoding="utf-8"?>                    <!--版本号和编码方式-->
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
                        package="com.bn.chap6.serben"
                        android:versionCode="1"
      android:versionName="1.0">                          <!--版本名称-->
    <application android:icon="@drawable/icon" android:label="@string/app_name">
              <activity android:name=".Sample6_2_Activity"
               android:label="@string/app_name">          <!--显示的程序名称-->
          <intent-filter>
             <action android:name="android.intent.action.MAIN" />
                                                          <!--显示的程序名称-->
             <category android:name="android.intent.category.LAUNCHER" />
          </intent-filter>
       </activity>
           <service
                   android:name=".MyService"
                   android:exported="true"
            android:process=":remote"></service>
    </application>
</manifest>
```

提示：上述代码中第 14～17 行为声明 MyService 和设定 exported 属性值为 true。

上述 xml 代码介绍的是本实例的 AndroidManifest.xml 的开发，接下来要介绍的是继承 Activity 类的开发，代码如下。

代码位置：见随书光盘中源代码/第 6 章/Sample6_2/src/com/bn/chap6/serben 目录下的 Sample6_2_Activity.java。

```
package com.bn.chap6.serben;                                    //声明包
……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
public class Sample6_2_Activity extends Activity {
    Button bStart;                                              //开始按钮
    Button bStop;                                               //关闭按钮
    public void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);                      //调用父类
       setContentView(R.layout.main);                           //跳到主界面
       bStart=(Button)this.findViewById(R.id.Button01);         //创建对象
       bStop=(Button)this.findViewById(R.id.Button02);
       bStart.setOnClickListener(                               //开始按钮监听器
         new OnClickListener(){
                 public void onClick(View v) {                  //重写的方法
                     //创建 Intent 对象，并指定为 MyService 服务
                     Intent    intent=new    Intent(Sample6_2_Activity.this,
MyService.class);
                     intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                                                                //创建新的 Task 方式
                     startService(intent);                      //启动 Service
                     Toast.makeText(                            //创建 Toast
                         Sample6_2_Activity.this,
                          "Service 已启动",
                         Toast.LENGTH_SHORT).show();    //显示 Toast
    }});
       bStop.setOnClickListener(                                //关闭按钮监听器
         new OnClickListener(){
                 public void onClick(View v) {                  //重写的方法
                     //创建 Intent 对象，并指定为 MyService 服务
                     Intent    intent=new    Intent(Sample6_2_Activity.this,
```

```
MyService.class);
    28                          stopService(intent);                //关闭 Service
    29                          Toast.makeText(                     //创建 Toast
    30                              Sample6_2_Activity.this,
    31                              "Service 已关闭",
    32                              Toast.LENGTH_SHORT).show();     //显示 Toast
    33  }});}}
```

其中：

- 第 4～10 行为声明 Button 引用并创建对象。
- 第 11～22 行为开始按钮的监听器，在 onClick 方法中创建 Intent 对象，并设定其为 Service 服务，然后启动 Service，并用 Toast 提示用户该 Service 已启动。
- 第 23～33 行为关闭按钮的监听器，在该 onClick 方法中同样创建 Intent 对象，并设定其为 Service 服务，然后关闭 Service，使用 Toast 提示用户该 Service 已关闭。

上述代码介绍了本程序的 Activity 类的开发，接下来介绍的是继承 Service 类，并重写 onStart、onCreate、onDestroy 与 onBind 方法的开发，代码如下。

代码位置：见随书光盘中源代码/第 6 章/Sample6_2/src/com/bn/chap6/serben 目录下的 MyService.java。

```
1   package com.bn.chap6.serben;                                    //声明包
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3   public class MyService extends Service{
4       static int count=0;                                         //声明计数器
5       boolean flag=true;                                          //声明标志位
6       public void onStart(Intent intent,int startId){ //重写 onStart 方法
7           new Thread(){                                           //创建一个线程
8               public void run(){
9                   while(flag){
10                      count++;                                    //计数器自加
11                      Log.d("count", count+"");                   //打印计数器
12                      try{
13                          Thread.sleep(1000);                     //睡眠 1 秒钟
14                      }catch(Exception e){e.printStackTrace();}   //打印异常
15                  }}}.start();                                    //启动线程
16          super.onStart(intent, startId);                         //调用父类方法
17      }
18      public void onCreate(){                                     //重写 onCreate 方法
19          super.onCreate();
20      }
21      public void onDestroy(){                                    //重写 onDestroy 方法
22          flag=false;                                             //将线程标志位设为 false
23          super.onDestroy();
24      }
25      public IBinder onBind(Intent intent) {                      //重写 onBind 方法
26          return null;
27  }}
```

其中：

- 第 4～17 行为声明计数器和标志位，重写 onStart 方法。在 inStart 方法中创建一个线程，每当 Service 启动时，开启该线程，每过 1 秒钟计数器增加 1，并在后台打印。
- 第 18～27 行为重写 onCreate、onDestroy 和 onBind 方法，在 onDestroy 方法中将标志位设为 false。每当 Service 被关闭时，则调用 onDestroy 方法，同时关闭线程。

实例 3　提醒用户收到短信

接发短信是手机最基本的功能，在 Android 手机中还能够自定义接收短信后的提示，在本节中将介绍如何运用 Service 实现上述功能。

【实例描述】

本软件主界面构造简单，启动本软件时在主界面提示等待接收短信。当接收到短信时，将短信中发信人的电话号码和短信内容分别获取，使用 Toast 显示在主界面中。

本实例的运行效果如图 6-3 所示。

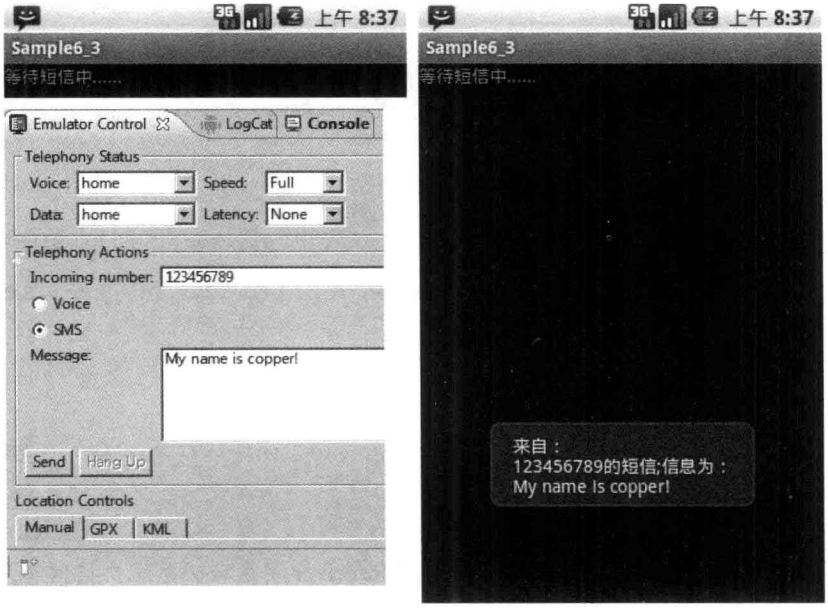


图 6-3　发送短信以及接收短信效果图

提示： 图 6-3 为本实例开始运行时效果和发送短信的界面，其中发送短信的界面在 Eclipse 中的“Emulator Control”窗口中，在该窗口中需要填写电话号码，选择 SMS 并填写短信信息，最后单击“发送”按钮，在程序中就能够捕捉到短信，将发信人电话号码和短信信息提取使用 Toast 显示，如图 6-3 右图所示。

【实现过程】

在本软件中最重要的一点为如何在系统中注册一个 BroadcastReceiver 对象，在后台监听短信事件，最后将短信的信息使用 Toast 显示出来。当继承 BroadcastReceiver 类时，需要重写 onReceiver()方法，才能够确保可以捕捉事件。

在判断是否为短信事件时，需要加上 Android.provider.Telephony.SMS_RECEIVER 作为过滤条件，若判断短信的内容不为空，则使用 Bundle.get()方法，将带有 pdus 字符串特征的对象取出，取出短信信息重新组装，使用 Toast 显示在主界面上。

【代码解析】

在本部分将详细介绍该实例的核心代码，首先介绍的是 AdoroidManifest.xml 的开发，代码如下。

代码位置：见随书光盘中源代码/第 6 章/Sample6_3 目录下的 AndroidManifest.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>                    <!--版本号和编码方式-->
2   <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3                 package="com.bn.chap6.tips"
4                     android:versionCode="1"
5       android:versionName="1.0">                            <!--版本名称-->
6     <application android:icon="@drawable/icon" android:label="@string/app_name">
7                     <activity android:name=".Sample6_3_Activity"
8                 android:label="@string/app_name">          <!--显示的程序名称-->
9           <intent-filter>
10             <action android:name="android.intent.action.MAIN" />
                                                              <!--标识程序开始-->
11              <category android:name="android.intent.category.LAUNCHER" />
12          </intent-filter>
13        </activity>
14      <!-- 创建 receive 聆听系统广播信息 -->
15      <receiver android:name=".MyReceiver">
16          <!-- 设置要捕捉的信息名称为 provider 中 Telephony.SMS_RECEIVED -->
17          <intent-filter>
18              <action android:name="android.provider.Telephony.SMS_
RECEIVED"></action>
19          </intent-filter>
20      </receiver>
21    </application>
22    <!-- 设置权限 -->
23     <uses-permission android:name="android.permission.RECEIVE_SMS">
</uses-permission>
24  </manifest>
```

提示：上述代码中第 14～21 行为声明 Receiver 用来聆听系统广播信息，同时设定 android.provider.Telephony.SMS_RECEIVED，用来过滤短信事件，第 23 行为设置接收短信的权限。

上述代码介绍的是对 AdoroidManifest.xml 开发，接下来要介绍的是继承 Activity 类的开发，代码如下。

代码位置：见随书光盘中源代码/第 6 章/Sample6_3/src/com/bn/chap6/tips 目录下的 Sample6_3_Activity.java。

```
1   package com.bn.chap6.tips;                                //声明包
2   import android.app.Activity;                              //导入包
3   import android.os.Bundle;                                 //导入包
4   public class Sample6_3_Activity extends Activity {
5      public void onCreate(Bundle savedInstanceState) {      //重写的方法
6         super.onCreate(savedInstanceState);                 //调用父类
7         setContentView(R.layout.main);                      //设置主界面
8   }}
```

提示：上述代码实现的功能为设置主界面为 main.xml。

上述代码介绍的是本程序的 Activity 类的开发，接下来介绍的是 MyReceiver 类的实现，该类继承自 BroadcastReceiver 类，重写 onReceiver()方法，在该方法中实现短信信息的处理，代码如下。

代码位置：见随书光盘中源代码/第 6 章/Sample6_3/src/com/bn/chap6/tips 目录下的 MyReceiver.java。

```
1   package com.bn.chap6.tips;                                    //声明包
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3   public class MyReceiver extends BroadcastReceiver{
4       public void onReceive(Context context, Intent intent) {
                                                                  //重写的 onReceiver 方法
5           if(intent.getAction().equals("android.provider.Telephony.SMS_
RECEIVED")){//判断是否为短信
6               StringBuilder sb=new StringBuilder();
7               Bundle bundle=intent.getExtras();          //创建 Bundle 对象，获取信息
8               if(bundle!=null){
9                   Object[] obj=(Object[])bundle.get("pdus"); //获取短信信息
10                  SmsMessage[] sm=new SmsMessage[obj.length];
                                                          //创建 SmsMessage 数组对象
11                  int length=obj.length;                      //获取长度
12                  for(int i=0;i<length;i++){
13                      sm[i]=SmsMessage.createFromPdu((byte[])obj[i]);
                                                          //获取短信信息
14                  }
15                  for(int i=0;i<length;i++){
16                      sb.append("来自: \n");      //组装信息
17                      sb.append(sm[i].getDisplayOriginatingAddress());
                                                          //电话号码
18                      sb.append("的短信;");
19                      sb.append("信息为: \n");
20                      sb.append(sm[i].getMessageBody()); //短信内容
21                  }
22                  Toast.makeText(                         //创建 Toast 对象
23                          context,
24                          sb.toString().trim(),
25                          Toast.LENGTH_SHORT              //显示时间
26                          ).show();                       //显示 Toast
27                  Intent  tempIntent=new  Intent(context,Sample6_3_Activity.
class);//创建 Intent 对象
28                  tempIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                                                          //设置新的 task
29                  context.startActivity(tempIntent);      //启动 Actvity
30  }}}}
```

其中：

- 第 5～14 行首先判断是否收到短信事件，若收到的信息为短信事件，则创建 Intent 对象，并获取信息，若获取的信息不为空，则获取短信信息，创建 SmsMessage 数组对象，最后需要将短信信息存储到数组中。
- 第 15～26 行为重新组装短信信息为 StringBuilder 对象，并使用 Toast 显示。
- 第 27～29 行为创建 Intent 对象，设置新的 task 返回值 Activity。

实例 4　查看手机电池剩余电量

手机待机时间的长短一直是用户关注的问题，在本节将详细介绍如何查看手机当前的剩余

电量。

【实例描述】

本软件实现简单，在本软件主界面中主要包含一个 Button 按钮控件和一个 TextView 控件。当单击按钮时，系统获取手机当前的剩余电量的百分比，并将电量使用 Toast 显示出来，同时在 TextView 中填写剩余电量的百分比。

本实例的运行效果如图 6-4 所示。

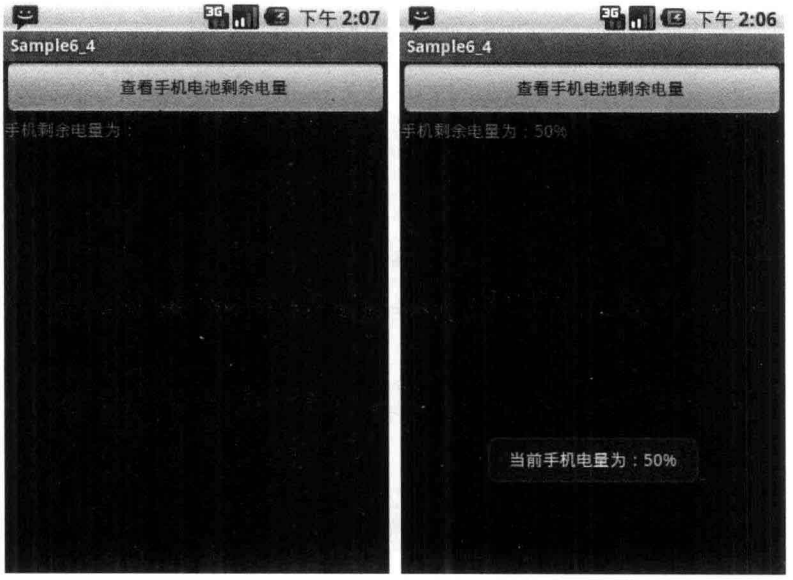


图 6-4　查看手机电池剩余电量效果图

提示：在图 6-4 中依次表示的是开始运行的界面，在主界面单击按钮后获取剩余电量的效果图。

【实现过程】

本实例通过创建 BroadcastReceiver 对象，当单击 Button 按钮时出发，向系统注册该对象。通过 IntentFilter 设置通知 BroadcastReceiver 过滤事件为 Intent.ACTION_BATTERY_CHANGED。当获取手机剩余电量时，弹出 Toast 提示当前剩余电量，通过 Handler 对象发送消息，更改 TextView 内容。

【代码解析】

在本部分将详细介绍该实例的核心代码，在 Sample6_4_Activity 类中首先创建 BroadcastReceiver 对象指定需要过滤的事件为 Intent.ACTION_BATTERY_CHANGED，其次创建 Handler 对象，在 onCreate()方法中为 Button 添加监听器，代码如下。

代码位置：见随书光盘中源代码/第 6 章/Sample6_4/src/com/bn/chap6/battery 目录下的 Sample6_4_Activity.java。

```
1   package com.bn.chap6.battery;                          //声明包
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
```

```
3   public class Sample6_4_Activity extends Activity {
4       int level;                                              //当前电池电量
5       int scale;                                              //手机总电量
6       Button button;                                          //查看按钮
7       BroadcastReceiver br=new BroadcastReceiver(){           //内部类
8           public void onReceive(Context arg0, Intent arg1) {
9               String s=arg1.getAction();                      //获取 Action
10              if(Intent.ACTION_BATTERY_CHANGED.equals(s)){
11                  level=arg1.getIntExtra("level", 0);     //获取当前电量大小
12                  scale=arg1.getIntExtra("scale", 100);   //手机的总电量大小
13                  hd.sendEmptyMessage(0);                 //发送消息改变 TextView 值
14                  Toast.makeText(                         //创建 Toast 对象
15                          Sample6_4_Activity.this,
16                          "当前手机电量为："+level*100/scale+"%", //转化为百分比
17                          Toast.LENGTH_SHORT              //Toast 显示的时间
18                          ).show();
19              }}};
20      Handler hd=new Handler(){                               //创建 Handler 对象
21          public void handleMessage(Message msg){             //重写的方法
22              switch(msg.what){
23                  case 0:
24                      TextView tv=(TextView)Sample6_4_Activity.this.findView
ById(R.id.TextView02);
25                      tv.setText(level*100/scale+"%");    //设置 TextView 内容
26                  break;
27              }}};
28      public void onCreate(Bundle savedInstanceState) {       //重写的 onCreate 方法
29          super.onCreate(savedInstanceState);
30          setContentView(R.layout.main);                      //设置主界面为 main.xml
31          button=(Button)this.findViewById(R.id.Button01);    //创建 Button 对象
32          button.setOnClickListener(                          //为 Button 添加监听器
33              new OnClickListener(){
34                  public void onClick(View v) {               //重写的方法
35                      registerReceiver(
36                          //注册访问电池剩余量的系统 BroadcastReceiver
37                          br,new IntentFilter(Intent.ACTION_BATTERY_
CHANGED)
38  );}});}}
```

其中：

- 第 1～19 行为成员变量的声明，创建 BroadcastReceiver 对象，在重写的 onReceive()方法中，首先获取 Action 事件，当判断该事件为 Intent.ACTION_BATTERY_CHANGED 时，获取手机当前电量以及总电量，发送消息更改 TextView 内容，并使用 Toast 提示手机剩余电量百分比。
- 第 20～27 行为创建 Handler 对象，在重写的 handleMessage()方法中捕捉信息编号为 0 的消息，其完成的业务为更改 TextView 内容。
- 第 28～38 行为设置当前主界面，并在 Button 监听器内注册访问电池剩余电量的系统 BroadcastReceiver。

实例 5　接收到短信时界面切换显示短信消息

Android 手机在接收到短信时，不仅仅可以提醒用户收到一条短信，还可以自定义实现阅读短信信息、将短信发送人以及短信的内容显示到自定义界面上。在本节将详细介绍如何实现

上述功能。

【实例描述】

本软件分为两个界面，首先进入的是主界面，在主界面等待接收短信。当手机接收到短信时，将短信的信息重新组合，获取发件人电话和短信内容。然后发送 Intent 返回 Activity，并判断 bundle 是否为空，若不为空，则切换界面进入短信信息界面，并在该界面显示短信的详细信息。

本实例的运行效果如图 6-5 所示。

图 6-5　本实例运行效果图

提示：在图 6-5 中依次表示的是开始运行时的主界面、短信信息界面和发送短信的 Emulator Control 窗口，在该窗口中可以填写电话号码和短信内容，并虚拟发送短信。

【实现过程】

本软件在后台创建 BroadcastReceiver 对象，用于接收短信。当收到短信后，将短信中发信人的电话号码与短信内容组装并使用 Bundle 封装。通过 Intent 返回 Activity，在 onCreate 方法中判断接收到的 Bundle 是否为空，若不为空则获取封装的内容，并将其分为电话号码和短信内容显示在短信信息界面中。

【代码解析】

在本部分会详细介绍本程序的实现过程。首先介绍的是 AndroidManifest.xml 的开发，代码如下。

代码位置：见随书光盘中源代码/第 6 章/Sample6_5 目录下的 AndroidManifest.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>                    <!--版本号和编码方式-->
2   <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3                         package="com.bn.chap6.chd"
4                     android:versionCode="1"
5                 android:versionName="1.0">                  <!--版本名称-->
6       <application android:icon="@drawable/icon" android:label="@string/app_name">
7                     <activity android:name=".Sample6_5_Activity"
```

```
8                android:label="@string/app_name">          <!--显示的程序名称-->
9            <intent-filter>
10              <action android:name="android.intent.action.MAIN" />
                                                            <!--标识程序开始-->
11              <category android:name="android.intent.category.LAUNCHER" />
12           </intent-filter>
13        </activity>
14        <!-- 创建 Receive 聆听系统广播信息 -->
15          <receiver android:name=".MyReceiver6_5">
16              <intent-filter>
17                  <action android:name="android.provider.Telephony.SMS_
RECEIVED"></action>
18              </intent-filter>
19          </receiver>
20     </application>
21      <!-- 设置权限 -->
22      <uses-permission android:name="android.permission.RECEIVE_SMS"></uses-
permission>
23  </manifest>
```

提示：上述代码中第 14～19 行为声明 Receiver，用于聆听系统广播信息，将过滤事件设置为 android.provider.Telephony.SMS_RECEIVED，第 22 行为设置接收短信的权限，确保手机可以接收短信。

上述 AndroidManifest.xml 的代码主要是声明允许查收消息的权限以及对 Receiver 设置，接下来介绍的是继承 Activity 类的开发，在该类中主要是对界面的切换以及实现短信消息的读取，代码如下。

代码位置：见随书光盘中源代码/第 6 章/Sample6_5/src/com/bn/chap6/chd 目录下的 Sample6_5_Activity.java。

```
1   package com.bn.chap6.chd;                                    //声明包
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3   public class Sample6_5_Activity extends Activity {
4      Handler hd=new Handler(){                                 //创建 Handler 对象
5       public void handleMessage(Message msg){                  //重写的方法
6           switch(msg.what){
7               case 0:                                         //编号为 0
8                   Bundle b=msg.getData();                     //获取数据
9                   String tempMsg=(String) b.get("xx");        //获取信息
10                  gotoXX(tempMsg);                            //进入短信信息界面
11              break;
12     }}};
13     public void onCreate(Bundle savedInstanceState) {        //重写的 onCreate 方法
14        super.onCreate(savedInstanceState);                   //调用父类
15      Bundle bundle=this.getIntent().getExtras();             //取得短信发来 bundle
16      if(bundle!=null){
17          String tempMsg=bundle.getString("change");          //获取信息
18          Bundle b=new Bundle();
19          b.putString("xx", tempMsg);                         //绑定信息
20          Message m=new Message();                            //创建 Message 对象
21          m.what=0;                                           //设置编号
22          m.setData(b);
23          hd.sendMessage(m);                                  //发送消息
24      }else{
25           setContentView(R.layout.main);                     //设置当前界面为主界面
26      }
```

```
27 }
28     public void gotoXX(String msg){                          //跳转到某一界面
29       setContentView(R.layout.xx);                           //设置当前界面
30       EditText et=(EditText)this.findViewById(R.id.EditText01);    //创建对象
31       EditText ett=(EditText)this.findViewById(R.id.EditText02);
32       String[] tempMsg=msg.split("\\|");                     //切分字符串
33       et.setText(tempMsg[0]);                                //设置电话号码
34       ett.setText(tempMsg[1]);                               //设置短信内容
35  } }
```

其中：

- 第 4～12 行为创建 Handler 对象，接收信息，并完成界面间的切换。
- 第 13～27 行为获取发送来的 bundle，若 bundle 为空则进入主界面，若不为空，取出短信信息，创建 Bundle 对象并绑定信息，通过 Handler 发送消息。
- 第 28～35 行为短信信息界面，在该方法中首先设置当前界面，然后创建 EditText 对象，切分获取的短信信息，并将电话号码和短信内容设置在该界面的 EditText 中。

上述代码介绍的是本程序的 Activity 类的开发，接下来介绍的是创建并继承 BroadcastReceiver 类。在该类中对接收信息过滤，将收到的短信内容重新组装，通过 Intent 返回至 Activity 的开发，代码如下。

代码位置：见随书光盘中源代码/第 6 章/Sample6_5/src/com/bn/chap6/chd 目录下的 MyReceiver6_5.java。

```
1   package com.bn.chap6.chd;                                  //声明包
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3   public class MyReceiver6_5 extends BroadcastReceiver{     //创建类
4       public void onReceive(Context context, Intent intent) {
5
if(intent.getAction().equals("android.provider.Telephony.SMS_RECEIVED")){
6               StringBuilder sb=new StringBuilder();          //创建对象
7               Bundle bundle=intent.getExtras();          //创建 Bundle 对象，获取信息
8               if(bundle!=null){
9                   Object[] obj=(Object[])bundle.get("pdus"); //获取短信信息
10                  SmsMessage[] sm=new SmsMessage[obj.length];
                                                           //创建 SmsMessage 对象数组
11                  int length=obj.length;                 //短信个数
12                  for(int i=0;i<length;i++){
13                      sm[i]=SmsMessage.createFromPdu((byte[])obj[i]);
14                  }
15                  for(int i=0;i<length;i++){
16                      sb.append(sm[i].getDisplayOriginatingAddress());
                                                           //电话号码
17                      sb.append("|");
18                      sb.append(sm[i].getMessageBody());     //短信信息
19                  }
20                  Toast.makeText(                        //创建 Toast 对象
21                          context,
22                          "接收到一条短信！",
23                          Toast.LENGTH_SHORT             //Toast 显示时间
24                          ).show();
25                  Intent  tempIntent=new  Intent(context,Sample6_5_Activity.
class);//创建 Intent 对象
26                  Bundle myBundle=new Bundle();          //创建 Bundle 对象
27                  myBundle.putString("change", sb.toString().trim());
28                  tempIntent.putExtras(myBundle);        //绑定 Bundle
29                  tempIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
```

```
                                                          //设置新的 task
30                    context.startActivity(tempIntent);      //启动 Actvity
31   }}}}
```

其中：

- 第 5～14 行为过滤 Action 为 android.provider.Telephony.SMS_RECEIVED 的事件，当条件成立时，创建 StringBuilder 和 Bundle 对象，并获取短信信息。
- 第 15～24 行为重新组装获取的短信信息，然后创建 Toast，显示提示信息。
- 第 25～31 行为创建 Intent 和 Bundle 对象，使用 Bundle 绑定信息，设置新的 task，最后将绑定好的信息通过 Intent 启动 Aactivity。

实例 6　通过后台定时发送提示

在程序运行的过程中，有时需要不断地向前台发送消息，发送消息后还需要用 Activity 对需要的消息进行捕捉，这时需要自定义的 Broadcast。本节将对其进行详细介绍。

【实例描述】

本软件框架简单，在主界面含有两个 Button 按钮，一个为启动 Service 按钮，另一个为关闭 Service 按钮。当单击启动按钮后，启动系统服务，同时启动线程，每隔 10 秒钟发送一次广播消息，而主程序中会捕捉到该信息，并将该信息使用 Toast 显示在主界面上。

在主界面单击关闭按钮后，停止系统服务，关闭线程。本实例的运行效果如图 6-6 所示。

图 6-6　本实例运行效果图

提示：在图 6-6 中依次表示的是启动 Service，捕获发送的广播消息，关闭 Service。

【实现过程】

在本实例中，通过在继承 Service 类的 onStart 方法中设置一个线程，使该线程每隔 10 秒钟通过 Intent 封装信息，然后向系统发送一次广播。而继承 BroadcastReceiver 类的

MyBroadcastReceiver 类，则通过重写 onReceiver()方法，在方法中拆解封装，然后接收传给自己的信息。

【代码解析】

在本部分会详细介绍本程序的实现过程。首先介绍的是 AndroidManifest.xml 的开发，代码如下。

代码位置：见随书光盘中源代码/第 6 章/Sample6_6 目录下的 AndroidManifest.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>                    <!--版本号和编码方式-->
2   <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3                         package="com.bn.chap6.bbs"
4                             android:versionCode="1"
5           android:versionName="1.0">                        <!--版本名称-->
6      <application  android:icon="@drawable/icon"  android:label="@string/app_
name">
7           <activity android:name=".Sample6_6_Activity"
8                android:label="@string/app_name">      <!--显示的程序名称-->
9            <intent-filter>
10              <action android:name="android.intent.action.MAIN" />
                                                          <!--显示的程序名称-->
11              <category android:name="android.intent.category.LAUNCHER" />
12           </intent-filter>
13        </activity>
14          <service
15              android:name=".MyReceiver6_6"
16              android:exported="true"
17              android:process=":remote">
18          </service>
19    </application>
20  </manifest>
```

提示：上述代码中第 14～18 行为声明 Servicer，并设置 exported 属性为 true，使其他 Activity 也可以访问该 Service。

上述代码主要介绍的是 AndroidManifest.xml 的开发，接下来将要介绍的是继承 Activity 类的开发，代码如下。

代码位置：见随书光盘中源代码/第 6 章/Sample6_6/src/com/bn/chap6/bbs 目录下的 Sample6_6_Activity.java。

```
1   package com.bn.chap6.bbs;                                    //声明包
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3   public class Sample6_6_Activity extends Activity {
4       Button bStart;                                           //启动按钮
5       Button bStop;                                            //停止按钮
6      public void onCreate(Bundle savedInstanceState) {
7         super.onCreate(savedInstanceState);                    //调用父类
8         setContentView(R.layout.main);                         //设置当前界面
9         bStart=(Button)this.findViewById(R.id.Button01);
10        bStop=(Button)this.findViewById(R.id.Button02);        //得到按钮引用
11        bStart.setOnClickListener(                             //启动按钮监听器
12           new OnClickListener(){
13              public void onClick(View v) {                   //重写的方法
14                 Intent intent=new Intent(Sample6_6_Activity.this,
MyReceiver6_6.class);//创建对象
15                 intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
```

```
16                  startService(intent);                    //启动 Service 服务
17                  Toast.makeText(                          //创建 Toast 对象
18                          Sample6_6_Activity.this,
19                          "成功启动 Service 服务",
20                          Toast.LENGTH_SHORT).show();      //显示 Toast
21                  }});
22          bStop.setOnClickListener(                        //停止按钮监听器
23            new OnClickListener(){
24                public void onClick(View v) {              //重写的方法
25                  Intent intent=new Intent(Sample6_6_Activity.this,
MyReceiver6_6.class);//创建对象
26                  if(stopService(intent)==true){           //若为 true
27                      Toast.makeText(                      //创建 Toast 对象
28                              Sample6_6_Activity.this,
29                              "成功停止 Service 服务",
30                              Toast.LENGTH_SHORT).show();    //显示 Toast
31                  }else{
32                      Toast.makeText(                          //创建 Toast 对象
33                              Sample6_6_Activity.this,
34                              "停止 Service 服务失败",
35                              Toast.LENGTH_SHORT).show();    //显示 Toast
36  }}}});}
37     public void onResume(){                               //重写 onResume 方法
38      super.onResume();
39      try{
40          IntentFilter intentFilter=new IntentFilter(MyReceiver6_6.MSG);
                                                              //自定义过滤信息
41          MyBroadcastReceiver mReceiver=new MyBroadcastReceiver();//创建对象
42          registerReceiver(mReceiver,intentFilter);         //注册 Receiver
43      }catch(Exception e){e.printStackTrace();}             //打印异常
44     }
45     public void onPause(){                                 //重写的方法
46      super.onPause();
47      MyBroadcastReceiver mReceiver=new MyBroadcastReceiver();     //创建对象
48      unregisterReceiver(mReceiver);                        //接触注册的 Receiver
49  }}
```

其中：

- 第 4～10 行为创建 Button 对象。
- 第 11～21 行表示对开始按钮添加监听器，在该监听器内创建 Intent 对象，用于启动 Service 服务，当启动成功后，使用 Toast 显示提示信息。
- 第 22～36 行表示对停止按钮添加监听器，首先在该监听器内创建 Intent 对象，然后判断是否已停止 Service 服务，若停止则提示停止 Service 服务失败，否则提示成功停止 Service 服务。
- 第 37～44 行表示继承 Activity 类需要重写的 onResume 方法。
- 第 44～59 行表示继承 Activity 类需要重写的 onPause 方法。

上述代码已经介绍了本程序 Activity 的开发，接下来介绍的是继承 Service 类的开发，代码如下。

代码位置：见随书光盘中源代码/第 6 章/Sample6_6/src/com/bn/chap6/bbs 目录下的 MyReceiver6_6.java。

```
1   package com.bn.chap6.bbs;                                 //声明包
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3   public class MyReceiver6_6 extends Service{
```

```
4       int count=0;                                          //计数标志位
5       static final String MSG="message";                    //消息字符串
6       boolean flag=true;                                    //进程运行标志位
7       public void onStart(Intent intent,int startId){
8           new Thread(){                                     //创建线程
9               public void run(){                            //重写的方法
10                  while(flag){
11                      count++;                              //计数器自加
12                      if(count==10){
13                          Intent i=new Intent(MSG);         //创建 Intent 对象
14                          i.putExtra("msg",MSG);            //封装参数
15                          sendBroadcast(i);                 //发送广播
16                          count=0;                          //计数器设为 0
17                      }
18                      try{
19                          Thread.sleep(1000);               //线程睡眠 1 秒钟
20                      }catch(Exception e){e.printStackTrace();    //打印异常
21                      }}}
22          }.start();                                        //启动线程
23          super.onStart(intent, startId);
24      }
25      public IBinder onBind(Intent intent) {                //重写 onBind 方法
26          return null;
27      }
28      public void onCreate(){                               //重写 onCreate 方法
29          super.onCreate();                                 //调用父类
30      }
31      public void onDestroy(){                              //重写 onDestroy 方法
32          flag=false;                                       //将标志位设为 false
33          super.onDestroy();
34  }}
```

其中：

- 第 4～24 行为重写 onStart()方法，在该方法中创建 Thread，每隔 10 秒钟使用 Intent 封装参数，向系统发送一次广播。
- 第 25～30 行为重写 onBind()和 onCreate()方法，其中 onBind()方法是必须重写的。
- 第 31～34 行为重写 onDestroy()方法，在该方法中将标志位设为 false。

上述代码介绍的是 Service 类的开发，接下来介绍的是继承 BoradcastReceiver 类的开发，代码如下。

代码位置：见随书光盘中源代码/第 6 章/Sample6_6/src/com/bn/chap6/bbs 目录下的 MyBroadcastReceiver.java。

```
1   package com.bn.chap6.bbs;                                 //声明包
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3   public class MyBroadcastReceiver extends BroadcastReceiver{
4       public void onReceive(Context context, Intent intent) {
                                                              //重写 onReceive 方法
5           try{
6               Bundle bundle=intent.getExtras();             //创建 bundle
7               String msg=bundle.getString("msg");           //获取信息
8               Toast.makeText(                               //创建 Toast 对象
9                       context,
10                      "这是来自后台服务的消息："+msg,          //提示信息
11                      Toast.LENGTH_SHORT).show();           //显示 Toast
12          }catch(Exception e){e.printStackTrace();}         //打印异常
13  }}
```

提示：上述代码中主要完成的任务为获取后台发送的信息，并将信息使用 Toast 显示在主界面中。

实例 7　短信群发功能的实现

当需要下达通知或需要发送祝福时，用户往往是对所编辑的短信进行群发，然而人数却有所限制。在本节将会详细介绍在 Android 手机中如何完成对通讯录中所有联系人发送短信功能的开发。

【实例描述】

本软件主界面中含有一个 EditText 文本编辑框和一个发送按钮，首先在 EditText 中填写需要发送的短信内容，当单击发送时会进入联系人列表中，查看需要发送的联系人，当单击其中一个联系人时，则将已编辑好的短信发送出去，此时会出现 Toas 显示发送成功。

本实例的运行效果如图 6-7 所示。

图 6-7　群发短信效果图

提示：在图 6-7 中依次表示的是填写短信内容的界面，单击“发送”按钮时，进入联系人界面，选中联系人，返回至主界面，并且将已编辑的短信发送给所有联系人，使用 Toast 对其进行提示。

【实现过程】

在本实例中首先在 EditText 编辑短信内容，当单击发送按钮时，获取短信内容。设置 Uri 并通过 Intent 访问联系人列表，重写 OnActivityResult()方法，在该方法中完成对联系人的搜索，并获取联系人的姓名和电话号码。

当获取联系人的电话号码后，便可以使用 SmsManager 发送短信。在完成上述功能前需要

为其设置读取联系人列表和发送短信的权限，否则在运行该软件时会出现运行错误，其权限代码分别为：<uses-permission android:name="android.permission.READ_CONTACTS"></uses-permission>和<uses-permission android:name="android.permission.SEND_SMS"></uses-permission>

【代码解析】

在本部分将对该实例的核心代码进行详细讲解，其中权限的设置已经在技术概要中给出，下面要讲解的是继承 Activity 类的开发，代码如下。

代码位置：见随书光盘中源代码/第 6 章/Sample6_7/src/com/bn/chap6/qfdx 目录下的 Sample6_7_Activity.java。

```
1   package com.bn.chap6.qfdx;                                    //导入相关类
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3   public class Sample6_7_Activity extends Activity {
4       EditText et;                                              //声明 EditText 引用
5       Button bSend;                                             //发送按钮
6       String msg;                                               //记录短信内容
7      public void onCreate(Bundle savedInstanceState) {
8         super.onCreate(savedInstanceState);
9         setContentView(R.layout.main);                          //设置当前界面
10        et=(EditText)this.findViewById(R.id.EditText01);        //创建对象
11        bSend=(Button)this.findViewById(R.id.Button01);
12        bSend.setOnClickListener(                               //发送按钮监听器
13          new OnClickListener(){
14                  public void onClick(View v) {
15                      msg=et.getText().toString().trim();//获取填写的短信内容
16                      Uri uri=Uri.parse("content://contacts/people");
                                                                  //访问联系人列表
17                      Intent intent=new Intent(Intent.ACTION_PICK,uri);
                                                                  //创建 Intent 对象
18                      startActivityForResult(intent,0);  //开启 Activity
19                  }});}
20     public void onActivityResult(int requestCode,int resultCode,Intent intent){
21      switch(requestCode){
22          case 0:
23              Uri uri=intent.getData();                         //创建 uri
24              if(uri!=null){
25                  try{
26                      Cursor cursor = getContentResolver().query(
                                                                  //获取联系人信息
27                          ContactsContract.Contacts.CONTENT_URI,
28                          null, null, null, null);
29                    while (cursor.moveToNext()){          //指向下一个
30                          String phoneName = cursor.getString(cursor.
getColumnIndex(
31                          ContactsContract.Contacts.DISPLAY_NAME));
                                                                  //获取用户姓名
32                          String contactId = cursor.getString(cursor.
getColumnIndex(
33                          ContactsContract.Contacts._ID));   //获取联系人_ID
34                          String hasPhone = cursor.getString(cursor.
getColumnIndex(
35                          ContactsContract.Contacts.HAS_PHONE_NUMBER));
                                                                  //获取号码信息
36                      if (hasPhone.compareTo("1") == 0){    //进行比较
37                      Cursor phones = getContentResolver().query(
                                                                  //获取用户所有电话
```

```
    38                          ContactsContract.CommonDataKinds.Phone.CONTENT_
URI,//uri
    39                          null,
    40                          ContactsContract.CommonDataKinds.Phone.CONTACT_ID +
    41                          " = "+ contactId,
    42                          null,
    43                          null);
    44                        while (phones.moveToNext()){
    45                          String phoneNumber = phones.getString(phones.
getColumnIndex(
    46                          ContactsContract.CommonDataKinds.Phone.NUMBER));
                                                    //所有电话号码
    47                          String phoneTpye = phones.getString(phones.
getColumnIndex(
    48                          ContactsContract.CommonDataKinds.Phone.TYPE));
                                                    //获取电话号码类型
    49                           if(phoneTpye.equals("2")){          //短信群发
    50                              SmsManager smsManager=SmsManager.getDefault();
    51                              PendingIntent  pi=PendingIntent.getBroadcast(
                                                    //创建对象
    52                              Sample6_7_Activity.this,0,new Intent(),0);
    53                              smsManager.sendTextMessage(
    54                                  phoneNumber, null, msg, pi, null);
                                                    //发送短信
    55                                  Toast.makeText(   //创建 Toast
    56                                          Sample6_7_Activity.this,
    57                                          "短信已发送到"+phoneName+"|"+
phoneNumber,
    58                                          Toast.LENGTH_SHORT).show();
                                                    //显示 Toast
    59                              }}
    60                          phones.close();             //关闭结果集
    61                      }}
    62                  }catch(Exception e){e.printStackTrace();}   //打印异常
    63              }
    64          break;
    65  }}}
```

其中：

- 第 4～19 行为声明 Button 以及 EditText 引用，并为 Button 按钮添加监听器，在监听器内首先获取 EditText 中内容，向联系人界面发送消息进入该界面，当单击其中一个联系人时，重新返回主界面。
- 第 20～43 行为首先获取 uri，并判断 uri 是否为空，若不为空则获取联系人信息，使用循环遍历所有联系人，从联系人中获取联系人姓名、联系人的 ID 以及联系人的号码。
- 第 44～65 行为循环遍历联系人的号码，从中获取联系人的详细电话号码，并将电话号码类型为 2 的作为发送短信的电话号码，给联系人列表中所有的电话号码类型为 2 的发送一条短信。

实例 8　开机程序自启动

每一部手机都会有自己开机自启动的程序，这些开机自启动的程序在有的时候大大方便了用户，在本节将会介绍在 Android 手机上如何实现开机自启动功能。

【实例描述】

在本软件中，主界面有四个按钮，依次实现的功能为开始 Service 服务，停止 Service 服务，绑定 Service 服务和解除绑定 Service 服务。当单击 Start Service 按钮时，启动 Service 服务，同时启动一个线程，不断打印信息。当单击 Stop Service 按钮时，停止 Service 服务，并关闭线程。

当单击 Bind Service 按钮时，启动 Service 服务，开启后台打印信息的线程，单击 Unbind Service 按钮时，关闭后台线程。若关闭该模拟器，然后重新启动，当模拟器启动完毕后，该软件将会自动运行，线程开启，并且开始打印信息。本实例运行效果图如图 6-8 所示。

图 6-8　本实例运行效果图

提示：在图 6-8 中，第一幅图为实例开始运行时的主界面，当单击 Start Service 按钮时，启动 Service，同时启动线程，在后台开始打印信息，如第二幅图所示，关闭模拟器，再重新开启时，则会直接运行后台线程，如第三幅图所示。

【实现过程】

在本实例中重点在于 BroadcastReceiver 的开发，在 Android 平台中，Broadcast 是一种被广泛运用在应用程序之间传输信息的机制。而 BroadcastReceiver 是一类不断地对发送出来的 BroadcastIntent 进行过滤、接收并响应的组件。

提示：在本实例中还需要对 BoradcastReceiver 进行注册。

注册 BroadcastReceiver 有两种方式，一种是静态地在 AndroidManifest.xml 文件中用<receiver>标签声明注册，并在标签内使用<intent-filter>标签设置过滤器。另一种是动态地在代码中先定义并设置好一个 IntentFilter 对象，然后在需要注册的地方调用 Context 对象的 registerReceiver 方法，如果取消注册就调用 Context 对象的 unregisterReceiver 方法。

另外若在使用 sendBroadcast 方法时指定了接收的权限，则只有在 AndroidManifest.xml 中使用<use-permission>标签声明了拥有此权限的 BroadcastReceiver，才会有可能接收到发送来的

那个 Broadcast Intent。

【代码解析】

经过上面的理论知识的介绍，读者对开机自启动程序应该有所了解。在本部分将对其核心代码进行详细讲解。首先需要介绍的是 AndroidManifest.xml 文件，在该文件中注册静态的 Receiver 服务，并添加权限设置，代码如下。

代码位置：见随书光盘中源代码/第 6 章/Sample6_8 目录下的 AndroidManifest.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>                    <!--版本号和编码方式-->
2   <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3                         package="com.bn.chap6.kjzqd"
4                         android:versionCode="1"
5             android:versionName="1.0">                      <!--版本名称-->
6       <application android:icon="@drawable/icon" android:label="@string/app_name">
7                 <activity android:name=".Sample6_8_Activity"
8                   android:label="@string/app_name">         <!--显示的程序名称-->
9               <intent-filter>
10                 <action android:name="android.intent.action.MAIN" />
                                                              <!--标识程序开始-->
11                 <category android:name="android.intent.category.LAUNCHER" />
12              </intent-filter>
13          </activity>
14            <service android:name=".MyService6_8"/>
15            <receiver android:name=".MyStartupReceiver">    <!-- 注册接收器 -->
16              <intent-filter>
17                <action android:name="android.intent.action.BOOT_COMPLETED"/>
18              </intent-filter>
19            </receiver>
20      </application>
21      <uses-sdk android:minSdkVersion="7" />
22       <!-- 在权限上要允许接收 BOOT_COMPLETED 消息 -->
23      <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
24  </manifest>
```

提示：在上述代码中第 15～19 行为注册 Service，并设置过滤器。第 22 行为添加权限设置。

上述代码主要介绍的是 AndroidManifest.xml 的开发，接下来要介绍的是该实例中主界面的实现类，在该类中主要为添加按钮的监听器，完成启动 Service 和启动线程，代码如下。

代码位置：见随书光盘中源代码/第 6 章/Sample6_8/src/com/bn/chap6/kjzqd 目录下的 Sample6_8_Activity.java。

```
1   package com.bn.chap6.kjzqd;                                   //声明包
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3   public class Sample6_8_Activity extends Activity {
4       ServiceConnection sc;                                     //声明引用
5       OnClickListener listener;                                 //声明引用
6       public void onCreate(Bundle savedInstanceState) {
7         super.onCreate(savedInstanceState);
8         setContentView(R.layout.main);                          //设置当前界面
9         sc=new ServiceConnection(){                             //创建对象
10              public void onServiceConnected(ComponentName name, IBinder service)
{
11              }
12              public void onServiceDisconnected(ComponentName name) {
```

```
13              }};
14        listener=new OnClickListener(){                    //创建对象
15              public void onClick(View v) {
16                  Intent intent=new Intent(Sample6_8_Activity.this,
MyService6_8.class);
17                  switch(v.getId()){
18                    case R.id.Button01:                    //Start Service 按钮
19                      startService(intent);                //启动 Service
20                    break;
21                    case R.id.Button02:                    //Stop Service 按钮
22                       stopService(intent);                //关闭 Service
23                  break;
24                    case R.id.Button03:                    //Bind Service 按钮
25                     bindService(intent,sc,BIND_AUTO_CREATE);
26                    break;
27                    case R.id.Button04:                    //Unbind Service 按钮
28                         unbindService(sc);                //接触绑定 Service
29                    break;
30                  }}};
31        this.findViewById(R.id.Button01).setOnClickListener(listener);
                                                           //添加监听器
32        this.findViewById(R.id.Button02).setOnClickListener(listener);
                                                           //添加监听器
33        this.findViewById(R.id.Button03).setOnClickListener(listener);
                                                           //添加监听器
34        this.findViewById(R.id.Button04).setOnClickListener(listener);
                                                           //添加监听器
35  }}
```

其中：

- 第 4～5 行为声明 ServiceConnection 和 OnClickListener 的引用。
- 第 7～30 行为创建 ServiceConnection 和 OnClickListener 对象，在创建的 ServiceConnection 对象中重写了 onServiceConnected()和 onServiceDisconnected()方法。在创建的 OnClickListener 的对象中，根据按钮的 ID 设定不同的业务逻辑。
- 第 31～34 行为给主界面的 Start Service、Stop Service、Bind Service 以及 Unbind Service 四个按钮添加监听器。

上述代码介绍的是本程序的 Activity 类的开发，接下来将要介绍的是继承 Service 类的开发，在该类中的 onCreate ()方法中创建线程对象，并启动线程，在 onDestroy()方法中关闭线程，代码如下。

代码位置：见随书光盘中源代码/第 6 章/Sample6_8/src/com/bn/chap6/kjzqd 目录下的 MyService6_8.java。

```
1   package com.bn.chap6.kjzqd;                              //声明包
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3   public class MyService6_8 extends Service{
4       TaskThread task;                                     //声明引用
5       public IBinder onBind(Intent arg0) {                 //重写 onBind()方法
6           Log.d("MyService", "=========onBind=========");  //打印信息
7           return null;
8       }
9       public boolean onUnbind(Intent arg0){                //重写onUnbind()方法
10          Log.d("MyService", "=========onUnbind=========");   //打印信息
11          return super.onUnbind(arg0);
12      }
13      public void onRebind(Intent arg0){                   /重写 onRebind ()方法
```

```
14          super.onRebind(arg0);
15          Log.d("MyService", "=========onRebind=========");    //打印信息
16      }
17      public void onCreate(){                                   //重写 onCreate ()方法
18          super.onCreate();
19          Log.d("MyService", "=========onCreate=========");    //打印信息
20          task=new TaskThread();                                //创建线程对象
21          task.start();                                         //启动线程
22      }
23      public void onDestroy(){                                  //重写 onDestroy()方法
24          super.onDestroy();
25          Log.d("MyService", "=========onDestroy=========");   //打印异常
26          task.flag=false;                                      //将线程标志位设反
27  }}
```

其中：

- 第 4～12 行为重写 onBind()和 onUnbind()方法，在这两个方法中主要完成的是打印信息。
- 第 13～16 行为重写 onRebind()方法，在该方法中也主要是完成打印信息。
- 第 17～27 行为重写 onCreate()和 onDestroy()方法，在 onCreate()方法中主要完成的是创建线程对象，并启动线程；在 onDestroy()方法中主要完成的是将线程标志位设为 false，关闭线程。

上述代码介绍的是继承 Service 类的开发，接下来介绍的是继承 BroadcastReceiver 类的开发，其重写 onReceiver()方法，通过过滤器接收系统发送的信息，代码如下。

代码位置：见随书光盘中源代码/第 6 章/Sample6_8/src/com/bn/chap6/kjzqd 目录下的 MyStartupReceiver.java。

```
1   package com.bn.chap6.kjzqd;                                   //声明包
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3   public class MyStartupReceiver extends BroadcastReceiver{    //创建类
4       public void onReceive(Context arg0, Intent arg1) {
5           Intent intent=new Intent(arg0,MyService6_8.class);  //创建对象
6           arg0.startService(intent);
7   }}
```

提示：上述代码主要完成的任务是通过过滤器接收系统发送的信息。

上述代码介绍的是本程序 MyStartupReceiver 类的开发，接下来介绍的是 TaskThread 线程类，在该类中主要完成的是每 1 秒钟打印 1 次信息，代码如下。

代码位置：见随书光盘中源代码/第 6 章/Sample6_8/src/com/bn/chap6/kjzqd 目录下的 TaskThread.java。

```
1   package com.bn.chap6.kjzqd;                                   //声明包
2   public class TaskThread extends Thread{
3       public boolean flag=true;                                 //声明标志位
4      public void run(){                                         //重写的方法
5       while(flag){
6           System.out.println("=========TaskThread=========");  //打印信息
7           try{
8               Thread.sleep(1000);                               //睡眠 1 秒钟
9           }
10          catch(Exception e){flag=false;}                       //将标志位设为 false
11  }}}
```

提示：上述代码主要完成的是每 1 秒钟打印 1 次信息。

实例 9　手机状态提醒

手机的状态可以分为待机状态、来电状态以及电话接通状态，如何来判断手机的当前状态，将在本节对其进行介绍。

【实例描述】

当运行本软件时，首先根据当前是否有电话拨入，判断手机的状态，若没有则为待机状态。然后当有电话拨入时，提醒用户有来电以及来电人的电话号码，并且手机开始震动。再然后当接听该电话时，提醒用户电话已接通，手机停止震动。若没有接听该电话，手机又重新回到待机状态，手机的上述三种状态如图 6-9 所示。

图 6-9　手机的三种状态

提示：图 6-9 中依次表示的是手机的待机状态，有来电状态和电话接通状态。

【实现过程】

想要获取手机当前的状态，首先需要设置的是读取电话状态的权限，该权限为 android:name="android.permission.READ_PHONE_STATE"，否则将无法获取手机当前的状态。然后在 onCreate()方法中获取系统的 TelephonyNumber 服务，创建 PhoneStateListener 对象。

重写 onCallStateChanged()方法，根据传入的状态值，判断手机处于何种状态，手机的三种状态分为 CALL_STATE_IDLE、CALL_STATE_RINGING 和 CALL_STATE_OFFHOOK，分别表示待机、有来电以及通话中。

【代码解析】

在本部分会详细介绍本程序的实现过程。首先介绍的是 AndroidManifest.xml 的开发，代码

如下。

代码位置：见随书光盘中源代码/第 6 章/Sample6_9 目录下的 AndroidManifest.xml。

```
<?xml version="1.0" encoding="utf-8"?>                    <!--版本号和编码方式-->
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
                  package="com.bn.chap6.ldxs"
                  android:versionCode="1"
          android:versionName="1.0">                      <!--版本名称-->
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".Sample6_9_Activity"
                  android:label="@string/app_name">       <!--设置 SDK 的值-->
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-permission android:name="android.permission.VIBRATE"></uses-
permission>
     <uses-permission android:name="android.permission.READ_PHONE_STATE">
</uses-permission>
</manifest>
```

提示：上述代码中，第 15 行为添加手机震动的权限，第 16 行为添加获取手机状态的权限。若不添加手机震动的权限，运行时会报错；若不添加获取手机状态的权限，则无法获取手机的当前状态。

上述代码介绍的是本程序开始时设置权限的 xml 文件，接下来要介绍的是继承 Activity 类的开发，在该类中完成的是本实例的主要功能，代码如下。

代码位置：见随书光盘中源代码/第 6 章/Sample6_9/src/com/bn/chap6/ldxs 目录下的 Sample6_9_Activity.java。

```
package com.bn.chap6.ldxs;                                    //声明包
……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
public class Sample6_9_Activity extends Activity {
    TextView tv;                                              //声明引用
    Vibrator vibrator;                                        //手机震动引用
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);                     //调用父类
      setContentView(R.layout.main);                          //设置当前界面
      tv=(TextView)this.findViewById(R.id.TextView01);        //获取 TextView 对象
      vibrator=(Vibrator)getApplication().getSystemService(
            Service.VIBRATOR_SERVICE);                        //创建 Vibrator 对象
      myPhoneStateListener mPSL=new myPhoneStateListener();   //创建对象
      TelephonyManager tm=(TelephonyManager)this.getSystemService(
    Context.TELEPHONY_SERVICE);                    //创建 TelephonyManager 对象
      tm.listen(mPSL, PhoneStateListener.LISTEN_CALL_STATE);//添加监听器
    }
    public class myPhoneStateListener extends PhoneStateListener{
     public void onCallStateChanged(int state,String incomingNumber){
                                                    //重写 onCallStateChanged 方法
        switch(state){
            case TelephonyManager.CALL_STATE_IDLE:        //手机待机状态
                Toast.makeText(                           //创建 Toast 对象
                        Sample6_9_Activity.this,
                        "手机处于待机状态！",
```

```
24                          Toast.LENGTH_SHORT).show();     //显示 Toast
25                      tv.setText("手机处于待机状态！");          //设置文本
26                  break;
27                  case TelephonyManager.CALL_STATE_OFFHOOK:   //接电话状态
28                      Toast.makeText(                         //创建 Toast 对象
29                              Sample6_9_Activity.this,
30                              "电话已接通！",
31                              Toast.LENGTH_SHORT).show();     //显示 Toast
32                      vibrator.cancel();                      //取消震动
33                      Toast.makeText(                         //创建 Toast 对象
34                              Sample6_9_Activity.this,
35                              "震动取消！",
36                              Toast.LENGTH_SHORT).show();     //显示 Toast
37                      tv.setText("电话已接通！");               //设置文本
38                  break;
39                  case TelephonyManager.CALL_STATE_RINGING:   //电话响铃状态
40                      Toast.makeText(                         //创建 Toast 对象
41                              Sample6_9_Activity.this,
42                              "有来电，电话号码："+incomingNumber+",请接听！",
43                              Toast.LENGTH_SHORT).show();     //显示 Toast
44                      vibrator.vibrate(new long[]{100,10,100,1000},0);
                                                    //设置手机震动,0 表示持续震动
45                      Toast.makeText(
46                              Sample6_9_Activity.this,
47                              "手机正在震动！",
48                              Toast.LENGTH_SHORT).show();     //显示 Toast
49                      tv.setText("有来电，电话号码："+incomingNumber+",请接听！");
50                  break;
51              }
52              super.onCallStateChanged(state, incomingNumber);//调用父类方法
53      }}}
```

其中：

- 第 4～16 行主要完成成员变量的声明与创建，在 onCreate()方法中，首先设置 main.xml 为当前界面，然后依次创建 TextView、Vibrator 和继承 PhoneStateListener 类的 myPhoneStateListener 的对象，并在创建 TelephonyManager 对象后为其添加监听器。
- 第 17～53 行主要完成对手机状态的判断。在重写的 onCallStateChanged()方法中，根据所传的状态（state），判断手机的当前状态，若为待机状态，则提示目前为待机状态，若有来电，则提示电话拨入，并且手机开始震动，接通该电话，则手机处于通话状态，手机停止震动。若没有接通该电话，则手机再次进入待机状态。

实例 10　有来电时，发送短信回复

在上面一节中介绍了如何获取手机当前的状态，有了手机的状态就可以完成一些其他任务，比如不方便接电话时，挂掉电话时，给来电用户发送一条短信，或者通知来电用户现在正在忙碌中，稍后回复等信息。在本节将对以上功能进行介绍。

【实例描述】

运行本软件时，手机获取当前状态，并将状态显示在主界面上，当有来电时，提醒用户有来电，并自动发送一条短信通知来电用户请等待。若手机用户挂掉电话，则发送短信给来电用户，通知来电用户稍后回复。本实例运行效果图如图 6-10 所示。

图 6-10　本实例运行效果图

提示：图 6-10 中依次表示的是本实例开始运行的效果，有来电时的效果图，以及关掉电话将短信发送到来电用户的效果图。

【实现过程】

在本实例中同样需要设置权限并获取当前手机的状态，权限代码为 android:name="android.permission.READ_PHONE_STATE"，否则将无法获取手机当前的状态。然后在 onCreate()方法中获取系统的 TelephonyNumber 服务，创建 PhoneStateListener 对象。

重写 onCallStateChanged()方法，根据传入的状态值，判断手机处于何种状态，在有来电状态下，使用 Handler 对象向来电用户发送等待消息。当用户忙挂掉电话时，向来电用户发送短信。发送短信需要设置权限，权限为 android:name="android.permission.SEND_SMS"。

【代码解析】

在本部分会详细介绍本程序的实现过程。首先介绍的是本 AndroidManifest.xml 的开发，代码如下。

代码位置：见随书光盘中源代码/第 6 章/Sample6_10 目录下的 AndroidManifest.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>                <!--版本号和编码方式-->
2   <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3                        package="com.bn.chap6.lddxhf"
4                    android:versionCode="1"
5       android:versionName="1.0">                        <!--版本名称-->
6     <application android:icon="@drawable/icon" android:label="@string/app_name">
7               <activity android:name=".Sample6_10_Activity"
8                android:label="@string/app_name">        <!--显示的程序名称-->
9           <intent-filter>
10             <action android:name="android.intent.action.MAIN" />
                                                          <!--标识程序开始-->
11             <category android:name="android.intent.category.LAUNCHER" />
12          </intent-filter>
13       </activity>
14    </application>
```

```
    <!-- 设置权限 -->
    <uses-permission android:name="android.permission.READ_PHONE_STATE">
</uses-permission>
    <uses-permission android:name="android.permission.SEND_SMS"></uses-
permission>
</manifest>
```

提示：上述代码中第 16 行为获取手机状态的权限设置，第 17 行为发送短信的权限设置。

上述代码已经介绍了 AndroidManifest.xml 的开发，接下来介绍的是继承 Activity 类的开发，在该类中开发的是本软件的主要功能，代码如下。

代码位置：见随书光盘中源代码/第 6 章/Sample6_10/src/com/bn/chap6/lddxhf 目录下的 Sample6_10_Activity.java。

```
package com.bn.chap6.lddxhf;                                    //声明包
……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
public class Sample6_10_Activity extends Activity {
    TextView tv;                                                //声明 TextView 引用
    int count=0;                                                //设置标志位
    Handler hd=new Handler(){                                   //创建 Handler 对象
        public void handleMessage(Message msg){                 //重写的方法
            switch(msg.what){
                case 0:
                    Bundle b=msg.getData();                     //获取数据
                    String incomingNumber=(String) b.get("number");
                                                                //获取来电号码
                    SmsManager smsManager=SmsManager.getDefault();
                PendingIntent pi=PendingIntent.getBroadcast(
                                                        //创建 PendingIntent 对象
                        Sample6_10_Activity.this,0,new Intent(),0);
                String tempMsg="请等待用户接听。";               //创建字符串
                smsManager.sendTextMessage(
                        incomingNumber,                         //目标电话号码
                        null,
                        tempMsg,                                //发送的消息
                        pi,
//PendingIntent 对象
                        null);                                  //发送短信
                Toast.makeText(                                 //创建 Toast 对象
                        Sample6_10_Activity.this,
                        "短信已发送到"+incomingNumber,
                        Toast.LENGTH_SHORT).show();             //提示信息
                break;
                case 1:
                    Bundle bb=msg.getData();                    //获取数据
                    String incomingNumber2=(String) bb.get("number");
                                                                //获取来电号码
                    SmsManager smsManager2=SmsManager.getDefault();
                PendingIntent pi2=PendingIntent.getBroadcast(
                                                        //创建 PendingIntent 对象
                        Sample6_10_Activity.this,0,new Intent(),0);
                smsManager2.sendTextMessage(
                        incomingNumber2,                        //来电号码
                        null,
                        "我现在不方便接电话，稍后打给你",          //发送的信息
                        pi2,
```

```
38                          null);                                  //发送短信
39                  Toast.makeText(                                 //创建 Toast 对象
40                          Sample6_10_Activity.this,
41                          "短信已发送到"+incomingNumber2,
42                          Toast.LENGTH_SHORT).show();             //提示信息
43                  break;
44              }}};
45      /*该处省略了该类的部分方法，将在后面给出*/
```

其中：

- 第 8～26 行为获取信息编号为 0 时，首先从获取的数据中取得电话号码，然后创建 SmsManager 和 PendingIntent 对象，向来电用户发送短信。
- 第 27～44 行为获取信息编号为 1 时，同样是获取数据中的电话号码，创建 SmsManager 和 PendingIntent 对象，向来电用户发送短信，短信的内容与前面的不同。
- 第 45 行为重写 onCreate 方法和声明内部类，在后面将会详细介绍。

上述代码介绍了继承 Activity 类的开发，接下来介绍的是 onCreate 方法的具体实现以及内部类 myPhoneStateListener 的开发，将代码插入到 Activity 类的开发的第 45 行。

代码位置：见随书光盘中源代码/第 6 章/Sample6_10/src/com/bn/chap6/lddxhf 目录下的 Sample6_10_Activity.java。

```
1       public void onCreate(Bundle savedInstanceState) {
2          super.onCreate(savedInstanceState);                      //调用父类
3          setContentView(R.layout.main);                           //设置当前界面
4          tv=(TextView)this.findViewById(R.id.TextView01);         //获取对象
5          myPhoneStateListener mPSL=new myPhoneStateListener();    //创建对象
6          TelephonyManager tm=(TelephonyManager)
7          this.getSystemService(Context.TELEPHONY_SERVICE);
                                                                    //创建 TelephonyManager 对象
8          tm.listen(mPSL, PhoneStateListener.LISTEN_CALL_STATE);//添加监听器
9       }
10      public class myPhoneStateListener extends PhoneStateListener{ //内部类
11       public void onCallStateChanged(int state,String incomingNumber){
                                                                    //重写的方法
12          switch(state){
13              case TelephonyManager.CALL_STATE_IDLE:              //手机待机状态
14                  Toast.makeText(                                 //创建 Toast
15                          Sample6_10_Activity.this,
16                          "手机处于待机状态！",
17                          Toast.LENGTH_SHORT).show();             //提示信息
18                  tv.setText("手机处于待机状态！");                //设置主界面内容
19                  if(count==1){                                   //若标志位为 1 时
20                      Bundle bundle=new Bundle();                 //创建 Bundle 对象
21                      bundle.putString("number", incomingNumber);//绑定号码
22                      Message m=new Message();                    //创建 Message 对象
23                      m.what=1;                                   //设置编号
24                      m.setData(bundle);                          //设定 Bundle
25                      hd.sendMessage(m);                          //Handler 发送消息
26                      count=0;                                    //标志位设为 0
27                  }
28              break;
29              case TelephonyManager.CALL_STATE_OFFHOOK:           //接电话状态
30                  Toast.makeText(                                 //创建 Toast 对象
31                          Sample6_10_Activity.this,
32                          "电话已接通！",                          //显示内容
33                          Toast.LENGTH_SHORT)                     //显示时间
```

```
34                          .show();                                    //显示信息
35                  tv.setText("电话已接通！");                          //显示信息
36                  count=0;                                            //标志位设为 0
37              break;
38              case TelephonyManager.CALL_STATE_RINGING:   //电话响铃状态
39                  Toast.makeText(                                     //创建对象
40                          Sample6_10_Activity.this,
41                          "有来电，电话号码："+incomingNumber+",请接听！",
42                          Toast.LENGTH_SHORT).show();          //显示信息
43                  tv.setText("有来电，电话号码："+incomingNumber+",请接听！");
44                  Bundle bundle=new Bundle();                         //创建 Bundle 对象
45                  bundle.putString("number", incomingNumber);
46                  Message m=new Message();                            //创建 Message 对象
47                  m.what=0;                                           //设置编号
48                  m.setData(bundle);
49                  hd.sendMessage(m);                                  //发送消息
50                  count=1;                                            //标志位设为 1
51              break;
52          }
53          super.onCallStateChanged(state, incomingNumber);//调用父类方法
54  }}
```

其中：

- 第 1～9 行主要完成的是创建 myPhoneStateListener 和 TelephonyManager 的对象，并为创建的 TelephonyManager 对象设置监听器。
- 第 13～28 行为手机处于待机状态时完成的业务，当标志位为 0 时，表示手机不是由响铃状态转入，若为标志位为 1，表示手机由响铃状态进入等待状态，此时通过 Handler 发送消息向来电用户发送短信。
- 第 29～36 行为手机处于接听状态时完成的业务，当电话接通时，使用 Toast 提示，并将标志位设为 1。
- 第 37～51 行为手机处于响铃状态时完成的业务，当电话进入响铃状态时，首先通过 Handler 发送消息向来电用户发送短信通知，并将标志位设为 1。

实例 11　手机存储卡容量的查询

每一个手机中都会有一张存储卡，在存储卡中存放着自己喜欢的相片或是音乐，在日常生活中为我们提供了很大的帮助，在本节将要介绍如何获取手机中存储卡的容量，使用户自己知道存储卡使用了多少，还剩余多少容量。

【实例描述】

当运行本软件时，首先进入的是主界面，在主界面中分为 SD 卡的总容量、SD 卡的已使用量和 SD 卡的可用量，以及查询 SD 卡容量的按钮，当单击按钮时，首先判断手机中是否已安装 SD 卡，若未安装则在界面上显示的容量为 0，若安装了 SD 卡，则从系统中获取 SD 卡的信息，将获取的 SD 卡信息分别填写到相应的位置，本实例的运行效果如图 6-11 所示。

通过该软件便可以轻松地获取 SD 卡的总容量以及剩余容量。

提示：图 6-11 中依次表示的是开始运行时的主界面，单击查询 SD 卡容量按钮后获取的 SD 卡信息。模拟器中 SD 卡的容量是在创建 AVD 时设置的。

图 6-11　SD 卡容量查询

【实现过程】

正如上面所提到的，在获取 SD 卡信息之前首先需要判断手机中是否已安装 SD 卡，判断 SD 卡的状态可以通过 Environment.getExternalStorageState()方法获取。然后为按钮添加监听器，在设置的监听器中，首先通过 Environment 获取 SD 卡路径，然后获取 SD 卡的详细信息。

创建 DecimalFormat 对象，并设置获取的容量大小每 3 位为一组进行分组，最后将计算得到的 SD 卡容量显示在主界面上。

【代码解析】

在本部分将要介绍的是该实例的核心代码，首先介绍的是主界面 main.xml 的配置，该文件主要完成的是主界面的搭建，代码如下。

代码位置：见随书光盘中源代码/第 6 章/Sample6_11/res/layout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>          <!--版本号和编码方式-->
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3                   android:orientation="vertical"
4                   android:layout_width="fill_parent"
5                   android:layout_height="fill_parent"
6       >                                           <!--设置 LinearLayout-->
7       <LinearLayout
8                   android:id="@+id/LinearLayout01"
9                   android:layout_width="fill_parent"
10                  android:layout_height="wrap_content"
11       android:orientation="horizontal">          <!--设置 LinearLayout-->
12           <TextView
13                  android:layout_width="100dip"
14                  android:layout_height="wrap_content"
15                  android:text="SD 卡总容量："
16          />                                      <!--设置 TextView-->
17          <EditText
18                  android:id="@+id/EditText01"
19                  android:layout_width="fill_parent"
20                  android:layout_height="wrap_content"
21                android:editable="false">         <!--设置 EditText-->
```

```
22          </EditText>
23      </LinearLayout>
24      ……//该处省略了部分类似代码，读者可自行查看随书光盘中源代码。
25        <Button
26                      android:text="查询 SD 卡容量"
27                      android:id="@+id/Button01"
28                      android:layout_width="wrap_content"
29            android:layout_height="wrap_content">       <!--设置 Button-->
30      </Button>
31  </LinearLayout>
```

其中：

- 第 7～23 行为 SD 卡总容量控件的设置，首先将 LinearLayout 的 orientation 属性设置为水平，然后加入一个 TextView 和 EditText 控件，将 EditText 控件的 editable 属性设置为 false。
- 第 25～30 行表示的是在 LinearLayout 中添加 Button 按钮控件，并设置该控件的具体属性。

上述代码主要介绍的是本程序的主界面的搭建，接下来要介绍的是继承 Activity 类的开发，在该类中主要完成的是获取 SD 卡容量，并将其设置为 EditText 的显示的方法，代码如下。

代码位置：见随书光盘中源代码/第 6 章/Sample6_11/src/com/bn/chap6/sd 目录下的 Sample6_11_Activity.java。

```
1   package com.bn.chap6.sd;
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3   public class Sample6_11_Activity extends Activity {
4       Button button;                                          //查询按钮
5       EditText etTotal;                                       //总容量
6       EditText etUsed;                                        //已使用量
7       EditText etAvailable;                                   //未使用量
8      public void onCreate(Bundle savedInstanceState) {
9         super.onCreate(savedInstanceState);
10        setContentView(R.layout.main);                        //设置当前界面
11        button=(Button)this.findViewById(R.id.Button01);      //查询按钮
12        etTotal=(EditText)this.findViewById(R.id.EditText01); //总容量对象
13        etUsed=(EditText)this.findViewById(R.id.EditText02);  //已使用量对象
14        etAvailable=(EditText)this.findViewById(R.id.EditText03);
                                                                //未使用量对象
15        button.setOnClickListener(                            //添加监听器
16          new OnClickListener(){
17                  public void onClick(View v) {
18                      if(Environment.getExternalStorageState().equals
19                          (Environment.MEDIA_MOUNTED)){ //判断存储卡是否插入
20                          File path=Environment.getExternalStorage
Directory();//获取路径
21                          StatFs sf=new StatFs(path.getPath());
                                                    //创建 StatFs 对象
22                          long size=sf.getBlockSize();  //SD 卡单位大小
23                          long total=sf.getBlockCount();     //总数量
24                          long available=sf.getAvailableBlocks();
                                                    //可使用的数量
25                          DecimalFormat df=new DecimalFormat();
                                                    //创建对象
26                          df.setGroupingSize(3);          //每 3 位分为一组
27                          String totalSize=(size*total)/1024>=1024? //总容量
28                              df.format((((size*total)/1024)/1024))+"MB":
29                              df.format((size*total)/1024)+"KB";
30                          String availableSize=(size*available)/1024>=1024?
```

```
                                                        //未使用量
    31                              df.format((((size*available)/1024)/1024))
+"MB":
    32                              df.format((size*available)/1024)+"KB";
    33                          String usedSize=(size*(total-available))/1024>=
1024?//已使用量
    34                              df.format((((size*(total-available))/1024)/
1024))+"MB":
    35                              df.format((size*(total-available))/1024)
+"KB";
    36                          etTotal.setText(totalSize);     //总容量
    37                          etUsed.setText(usedSize);       //已使用量
    38                          etAvailable.setText(availableSize);     //未使用量
    39                      }else if(Environment.getExternalStorageState().equals
    40                          (Environment.MEDIA_REMOVED))
    41                      {                                   //SD 卡已移除
    42                          etTotal.setText(0);             //总容量
    43                          etUsed.setText(0);              //已使用量
    44                          etAvailable.setText(0);         //未使用量
    45  }}});}}
```

其中：

- 第 4～14 行为声明引用，并创建对象。
- 第 15～38 行主要完成的是为 Button 按钮添加监听器，在监听器中首先根据 Environment.getExternalStorageState()获取 SD 卡的状态，若已安装 SD 卡，接下来是获取路径，创建 StatFs 对象，获取 SD 卡单位大小和总数量等信息，通过 DecimalFormat 的 format()方法，将计算出的数据每 3 位分为一组，最后将其设置显示在主界面上。
- 第 39～45 行为未安装 SD 卡的情况，若手机未安装 SD 卡，直接将总容量、已使用量和未使用量设置为 0。

实例 12　备忘录的定时提醒

现在手机的一个很重要的功能就是能够设置备忘录，并且在设定的时间提醒用户。Android 平台下的手机同样可以设定，并且可以自己制作备忘录，在设定好的时间提醒用户。在本节将对上述功能进行详细的介绍。

【实例描述】

本软件简单实现了备忘录的设置与提醒，当运行软件时，在主界面可以设定备忘录内容，单击按钮设置提醒的时间。每次设置一个备忘录时，会在按钮下方记录所设定的备忘录，方便用户查看。当设定备忘录的时间到时，在主界面上自动弹出一个对话框，进行提示，本实例的运行效果如图 6-12 所示。

提示：在图 6-12 中依次为主界面的备忘录内容设置与查看，时间的设置和到设定时间时的提醒。

【实现过程】

首先需要获取的是 AlarmManager，AlarmManager 是通过 getSystemService(ALARM_SERVICE)获取的，并使用 set()方法设定闹钟。

在 Button 按钮的单击事件中，弹出时间对话框 TimeDialog 进行时间的设定，设置完成后，以 PendingIntent.getBroadcast()产生 PendingIntent，通过 AlarmManager 的 set()方法将设置的时间与 PendingIntent 传入 AlarmManager，当到达设定时间时，使用自定义对话框对用户进行提示。备忘录的设置与提醒如图 6-12 所示。

图 6-12　备忘录的设置与提醒

提示：AlarmManager 的 set()方法设定闹钟只是提醒一次，而 setRepeating()方法可以设定闹钟的重复提醒，在 setRepeating()方法中需要设定闹钟的开始时间和时间间隔。

自定义对话框需要重写 onCreateDialog(int id)和 onPrepareDialog(int id,Dialog dialog)两个方法，显示对话框是使用 showDialog()方法实现的，当对话框第一次创建时，首先执行的是 onCreateDialog(int id)方法，然后执行的是 onPrepareDialog(int id,Dialog dialog)方法，以后显示该对话框时只执行 onPrepareDialog(int id,Dialog dialog)方法。

【代码解析】

通过上面的介绍，相信读者对此有了一定的了解，接下来要介绍的是核心代码部分，首先介绍的是 AndroidManifest.xml 文件，在该文件中主要完成的是 receiver 的注册，代码如下。

代码位置：见随书光盘中源代码/第 6 章/Sample6_12 目录下的 AndroidManifest.xml。

```
<?xml version="1.0" encoding="utf-8"?>                    <!--版本号和编码方式-->
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
                  package="com.bn.chap6.alarm"
                  android:versionCode="1"
     android:versionName="1.0">                           <!--版本名称-->
    <application android:icon="@drawable/icon" android:label="@string/app_name">
              <activity android:name=".Sample6_12_Activity"
                android:label="@string/app_name">        <!--显示的程序名称-->
            <intent-filter>
               <action android:name="android.intent.action.MAIN" />
               <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
      <receiver android:name=".AlarmReceiver"            <!--注册 receiver-->
                android:process=":remote"></receiver><!--设置 process 属性-->
    </application>
</manifest>
```

提示：上述代码中第 14、15 行为注册 receiver，并且设定 process 的属性为 remote。

上述代码介绍的是 AndroidManifest.xml 的开发，接下来介绍的是继承 Activity 类的开发，在该类中主要完成的是备忘录的设置，代码如下。

代码位置：见随书光盘中源代码/第 6 章/Sample6_12/src/com/bn/chap6/alarm 目录下的 Sample6_12_Activity.java。

```
1   package com.bn.chap6.alarm;                                          //导入相关包
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3   public class Sample6_12_Activity extends Activity {
4      EditText et;                                                      //备忘录编辑框
5      Button button;                                                    //设置按钮
6      String msg;                                                       //备忘录信息
7      Dialog dialog;                                                    //对话框
8      private final int DIALOG=0;
9      TextView tv;                                                      //记录备忘录
10     StringBuilder sb;                                                 //声明引用
11     int count;                                                        //计数器
12     public void onCreate(Bundle savedInstanceState) {
13           super.onCreate(savedInstanceState);
14           setContentView(R.layout.main);                              //设置当前界面
15           et=(EditText)this.findViewById(R.id.EditText01);
16           button=(Button)this.findViewById(R.id.Button01);
17           tv=(TextView)this.findViewById(R.id.TextView01);            //创建对象
18           Bundle bundle=this.getIntent().getExtras();      //取得短信发来 bundle
19           sb=new StringBuilder();                          //创建 StringBuilder 对象
20           if(bundle!=null){
21               showDialog(DIALOG);                                     //显示对话框
22           }
23           final Calendar c=Calendar.getInstance();
24           button.setOnClickListener(                                  //设置按钮监听器
25           ……//该处省略了部分代码，将在下面详细介绍。
26  );}
27       public Dialog onCreateDialog(int id){
28           Dialog result=null;
29           switch(id){
30               case DIALOG:                                            //对话框的初始化
31                   AlertDialog.Builder mywktzb=new AlertDialog.Builder(this);
32                   mywktzb.setItems(                                   //设置具体属性
33                   null,
34                   null
35                   );
36                   dialog=mywktzb.create();                            //创建对话框
37                   result=dialog;                                      //为 result 赋值
38               break;                                                  //退出
39               }
40           return result;                                              //返回对话框
41      }
42      public void onPrepareDialog(int id, final Dialog dialog){ //弹出对话框的方法
43       switch(id){
44           case DIALOG:                                                //提醒对话框
45               dialog.setContentView(R.layout.dialog);                 //设置界面
46               Button mywktzBOk=(Button)dialog.findViewById(R.id.mywktzOk);
47               mywktzBOk.setOnClickListener(                           //添加监听器
48               new OnClickListener(){                                  //匿名内部类
```

```
49                  public void onClick(View arg0) {            //重写的方法
50                      dialog.cancel();                        //关闭对话框
51                      Sample6_12_Activity.this.finish();      //结束 Activity
52                  }});
53              dialog.setCancelable(true);                     //允许单击返回键
54          break;
55  }}}
```

其中：

- 第 4～11 行为声明该类的成员变量。
- 第 13～26 行创建对象，并判断 Bundle 对象是否为空，若为空则表示备忘录的设定时间尚未到，若不为空，则表示备忘录时间已到，弹出对话框进行提醒。
- 第 27～41 行为重写 onCreateDialog()方法，该方法在该程序运行过程中只执行一次，将创建好的对话框返回。
- 第 42～55 行为重写 onPrepareDialog()方法，该方法为每次弹出对话框时调用的方法，在该方法中首先设定界面，创建 Button 对象，并为 Button 添加监听器，当单击“Button”按钮时，关闭对话框并结束 Activity。

接下来要介绍的是主界面中设置闹钟按钮监听器的实现，将下列代码插入到 Activity 类的第 25 行。

代码位置：见随书光盘中源代码/第 6 章/Sample6_12/src/com/bn/chap6/alarm 目录下的 Sample6_12_Activity.java。

```
1   new OnClickListener(){
2       public void onClick(View v) {
3           msg=et.getText().toString().trim();             //获取备忘录信息
4           sb.append(count++);
5           sb.append(".备忘录内容为：");                    //组装字符串
6           sb.append(msg);
7           sb.append("\n");
8           tv.setText(sb.toString().trim());               //设置内容
9           c.setTimeInMillis(System.currentTimeMillis()); //将当前时间设为默认时间
10          int hour=c.get(Calendar.HOUR_OF_DAY);           //小时
11          int minute=c.get(Calendar.MINUTE);              //分钟
12          new TimePickerDialog(
13              Sample6_12_Activity.this,
14              new TimePickerDialog.OnTimeSetListener() {
15                  public void onTimeSet(TimePicker view, int hourOfDay, int
minute) {
16                      c.setTimeInMillis(System.currentTimeMillis());
                                                            //设置当前时间
17                      c.set(Calendar.HOUR_OF_DAY, hourOfDay);     //设置小时
18                      c.set(Calendar.MINUTE, minute);             //设置分钟
19                      c.set(Calendar.SECOND, 0);                  //设置秒
20                      c.set(Calendar.MILLISECOND, 0);             //设置毫秒
21                      Intent intent=new Intent(
22                      Sample6_12_Activity.this,AlarmReceiver.class);
                                                    //运行 AlarmReceiver 类
23                      PendingIntent pi=PendingIntent.getBroadcast(
                                                    //创建 PendingIntent 对象
24                      Sample6_12_Activity.this, 0, intent, 0);
25                      AlarmManager alarm=(AlarmManager)
                                                    //创建 AlarmManager 对象
26                      Sample6_12_Activity.this.getSystemService(ALARM_
SERVICE);
25                    alarm.set(AlarmManager.RTC_WAKEUP, c.getTimeInMillis(),
```

```
pi);//设置闹钟
    26                          String   tempHour=(hourOfDay+"").length()>1?hourOfDay+
"":"0"+hourOfDay;
    27                          String tempMinute=(minute+"").length()>1?minute+
"":"0"+minute;
    28                          Toast.makeText(                    //创建 Toast
    29                              Sample6_12_Activity.this,
    30                              "设置的时间为："+tempHour+":"+tempMinute,
    31                              Toast.LENGTH_SHORT).show();    //显示 Toast
    32                              }
    33                  },hour,minute,true).show();                //显示时间对话框
    34  }}
```

其中：

- 第 1～11 行为获取输入的备忘录信息，并将备忘录信息进行重新组装，将当前时间设定为默认时间。
- 第 12～34 行为时间对话框的设定，首先是获取设定的时间，然后创建 AlarmManager 和 PendingIntent 对象，将 PendingIntent 传入到 AlarmManager 中。在时间对话框上显示的时间通过正则式进行规划，最后使用 Toast 将设定好的时间显示在界面上。

上述代码介绍的是本程序 Activity 类的开发，接下来介绍的是 AlarmReceiver 类的开发，代码如下。

代码位置：见随书光盘中源代码/第 6 章/Sample6_12/src/com/bn/chap6/alarm 目录下的 Sample6_12_Activity.java。

```
1   package com.bn.chap6.alarm;                                  //声明包
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3   public class AlarmReceiver extends BroadcastReceiver{
4       public void onReceive(Context context, Intent intent) {
                                                                 //重写的 onReceive 方法
5           Intent  tempIntent=new  Intent(context,Sample6_12_Activity.class);
//创建 Intent 对象
6           Bundle myBundle=new Bundle();                        //创建 Bundle 对象
7           myBundle.putString("msg", "msg");
8           tempIntent.putExtras(myBundle);                      //绑定 Bundle
9           tempIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK); //设置新的 task
10          context.startActivity(tempIntent);                   //启动 Actvity
11  }}
```

提示：上述代码主要完成的是，当设定的时间到时，BroadcastReceiver 会自动执行 onReceiver()方法，返回至 Sample6_12_Activity。

实例 13　设置手机静音和固定号码来电时手机震动

有时候需要及时地将手机设置为静音模式，有时又希望一些重要号码来电时手机震动，在本节将对上述功能在 Android 手机上的实现进行介绍。

【实例描述】

本软件设定的模式简单，在主界面仅有两种模式选择，一种是会议模式，另一种是正常模式，当选择会议模式时，手机设定为静音，当选择正常模式时，手机设定为响铃。当单击添加按钮时，系统会自动从编辑框中提取输入电话号码，若输入的电话号码中不全为数字时，使用

Toast 对用户进行错误提醒。

当有需要震动提醒的号码拨入时，手机将自动开始震动；接通电话或挂掉电话时，震动停止。模式的设定和需要震动提醒的号码，如图 6-13 所示。

图 6-13　模式的设定和需要震动提醒的号码

提示：在图 6-13 中，第一幅和第二幅图（从左到右）为设定正常模式和手机的会议（静音）模式，添加需要震动提醒的电话号码，最后一幅图为已设定号码拨入时，手机开始震动。

【实现过程】

在本软件中首先进行判断的是手机当前的状态，若手机处于待机状态时，进入主界面后，可以选择的是由两个 RadioButton 组成的 RadioGroup，为 RadioGroup 添加监听器，若选择正常模式，则设置为正常模式。若单击会议模式，则设定为静音模式。

在 EditText 中填写需要震动提醒的电话号码，单击 Button 按钮将号码存储到列表中，当有电话拨入时，首先检测是否为设定的号码，若是设定的号码，则进行震动提醒。

在本软件中需要添加获取手机状态和手机震动的权限，其权限代码分别为<uses-permission android:name="android.permission.VIBRATE"></uses-permission>和<uses-permission android:name="android.permission.READ_PHONE_STATE"></uses-permission>。

【代码解析】

在本部分将要详细介绍本软件的核心代码，首先介绍的是包含权限设置的 AndroidManifest.xml 的开发，代码如下。

代码位置：见随书光盘中源代码/第 6 章/Sample6_13 目录下的 AndroidManifest.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>                    <!--版本号和编码方式-->
2   <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3                   package="com.bn.chap6.ldjy"
4                   android:versionCode="1"
5       android:versionName="1.0">                            <!--版本名称-->
6     <application android:icon="@drawable/icon" android:label="@string/app_name">
```

```
7              <activity android:name=".Sample6_13_Activity"
8                android:label="@string/app_name">          <!--显示的程序名称-->
9            <intent-filter>
10              <action android:name="android.intent.action.MAIN" />
                                                            <!--标识程序开始-->
11              <category android:name="android.intent.category.LAUNCHER" />
12           </intent-filter>
13         </activity>
14      </application><!--手机震动和获取手机状态的权限设置-->
15       <uses-permission android:name="android.permission.VIBRATE">
</uses-permission>
16       <uses-permission android:name="android.permission.READ_PHONE_STATE">
</uses-permission>
17   </manifest>
```

提示：上述代码中第 15 行为设置手机震动的权限代码，第 16 行为设置获取手机状态的权限代码。

上述代码介绍的是 AndroidManifest.xml 的开发，接下来介绍的是继承 Activity 类的开发，代码如下。

代码位置：见随书光盘中源代码/第 6 章/Sample6_13/src/com/bn/chap6/ldjy 目录下的 Sample6_13_Activity.java。

```
1   package com.bn.chap6.ldjy;                                  //声明包
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3   public class Sample6_13_Activity extends Activity {
4       RadioGroup rg;                                          //RadioGroup 引用
5       RadioButton rbMeeting;                                  //会议模式
6       RadioButton rbNormal;                                   //正常模式
7       Button bAdd;                                            //添加按钮
8       EditText et;                                            //输入框
9       TextView tv;                                            //显示文本框
10      List<String> number;                                    //存放电话号码
11      Vibrator vibrator;                                      //手机震动引用
12      StringBuilder sb=new StringBuilder();                   //StringBuilder 对象
13      int count;                                              //计数器
14      public void onCreate(Bundle savedInstanceState) {
15          super.onCreate(savedInstanceState);
16          setContentView(R.layout.main);                      //设置当前界面
17          rbMeeting=(RadioButton)this.findViewById(R.id.RadioButton01);
                                                                //会议模式
18          rbNormal=(RadioButton)this.findViewById(R.id.RadioButton02);
                                                                //正常模式
19          bAdd=(Button)this.findViewById(R.id.Button01);      //添加按钮
20          et=(EditText)this.findViewById(R.id.EditText01);    //输入框
21          tv=(TextView)this.findViewById(R.id.TextView02);    //显示文本框
22          rg=(RadioGroup)this.findViewById(R.id.RadioGroup01);
                                                                //RadioButton 按钮组
23          number=new ArrayList<String>();                     //创建对象
24          vibrator=(Vibrator)getApplication().getSystemService(
25          Service.VIBRATOR_SERVICE);                          //创建 Vibrator 对象
26          myPhoneStateListener mPSL=new myPhoneStateListener();   //创建对象
27          TelephonyManager tm=(TelephonyManager)this.getSystemService(
28          Context.TELEPHONY_SERVICE);                         //创建对象
29          tm.listen(mPSL, PhoneStateListener.LISTEN_CALL_STATE); //添加监听器
30          bAdd.setOnClickListener(                            //添加监听器
```

```
        new OnClickListener(){
                public void onClick(View v) {
                    String msg=et.getText().toString().trim();
                                                                    //获取电话号码
                    String parent="[0-9]*";                         //匹配字符串
                    if(msg.matches(parent)){
                        number.add(msg);                            //记录电话号码
                        et.setText("");
                        sb.append(count++);                         //编号
                        sb.append(".");                             //组装字符串
                        sb.append(msg);
                        sb.append("\n");
                        tv.setText(sb.toString().trim());           //设置文本
                    }else{
                        Toast.makeText(                             //创建 Toast 对象
                                Sample6_13_Activity.this,
                                "电话号码中只允许有数字",
                                Toast.LENGTH_SHORT                  //显示时长
                                ).show();                           //显示 Toast
}}});
      RadioGroup.OnCheckedChangeListener mChange=                   //创建对象
        new RadioGroup.OnCheckedChangeListener() {
        public void onCheckedChanged(RadioGroup group, int checkedId){
            if(checkedId==rbMeeting.getId()){                       //会议模式
                try{
                    AudioManager audioManager=(AudioManager)
getSystemService(
                            Context.AUDIO_SERVICE);                 //创建对象
                    if(audioManager!=null){                         //判断是否为空
                        audioManager.setRingerMode(
                        AudioManager.RINGER_MODE_SILENT);//设置为静音模式
                        audioManager.getStreamVolume(AudioManager.STREAM_
RING);
                        Toast.makeText(                             //创建 Toast 对象
                            Sample6_13_Activity.this,
                            "手机已设定为静音模式",
                            Toast.LENGTH_SHORT                      //显示时长
                            ).show();                               //显示 Toast
                    }
                }catch(Exception e){e.printStackTrace();}           //打印异常
            }else if(checkedId==rbNormal.getId()){                  //正常模式
                try{
                    AudioManager audioManager=(AudioManager)
                        getSystemService(Context.AUDIO_SERVICE);//创建对象
                    if(audioManager!=null){
                        audioManager.setRingerMode(
                        AudioManager.RINGER_MODE_NORMAL);//设置为正常模式
                        audioManager.getStreamVolume(AudioManager.STREAM_
RING);
                        Toast.makeText(                             //创建对象
                            Sample6_13_Activity.this,
                            "手机已设定为正常模式",
                            Toast.LENGTH_SHORT                      //设置显示时长
                            ).show();
                    }
                }catch(Exception e){e.printStackTrace();}           //打印异常
}}};
      rg.setOnCheckedChangeListener(mChange);                       //添加监听器
}
```

```
86        public class myPhoneStateListener extends PhoneStateListener{
87        ……//该处省略了部分代码，将在后面详细给出。
88    }}
```

其中：

- 第 4～13 行为声明成员变量。
- 第 14～29 行为设置当前界面，创建对象。
- 第 30～49 行为添加按钮监听器，在该监听器内主要完成的是对输入电话号码的检测，若输入的电话号码不全为数字，则提示用户输入正确的电话号码，若输入正确，则重新组装字符串，并将组装的字符串显示在主界面中。
- 第 50～85 行为 RadioGroup 的监听器，首先创建 RadioGroup.OnCheckedChangeListener 对象，重写 onCheckedChanged()方法，在该方法中检测 checkedId，若选中的是会议模式 RadioButton，则设置为静音模式，若选中的是正常模式的 RadioButton，则设置为正常模式。
- 第 86～88 行为内部类 myPhoneStateListener 的实现，该类的详细介绍会在下面给出。

接下来要介绍的是内部类 myPhoneStateListener 的实现，该内部类主要完成的是手机状态发生变化时所完成的业务逻辑，将代码插入到上述代码的第 87 行。

代码位置：见随书光盘中源代码/第 6 章/Sample6_13/src/com/bn/chap6/ldjy 目录下的 Sample6_13_Activity.java。

```
1    public class myPhoneStateListener extends PhoneStateListener{    //内部类
2        public void onCallStateChanged(int state,String incomingNumber){
                                                                    //重写的方法
3            switch(state){
4                case TelephonyManager.CALL_STATE_IDLE:      //待机状态
5                    Toast.makeText(                         //Toast 提示
6                            Sample6_13_Activity.this,
7                            "手机处于待机状态",                  //显示内容
8                            Toast.LENGTH_SHORT              //显示时间
9                            ).show();                       //显示 Toast
10                   if(rbMeeting.isChecked()==true){        //会议模式，设置为静音
11                       try{
12                           AudioManager audioManager=(AudioManager)
13                           getSystemService(Context.AUDIO_SERVICE);//创建对象
14                           if(audioManager!=null){
15                               audioManager.setRingerMode(
16                               AudioManager.RINGER_MODE_SILENT);
                                                            //设置为静音模式
17                               audioManager.getStreamVolume(AudioManager.
STREAM_RING);
18                               Toast.makeText(            //创建 Toast 对象
19                                   Sample6_13_Activity.this,
20                                   "手机已设定为静音模式",
21                                   Toast.LENGTH_SHORT     //显示的时长
22                                   ).show();
23                           }
24                       }catch(Exception e){e.printStackTrace();}   //打印异常
25                   }else{                                 //正常模式，设置为响铃
26                       try{
27                           AudioManager audioManager=(AudioManager)
28                           getSystemService(Context.AUDIO_SERVICE);//创建对象
29                           if(audioManager!=null){
30                               audioManager.setRingerMode(
31                               AudioManager.RINGER_MODE_NORMAL);
```

```
                                                                    //设置为正常模式
32
audioManager.getStreamVolume(AudioManager.STREAM_RING);
33                                  Toast.makeText(                 //创建 Toast 对象
34                                      Sample6_13_Activity.this,
35                                      "手机已设定为正常模式",
36                                      Toast.LENGTH_SHORT          //Toast 显示时长
37                                      ).show();
38                                  }
39                              }catch(Exception e){e.printStackTrace();}   //打印异常
40                          }
41                      break;
42                      case TelephonyManager.CALL_STATE_RINGING:   //响铃状态
43                          if(number.size()==0){                   //若没有记录的号码
44                              return;                             //返回
45                          }
46                          for(int i=0;i<number.size();i++){
47                              if(incomingNumber.equals(number.get(i))){//若拨入电话号
码与列表中号码匹配
48                                  vibrator.vibrate(new  long[]{100,10,100,1000},0);
                                                                    //设置手机震动
49                                  Toast.makeText(
50                                      Sample6_13_Activity.this,
51                                      "手机正在震动",
52                                      Toast.LENGTH_SHORT
53                                      ).show();                   //创建 Toast 显示信息
54                              }}
55                      break;
56                      case TelephonyManager.CALL_STATE_OFFHOOK:   //接听状态
57                          vibrator.cancel();                      //取消震动
58                          Toast.makeText(
59                                  Sample6_13_Activity.this,
60                                  "手机震动已取消",
61                                  Toast.LENGTH_SHORT
62                                  ).show();                       //创建 Toast 显示信息
63                          break;
64                  }
65                  super.onCallStateChanged(state, incomingNumber);//调用父类方法
66  }}
```

其中：

- 第 4～41 行为手机处于待机状态，首先使用 Toast 对用户进行提示，然后检测 RadioGroup 中被选中的 RadioButton，若会议模式被选中，则将手机设置为静音模式；若正常模式被选中，则设置为正常模式。
- 第 42～55 行为手机处于响铃状态，在该状态中，若记录号码的列表为 0，则直接返回，若不为 0 则检测拨入的电话号码是否为需要震动提醒的号码，若是手机震动进行提醒，若不是则手机不进行震动。
- 第 56～65 行为手机处于接听状态，在该状态中将震动取消。第 65 行为调用父类方法。

实例 14　根据手机姿态改变手机模式

在 Android 的 API 中拥有可以判断手机在真实环境里的速度、倾斜、旋转的 SensorManager。在本节将介绍 SensorManager 对手机姿态的检测，并且利用其获取的数据对手机进行模式设置。

【实例描述】

本软件运行时，根据手机的姿态获取手机 pitch 轴的角度，若所得的角度小于–120°，则认为手机现在处于屏幕向下的状态，将手机设置为震动模式；若所得的角度大于–120°，则认为手机现在处于屏幕向上的状态，将手机设置为正常模式。

通过改变手机的姿态便可以轻松设置手机的模式，可以提高用户体验，给用户带来方便。模式的转换如图 6-14 所示。

图 6-14　模式的转换

提示：在图 6-14 中依次为手机屏幕向上，模式为正常模式；手机屏幕向下，模式为震动模式。

【实现过程】

为了可以使主程序在进入时就检测手机的旋转状态，所以在 onResume()方法中使用 SensorListener. registerListener()方法注册自定义的 SersorListener，并且在离开程序时，需要在 onPause()方法中使用 SensorListener. unregisterListener()方法取消自定义的 SersorListener。

当获取手机当前的姿态后，就可以根据其数据使用 AudioManager 对象设置手机的模式，若是手机屏幕向上，则设置为 AudioManager.RINGER_MODE_NORMAL，若手机屏幕向下，则设置为 AudioManager.RINGER_MODE_VIBRATE。

【代码解析】

经过前面的理论介绍相信读者对传感器有了一定的了解，接下来要介绍的是该程序的核心代码 Sample6_14_Activity 类的设计与实现，代码如下。

代码位置：见随书光盘中源代码/第 6 章/Sample6_14/src/com/bn/chap6/sensor 目录下的 Sample6_14_Activity.java。

```
1   package com.bn.chap6.sensor;                    //声明包
```

```
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3   public class Sample6_14_Activity extends Activity { //创建继承 Activity 的类
4       ……//该处省略了部分不重要代码，读者可自行查看随书光盘中源代码。
5      public void onCreate(Bundle savedInstanceState) {     //重写的方法
6         super.onCreate(savedInstanceState);
7         setContentView(R.layout.main);                      //设置当前界面
8         this.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_
PORTRAIT);//强制竖屏
9         etYaw=(EditText)this.findViewById(R.id.EditText01);    //Yaw 轴对象
10        etPitch=(EditText)this.findViewById(R.id.EditText02); //pitch 轴对象
11        etRow=(EditText)this.findViewById(R.id.EditText03);    //row 轴对象
12        tv=(TextView)this.findViewById(R.id.TextView04);
13        sm=(SensorManager)getSystemService(Context.SENSOR_SERVICE);
                                                  //SensorManager 对象
14        try{                                    //获取当前的 AudioManager 模式
15          AudioManager am=(AudioManager)getSystemService(Context.AUDIO_SERVICE);
16          if(am!=null){                                  //若不为空
17              msg=am.getRingerMode();                    //获取状态常数
18          }
19        }catch(Exception e){e.printStackTrace();}      //打印异常
20        switch(msg){
21            case AudioManager.RINGER_MODE_NORMAL:      //正常模式
22              Toast.makeText(                          //创建 Toast 对象
23                          Sample6_14_Activity.this,
24                          "正常模式",
25                          Toast.LENGTH_SHORT           //设置 Toast 显示时长
26                          ).show();
27              tv.setText("手机现在的模式为："+"正常模式");     //设置文本
28            break;
29            case AudioManager.RINGER_MODE_SILENT:          //静音模式
30              Toast.makeText(                          //创建 Toast 对象
31                          Sample6_14_Activity.this,
32                          "静音模式",
33                          Toast.LENGTH_SHORT
34                          ).show();                    //显示 Toast 提示
35              tv.setText("手机现在的模式为："+"静音模式");     //设置文本
36            break;
37            case AudioManager.RINGER_MODE_VIBRATE:         //震动模式
38              Toast.makeText(                          //创建 Toast 对象
39                          Sample6_14_Activity.this,
40                          "震动模式",
41                          Toast.LENGTH_SHORT           //设置 Toast 显示时长
42                          ).show();
43              tv.setText("手机现在的模式为："+"震动模式");     //设置文本
44            break;
45        }}
46    private SensorListener mySensorListener=new SensorListener(){
                                                  //自定义 SensorListener 对象
47          public void onAccuracyChanged(int sensor, int accuracy) {
48          }
49          public void onSensorChanged(int sensor, float[] values) {
50              float yaw=values[SensorManager.DATA_X];     //yaw 轴值的大小
51              float pitch=values[SensorManager.DATA_Y];   //pitch 轴值的大小
52              float row=values[SensorManager.DATA_Z];     //row 轴值的大小
53              etYaw.setText(yaw+"");                      //设置显示的文本
54              etPitch.setText(pitch+"");
55              etRow.setText(row+"");
```

```
56                if(pitch<-120){                          //规定小于-120° 时手机为背面
57                    try{
58                        AudioManager am=(AudioManager) //创建 AudioManager 对象
59                        getSystemService(Context.AUDIO_SERVICE);
60                        if(am!=null){                    //若对象不为空
61
am.setRingerMode(AudioManager.RINGER_MODE_VIBRATE);//震动模式
62                            tv.setText("手机现在的模式为："+"震动模式");
63                        }
64                    }catch(Exception e){e.printStackTrace();}   //打印异常
65                }else{
66                    try{
67                        AudioManager am=(AudioManager) //创建对象
68                        getSystemService(Context.AUDIO_SERVICE);
69                        if(am!=null){
70
am.setRingerMode(AudioManager.RINGER_MODE_NORMAL);//正常模式
71                            tv.setText("手机现在的模式为："+"正常模式");
72                        }
73                    }catch(Exception e){e.printStackTrace();}   //打印异常
74                }}};
75      public void onResume(){                                  //重写 onResume 方法
76        super.onResume();
77        sm.registerListener(                                   //注册自定义的 SersorListener
78            mySensorListener,
79            SensorManager.SENSOR_ORIENTATION,
80            SensorManager.SENSOR_DELAY_NORMAL);
81      }
82      public void onPause(){                                   //重写 onPause 方法
83        super.onPause();
84        sm.unregisterListener(mySensorListener);    //取消注册 SersorListener
85      }
86        public boolean onKeyDown(int keyCode,KeyEvent e){    //重写键按下方法
87        if(keyCode==4){                                      //按下返回键
88            System.exit(0);                                  //退出程序
89        }
90        return true;
91    }}
```

其中：

- 第 6～19 行为创建该类中的对象，其中第 9～12 行为创建 EditText 和 TextView 对象，第 13 行为创建 SensorManager 对象，第 14～19 行为通过 AudioManager 对象获取当前手机的模式状态常数。
- 第 20～45 行为根据获取的模式状态常数判断手机处于何种状态，然后使用 Toast 进行提示，并将 TextView 中的文字设置为当前的模式。
- 第 46～74 行为自定义 SensorListener 对象，在重写的 onSensorChanged 方法中，通过 values 数组获取手机目前姿态的 yaw、pitch、row 轴的值，并根据所获得的 pitch 值的大小设定手机的 AudioManager 模式，笔者规定当 pitch 值小于–120° 时，认为手机屏幕向下，此时设置手机为震动模式，否则设置手机为正常模式。
- 第 75～91 行为重写 onResume、onPause 和 onKeyDown 方法，在 onResume 方法中注册监听手机姿态的 SersorListener 对象，在 onPause 方法中取消该对象。重写 onKeyDown 方法检测是否按下手机的返回键，若按下则退出该程序。

实例 15　定时更改手机模式

生活中有一定的规律可以增加工作效率，在 Android 手机中同样可以设定有规律的工作，这样可以大大减少在手机设置上花费的时间。

【实例描述】

本软件设计简单，在主界面含有两个 Button 按钮，一个为启动按钮，用于设置启动服务的时间，并开启该服务；另一个为关闭按钮，用于关闭服务。当单击启动按钮后，弹出时间对话框设置启动的时间，当时间到达时，服务自动开启。

根据当前时间，设定手机的正常模式、静音模式和震动模式。在该软件中，若时间处于 8 点到 20 点则将手机设置为正常模式，若时间处于 20 点到 24 点则将手机设置为震动模式，若时间处于 0 点到 8 点，则将手机设置为静音模式。该系统每 4 个小时便会获取当前时间，并设置手机为对应模式。定时更改手机模式的运行效果图如图 6-15 所示。

图 6-15　定时更改手机模式运行效果图

提示：在图 6-15 中，第一幅图为开始运行时的主界面，第二幅图为设置开始服务的时间，第三幅图为取消模式切换的服务。

【实现过程】

在本软件中主要运用了 AlarmManager 对象实现手机定时发送广播的功能。在 Button 按钮的 onClick 事件中，创建 Calendar 对象，并弹出时间对话框设置服务开始时间。使用 AlarmManager.setRepeating 方法设置一个会重复调用的 AlarmManager。

每隔 4 个小时唤起自定义的 ModeReceiver，由 ModeReceiver 调用 Sample6_15_Activity 完成手机模式自动切换的功能。

【代码解析】

在本部分将要对本软件的各个类进行一一介绍，首先介绍的是声明 receiver 的 AndroidManifest.xml 的开发，代码如下。

代码位置：见随书光盘中源代码/第 6 章/Sample6_15 目录下的 AndroidManifest.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>                    <!--版本号和编码方式-->
2   <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3               package="com.bn.chap6.wallpaper"
4                   android:versionCode="1"
5               android:versionName="1.0">                    <!--版本名称-->
6     <application android:icon="@drawable/icon" android:label="@string/app_
name">
7               <activity android:name=".Sample6_15_Activity"
8                android:label="@string/app_name">            <!--显示的程序名称-->
9           <intent-filter>
10             <action android:name="android.intent.action.MAIN" />
                                                              <!--标识程序开始-->
11             <category android:name="android.intent.category.LAUNCHER" />
12          </intent-filter>
13        </activity>
14          <receiver android:name=".ModeReceiver"
15               android:process=":remote"></receiver>
16    </application>
17  </manifest>
```

提示：上述代码中第 14、15 行为声明 Receiver，并且将 process 属性设置为 remote。

上述代码介绍的是 AndroidManifest.xml 的开发，接下来要介绍的是 Sample6_15_Activity 类的设计与实现，代码如下。

代码位置：见随书光盘中源代码/第 6 章/Sample6_15/src/com/bn/chap6/wallpaper 目录下的 Sample6_15_Activity.java。

```
1   package com.bn.chap6.wallpaper;
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3   public class Sample6_15_Activity extends Activity {
4       Button bStart;                                        //开始按钮
5       Button bClose;                                        //结束按钮
6       TextView tv1,tv2,tv3,tv4;                             //文本框
7      public void onCreate(Bundle savedInstanceState) {
8         super.onCreate(savedInstanceState);
9         setContentView(R.layout.main);                      //设置当前界面
10        bStart=(Button)this.findViewById(R.id.Button01);    //创建 Button 对象
11        bClose=(Button)this.findViewById(R.id.Button02);
12        tv1=(TextView)this.findViewById(R.id.TextView01);   //创建 TextView 对象
13        tv2=(TextView)this.findViewById(R.id.TextView02);
14        tv3=(TextView)this.findViewById(R.id.TextView03);
15        tv4=(TextView)this.findViewById(R.id.TextView04);
16        Bundle bundle=this.getIntent().getExtras();         //取得 bundle
17        if(bundle!=null){                                   //若 Bundle 不为空
18          Calendar c=Calendar.getInstance();
19          int tempHour=c.get(Calendar.HOUR_OF_DAY);         //获取当前时间
20          if(tempHour>=8&&tempHour<=20){
21              try{                                          //正常模式
22                  AudioManager am=(AudioManager)getSystemService(Context.
AUDIO_SERVICE);
23                  if(am!=null){
```

```
24                          am.setRingerMode(AudioManager.RINGER_MODE_NORMAL);
                                                              //正常模式
25                          Toast.makeText(                   //创建 Toast 对象
26                                  Sample6_15_Activity.this,
27                                  "正常模式",
28                                  Toast.LENGTH_SHORT        //设置显示的时长
29                                  ).show();                 //显示 Toast
30                      }
31                      this.finish();                        //调用 finish 方法
32                  }catch(Exception e){e.printStackTrace();}  //打印异常
33              }else if(tempHour>20&&tempHour<24){
34                  try{                                      //震动模式
35                      AudioManager am=(AudioManager)getSystemService(Context.
AUDIO_SERVICE);
36                      if(am!=null){
37                          am.setRingerMode(AudioManager.RINGER_MODE_VIBRATE);
                                                              //震动模式
38                          Toast.makeText(                   //创建 Toast 对象
39                                  Sample6_15_Activity.this,
40                                  "震动模式",
41                                  Toast.LENGTH_SHORT        //设置显示的时长
42                                  ).show();                 //显示 Toast
43                      }
44                  }catch(Exception e){e.printStackTrace();}  //打印异常
45              }else if(tempHour>=0&&tempHour<8){
46                  try{                                      //静音模式
47                      AudioManager am=(AudioManager)getSystemService(Context.
AUDIO_SERVICE);
48                      if(am!=null){
49                  am.setRingerMode(AudioManager.RINGER_MODE_SILENT);  //静音模式
50                          Toast.makeText(                   //创建 Toast 对象
51                                  Sample6_15_Activity.this,
52                                  "静音模式",
53                                  Toast.LENGTH_SHORT        //设置显示的时长
54                                  ).show();
55                      }
56                  }catch(Exception e){e.printStackTrace();}  //打印异常
57              }}
58          /*该处省略了部分方法，将在下面给出*/
```

其中：

- 第 4～15 行为声明成员变量，并创建对象。
- 第 16～32 行为首先创建 Bundle 对象，并判断 Bundle 是否为空，若不为空，创建 Calendar 对象，并获取当前时间，若时间处于 8 点到 20 点之间，则将手机设置为正常模式。
- 第 33～57 行为根据时间将手机设置为震动模式或静音模式。
- 第 58 行为启动按钮和关闭按钮的监听器，该代码将在后面进行详细介绍。

经过上面对 Sample6_15_Activity 的总体介绍，相信读者对该类有了一定的了解，在下面将对启动按钮和关闭按钮监听器的实现进行详细介绍，将代码插入到上述代码的 58 行。

代码位置：见随书光盘中源代码/第 6 章/Sample6_15/src/com/bn/chap6/wallpaper 目录下的 Sample6_15_Activity.java。

```
1   bStart.setOnClickListener(                                //启动按钮监听器
2     new OnClickListener(){                                  //匿名内部类
3       public void onClick(View v) {                         //重写的方法
4           final Calendar c=Calendar.getInstance();
```

```
5            c.setTimeInMillis(System.currentTimeMillis()); //将当前时间设置为默认时间
6            int hour=c.get(Calendar.HOUR_OF_DAY);          //小时
7            int minute=c.get(Calendar.MINUTE);             //分钟
8            new TimePickerDialog(                          //匿名内部类
9                Sample6_15_Activity.this,
10               new TimePickerDialog.OnTimeSetListener() { //重写的方法
11                   public void onTimeSet(TimePicker view, int hourOfDay, int
minute) {
12                       c.setTimeInMillis(System.currentTimeMillis());
                                                            //设置当前时间
13                       c.set(Calendar.HOUR_OF_DAY, hourOfDay);//设置小时
14                       c.set(Calendar.MINUTE, minute);        //设置分钟
15                       c.set(Calendar.SECOND, 0);             //设置秒
16                       c.set(Calendar.MILLISECOND, 0);        //设置毫秒
17                       Intent intent=new Intent(Sample6_15_Activity.this,
ModeReceiver.class);
18                       PendingIntent pi=PendingIntent.getBroadcast(
                                                    //创建 PendingIntent 对象
19                       Sample6_15_Activity.this, 0, intent, 0);
20                       AlarmManager alarm=(AlarmManager)
                                                    //创建 AlarmManager 对象
21                       Sample6_15_Activity.this.getSystemService(ALARM_
SERVICE);
22                       long time=4*60*60*1000;        //设置时间间隔
23                       alarm.setRepeating(
24                      AlarmManager.RTC_WAKEUP, c.getTimeInMillis(), time, pi);
    //每 4 小时提醒一次 25                    String tempHour=(hourOfDay+"").
length()>1?hourOfDay+"":"0"+hourOfDay;
26                       String tempMinute=(minute+"").length()>1?minute+
"":"0"+minute;
27                       Toast.makeText(                //创建 Toast 对象
28                           Sample6_15_Activity.this,
29                           "设置的时间为："+tempHour+":"+tempMinute,
30                           Toast.LENGTH_SHORT).show();    //显示 Toast
31                       tv4.setText("模式转换设置的开始时间为："+tempHour+":"+
tempMinute);
32                       }
33                   },hour,minute,true).show();        //显示时间对话框
34       }});
35   bClose.setOnClickListener(                         //关闭按钮监听器
36       new OnClickListener(){                         //匿名内部类
37           public void onClick(View v) {              //重写的方法
38           Intent   intent=new   Intent(Sample6_15_Activity.this,ModeReceiver.
class);//运行 ModeReceiver 类
39           PendingIntent pi=PendingIntent.getBroadcast(//创建 PendingIntent 对象
40           Sample6_15_Activity.this, 0, intent, 0);
41           AlarmManager alarm=(AlarmManager)      //创建 AlarmManager 对象
42           Sample6_15_Activity.this.getSystemService(ALARM_SERVICE);
43           alarm.cancel(pi);                          //取消设置
44           Toast.makeText(                            //创建 Toast 对象
45               Sample6_15_Activity.this,
46               "模式切换已取消",
47               Toast.LENGTH_SHORT).show();            //显示 Toast
48       }});
```

其中：

- 第 1～34 行为开始按钮的监听器，在该监听器中首先创建 Calendar 对象，将当前时间设置为默认时间，然后创建时间对话框对象，在时间对话框中进行服务开始时间的设定，

并将提醒时间间隔设置为 4 个小时。

- 第 35～48 行为关闭按钮的监听器，在该监听器内创建 Intent 对象，随后创建 PendingIntent 对象，将 Intent 进行封装，通过 AlarmManager 对象取消设置，最后使用 Toast 进行提示。

上述代码介绍了为本程序的按钮添加监听器，接下来介绍的是 ModeReceiver 类的设计与实现，在该类中主要完成的是获取系统发送的广播，启动 Activity，代码如下。

代码位置：见随书光盘中源代码/第 6 章/Sample6_15/src/com/bn/chap6/wallpaper 目录下的 Sample6_15_Activity.java。

```
1   package com.bn.chap6.wallpaper;                                    //声明包
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3   public class ModeReceiver extends BroadcastReceiver{
4       public void onReceive(Context context, Intent intent) { //重写的方法
5           Intent  tempIntent=new  Intent(context,Sample6_15_Activity.class);
//创建 Intent 对象
6           Bundle myBundle=new Bundle();                          //创建 Bundle 对象
7           myBundle.putString("msg", "msg");
8           tempIntent.putExtras(myBundle);                        //绑定 Bundle
9           tempIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                                                                   //设置新的 task
10          context.startActivity(tempIntent);                     //启动 Actvity
11  }}
```

提示：上述代码主要为获取系统广播，并通过创建的 Intent 对象启动 Activity。

小结

在本章中主要介绍手机的自动服务功能，其主要是对 Service 的实现，需要使用 BroadcastReceiver 对信息进行过滤接收并响应，同时需要在 AndroidManifest.xml 中声明对应权限。通过本章的学习，读者应该可以开发简单的手机自动服务功能。

第 7 章　手机文件 I/O 与数据库的应用

在本章中将详细介绍 Android 平台下文件 I/O 的相关知识以及数据库的应用。下面将通过对几个小软件开发的讲解，向读者介绍具体知识的应用。

实例 1　手机 SD 卡文本阅读器

由于手机的内存有限，所以有时需要将文件存储在 SD 卡中。这就涉及如何在 SD 卡中读取数据。本节主要讲解的是如何在 SD 卡中读取文本文件。

【实例描述】

本软件主要用于阅读手机 SD 卡中存放的 txt 文本文件。在该软件中，用户可以直接输入文件名称，单击“打开”按钮，即可显示相应的文本文件。本实例的运行效果如图 7-1 和图 7-2 所示。

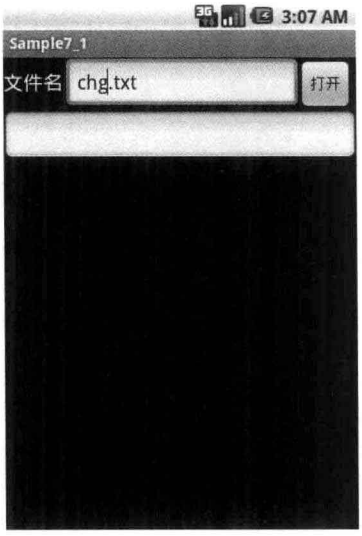


图 7-1　程序运行界面

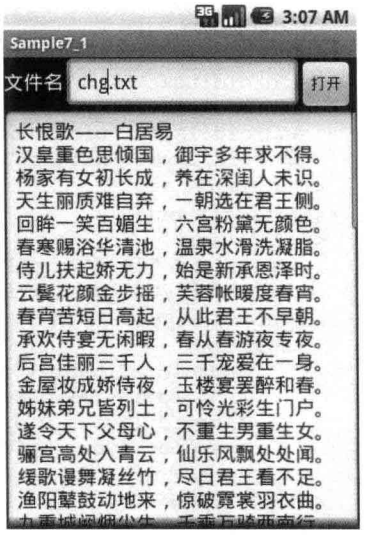


图 7-2　打开文本文件

提示：图 7-1 为程序开始运行时的界面，图 7-2 为打开文本文件，显示文本的界面。

【实现过程】

在本软件开发过程中，主要运用了 xml 文件界面生成技术和文件 I/O 操作。

【代码解析】

在本部分会详细介绍该软件的实现过程。首先介绍的是该软件中 main.xml 文件的设置，代码如下。

代码位置：见随书光盘中源代码/第 7 章/Sample7_1/ res/layout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>                    <!--版本号和编码方式-->
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="vertical"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent"
6      >                                                      <!--LinearLayout-->
7      <LinearLayout
8      android:orientation="horizontal"
9      android:layout_width="fill_parent"
10     android:layout_height="wrap_content"
11     >                                                      <!--LinearLayout-->
12     <TextView
13     android:textSize="18dip"
14     android:textColor="@color/white"
15     android:paddingRight="3dip"
16     android:layout_width="wrap_content"
17     android:layout_height="wrap_content"
18     android:text="文件名"
19     />                                                     <!--TextView 控件-->
20     <EditText
21     android:text="ppx.txt"
22     android:id="@+id/EditText01"
23     android:layout_width="210dip"
24     android:layout_height="wrap_content">
25    </EditText>                                             <!--EditText 控件-->
26    <Button
27      android:text="打开"
28      android:id="@+id/Button01"
29      android:layout_width="wrap_content"
30      android:layout_height="wrap_content">
31    </Button>                                               <!--Button 控件-->
32    </LinearLayout>
33    <ScrollView
34      android:id="@+id/ScrollView01"
35      android:layout_width="fill_parent"
36      android:layout_height="wrap_content">                 <!--ScrollView 控件-->
37      <EditText
38          android:editable="false"
39          android:id="@+id/EditText02"
40          android:layout_width="fill_parent"
41          android:layout_height="wrap_content">
42       </EditText>                                          <!--EditText 控件-->
43    </ScrollView>
44  </LinearLayout>
```

提示： 上述 main.xml 的代码主要是搭建程序开始运行的主界面。首先在 LinearLayout 中设置摆放方式为竖直摆放，然后添加相应的布局以及控件。

上面已经介绍完本程序主界面的搭建。接下来将要介绍的是单击“打开”按钮可以打开对应的文件，并在文本框中显示的功能，代码如下。

代码位置：见本书随书光盘中源代码/第 7 章/Sample7_1/src/com/bn/ex7a 目录下的

Sample7_1_Activity。

```
1   package com.bn.ex7a;                                          //声明包
2   ……//该处省略了部分类的导入代码，读者可自行查阅随书光盘源代码
3   import android.widget.Toast;                                  //导入相关类
4   public class Sample7_1_Activity extends Activity{    //创建继承 Activity 的类
5      @Override
6      public void onCreate(Bundle savedInstanceState) {     //继承类需重写的方法
7         super.onCreate(savedInstanceState);                    //调用父类
8         setContentView(R.layout.main);                         //显示主界面
9         Button b=(Button)findViewById(R.id.Button01);          //获取按钮对象
10        b.setOnClickListener(                                  //为按钮添加监听器
11          new OnClickListener(){                               //匿名内部类
12              @Override
13              public void onClick(View v){                     //实现监听器中的方法
14                  EditText etFileName=(EditText)findViewById(R.id.
EditText01);      //获取文本框对象
15                  String contentStr=loadFromSDFile(etFileName.getText().
toString()); //获取文件内容
16                EditText etContent=(EditText)findViewById(R.id.EditText02);
                                                               //获取文本框对象
17                  etContent.setText(contentStr);             //为文本框添加内容
18      }});}
19     public String loadFromSDFile(String fname){          //查找文本文件获取内容
20      String result=null;
21      try{
22          File f=new File("/sdcard/"+fname);                 //创建路径
23          int length=(int)f.length();                        //返回该文件的长度
24          byte[] buff=new byte[length];
25          FileInputStream fin=new FileInputStream(f);        //创建输入流对象
26          fin.read(buff);                                    //从输入流中读取数据
27          fin.close();                                       //关闭输入流
28          result=new String(buff,"UTF-8");                   //转换字符串
29          result=result.replaceAll("\\r\\n","\n");
30      }catch(Exception e){                                   //捕获异常
31          Toast.makeText(this,"对不起，没有找到指定文件!",Toast.LENGTH_SHORT).show();
32      }
33      return result;                                         //返回字符串
34  }}
```

其中：

- 第 10～18 行表示为“打开”按钮添加监听器，单击“打开”按钮获取相应的文件内容，并将获取的内容显示在文本框中。
- 第 19～34 行表示在 SD 卡中查找文件，获取文件内容的方法。得到文件的路径，通过输入流，将读取的数据存入到 byte[]数组中，并关闭输入流。

实例 2　修改手机中的文件

平时我们开发的软件需要对文件进行不同的操作，其中包括改变文件的名称。本节主要介绍如何修改手机文件的名称。

【实例描述】

运行本程序进入主界面，单击“根目录”按钮查询系统的文件名称并显示。在名称界面单

击文件名称，如果存在子目录则进入子目录，否则弹出修改文件名称对话框。在对话框中输入将要修改的名称，单击“确定”按钮可以改变文件的名称。单击“上翻”按钮，将跳转到本界面的上一界面。

本实例的运行效果图如图 7-3、图 7-4 和图 7-5 所示。

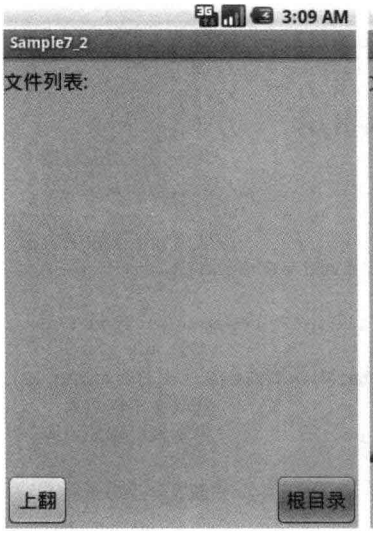


图 7-3　程序运行界面

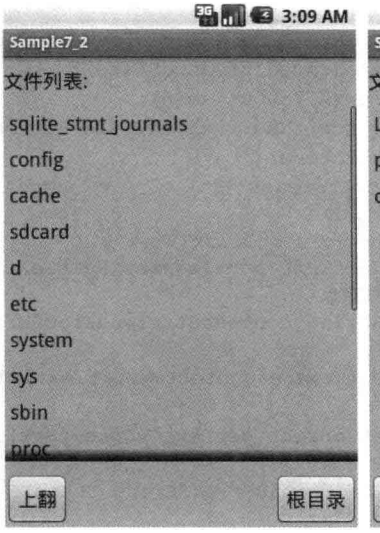


图 7-4　根目录界面

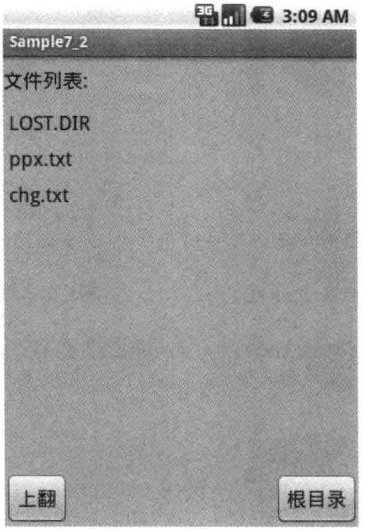


图 7-5　子目录界面

提示：本实例的运行效果如图 7-3、图 7-4 和图 7-5 所示，图 7-3 为程序开始运行时的界面。图 7-4 为单击根目录界面，图 7-5 为文件的子目录界面。

提示：下面的图片为本程序其余的效果图，其中左侧的图 7-6 为对话框中修改文件名称的界面，右侧的图 7-7 为修改名称后的界面。

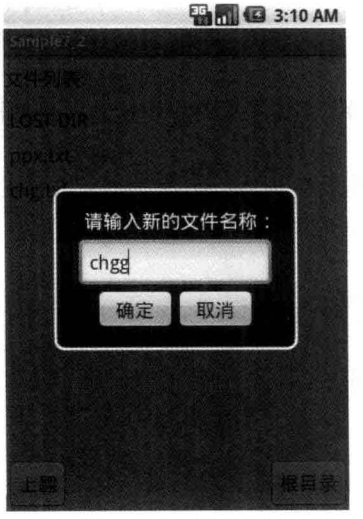


图 7-6　修改文件名称的界面

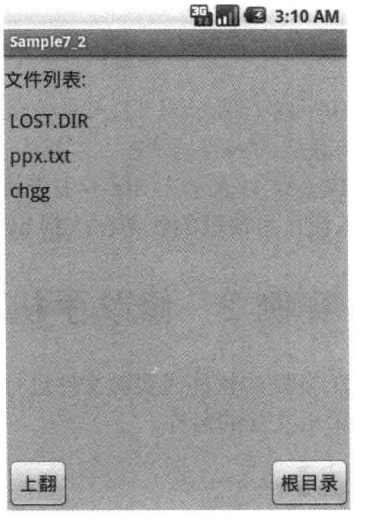


图 7-7　修改名称后的界面

【实现过程】

在本程序的开发中，主要是通过 File 获取对应目录的文件名称。判断 ListView 是否发生单击事件。如果单击没有子目录的文件，则会弹出对话框，并可以更改该文件的名称。

【代码解析】

在本部分会详细介绍本程序的实现过程。首先介绍该软件中 main.xml 文件的设置，其代码如下。

代码位置：见随书光盘中源代码/第 7 章/Sample7_2/ res/layout 目录下的 main.xml。

```
<?xml version="1.0" encoding="utf-8"?>                    <!--版本号和编码方式-->
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:background="#EDAB4A"
   android:paddingTop="10dip">                            <!--FramLayout->
   <LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:gravity="bottom|right">
    <Button                                               <!--Button 控件-->
        android:id="@+id/bgml"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="根目录"
        android:textSize="18dip" />
   </LinearLayout>                                        <!--LinearLayout-->
   <LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:gravity="bottom|left">
    <Button
        android:id="@+id/bsf"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="上翻"
        android:textSize="18dip"/>                        <!--Button 控件-->
   </LinearLayout>
   <LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="360dip"
    android:orientation="vertical">                       <!--LinearLayout-->
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">
        <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="文件列表:"
                android:textSize="18dip"
                android:textColor="#000000"/>             <!--TextView 控件-->
        </LinearLayout>                                   <!--LinearLayout-->
        <LinearLayout
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:paddingTop="10dip">                   <!--LinearLayout-->
            <ListView
```

```
50                  android:id="@+id/lvwjlb"
51                  android:layout_width="fill_parent"
52                  android:layout_height="fill_parent"
53                  android: textSize="18dip"
54                  android:textColor="#000000"
55                  android:divider="#EDAB4A"/>          <!-- ListView 控件-->
56          </LinearLayout>                              <!--LinearLayout-->
57      </LinearLayout>
58  </FrameLayout>                                       <!--FrameLayout-->
```

其中：

- 第 2～8 行表示的是设置布局为帧布局，并且设置控件的排列方式为竖直排列。
- 第 9～19 行表示的是设置一个 LinearLayout 布局，其控件的摆放位置为右侧下方。在其中添加一个 Button 控件，并设置其属性。
- 第 20～30 行表示的是设置一个 LinearLayout 布局，其控件的摆放位置为左侧下方。在其中添加一个 Button 控件，并设置其属性。
- 第 31～57 行表示的是再次设置一个 LinearLayout 布局，其排列方式为竖直排列，在其中添加两个 LinearLayout 布局，在每个布局中添加控件，并设置其具体属性。

上面已经介绍完本程序主界面的搭建。接下来主要介绍主控制类的框架，了解该框架有助于读者以后学习该程序，代码如下。

代码位置：见本书随书光盘中源代码/第 7 章/Sample7_2/src/com/bn/ex7b 目录下的 Sample7_2_Activity。

```
1   package com.bn.ex7b;                                          //声明包
2   ……//该处省略了部分类的导入代码，读者可自行查阅随书光盘源代码
3   import android.widget.AdapterView.OnItemClickListener;   //导入相关类
4   public class Sample7_2_Activity extends Activity{    //创建继承 Activity 的类
5      String currentPath;                                        //记录当前文件列表的父路径
6      String rootPath="/";                                       //根目录
7      String leavePath;                                          //叶子文件
8      Dialog gmDialog;                                           //声明改名对话框
9      ListView lv;                                               //ListView 控件对象声明
10     @Override
11     public Dialog onCreateDialog(int id){                      //创建对话框的方法
12      Dialog result=null;                                       //Dialog 为空
13      switch(id){                                               //判断 id
14          case 0:                                               //id 为 0
15           AlertDialog.Builder b=new AlertDialog.Builder(this);
                                                                  //建立 AlertDialog
16              b.setItems(null,null);
17              b.setCancelable(false);
18              gmDialog=b.create();                              //创建 Dialog
19              result=gmDialog;                                  //返回 Dialog
20          break;                                                //退出
21      }
22      return result;                                            //返回 Dialog
23     }
24     @Override
25     public void onPrepareDialog(int id, final Dialog dialog){ //创建对话框
26      switch(id){
27          case 0:                                                   //id 为 0
28              dialog.setContentView(R.layout.dialog);               //跳转到界面
29              Button bok=(Button)dialog.findViewById(R.id.bOk); //创建 Button 引用
30              Button bcancel=(Button)dialog.findViewById(R.id.bCancle);
                                                                  //创建 Button 引用
```

```
31              final EditText et=(EditText)dialog.findViewById(R.id.et);
                                                        //创建 EditText 引用
32              bok.setOnClickListener(                 //为按钮添加监听器
33                  new OnClickListener(){              //匿名内部类
34                      @Override
35                      public void onClick(View arg0){ //实现监听器中的方法
36                          String newName=et.getText().toString().trim();
                                                        //得到名称字符串
37                          File xgf=new File(leavePath); //得到路径
38                          String newPath=xgf.getParentFile().getPath()+"/"+
newName;
39                          xgf.renameTo(new File(newPath));   //改变名称
40                          final File[] files=getFiles(currentPath);
                                                        //获取根节点文件列表
41                          intoListView(files,lv);    //为 ListView 设置适配器
42                          dialog.cancel();            //退出对话框
43              }});
44              bcancel.setOnClickListener(             //为按钮添加监听器
45                  new OnClickListener(){              //匿名内部类
46                      @Override
47                      public void onClick(View arg0){ //实现监听器中的方法
48                          dialog.cancel();            //退出对话框
49              }});
50              dialog.setCancelable(true);
51          break;                                      //退出
52      }}
53      @Override
54      public void onCreate(Bundle savedInstanceState){    //弹出对话框时被回调
55          super.onCreate(savedInstanceState);             //调用父类
56          setContentView(R.layout.main);                  //显示主界面
57          lv=(ListView)Sample7_2_Activity.this.findViewById(R.id.lvwjlb);
                                                        //获取 ListView 控件对象
58          Button bgml=(Button)this.findViewById(R.id.bgml); //搜索根目录文件按钮
59          Button bsf=(Button)this.findViewById(R.id.bsf);   //搜索父目录文件按钮
60          bgml.setOnClickListener(                        //为按钮添加监听器
61              new OnClickListener(){                      //匿名内部类
62                      @Override
63                      public void onClick(View v){        //实现监听器中的方法
64                          currentPath=rootPath;
65                          final File[] files=getFiles(currentPath);
                                                        //获取根节点文件列表
66                          intoListView(files,lv);    //文件添加到 ListView 列表中
67              }});
68          bsf.setOnClickListener(                         //为按钮添加监听器
69              new OnClickListener(){                      //匿名内部类
70                  @Override
71                  public void onClick(View v){            //实现监听器中的方法
72                      if((currentPath!=null)&&(!currentPath.equals(rootPath))){
                                                    //当前父路径是不是 rootPath
73                          File cf=new File(currentPath);  //获取路径对应的文件
74                          cf=cf.getParentFile();          //获取父目录文件
75                          currentPath=cf.getPath();       //记录当前文件列表路径
76                          intoListView(getFiles(currentPath),lv);
                                                    //文件添加到 ListView 列表中
77          }}});}
78      public File[] getFiles(String filePath)             //得到文件名称的方法
79      {
```

```
80        File[] files=new File(filePath).listFiles();          //获取当前目录下的文件
81        return files;                                         //返回名称
82      }
83      /*该处省略了创建适配器并未 ListView 添加适配器的代码，将在下面给出*/
84  }
```

其中：

- 第 11～23 行表示重写的创建 Dialog 方法，要弹出对话框必须重写 onCreateDialog 方法。
- 第 25～52 表示每次弹出对话框时被回调，以动态更新对话框内容的方法。
- 第 60～67 行表示为搜索根目录文件按钮添加监听器。
- 第 68～77 行表示为搜索父目录文件按钮添加监听器。
- 第 78～82 行表示获取当前目录下的文件名称，并返回文件名称数组。

上面已经介绍了主控制类的框架，接下来将要介绍的是如何设置适配器，并为 ListView 添加适配器的代码，将下列代码插入到上面主控制类框架的第 83 行。

代码位置：见本书随书光盘中源代码/第 7 章/Sample7_2/src/com/bn/ex7b 目录下的 Sample7_2_Activity。

```
1   public void intoListView(final File[] files,final ListView lv){
                                                              //自定义的方法
2       if(files!=null){                                      //判断 file 是否为空
3        if(files.length==0){                                 //当前目录为空
4            File cf=new File(currentPath);                 //获取当前文件列表的路径
5                cf=cf.getParentFile();                       //获取父目录文件
6                currentPath=cf.getPath();                    //记录当前文件列表路径
7                Toast.makeText(Sample7_2_Activity.this,      //弹出 Toast
8                            "该文件夹为空!! ",
9                            Toast.LENGTH_SHORT).show();
10      }else{
11           BaseAdapter ba=new BaseAdapter(){               //创建适配器
12               @Override
13               public int getCount() {                     //重写的方法
14                   return files.length;                    //数组长度
15               }
16               @Override
17               public Object getItem(int position) {       //重写的方法
18                   return null;                            //返回空
19               }
20               @Override
21               public long getItemId(int position) {       //重写的方法
22                   return 0;                               //返回 0
23               }
24               @Override
25               public View getView(int arg0, View arg1, ViewGroup arg2) {
                                                              //重写的方法
26                   LinearLayout ll=new LinearLayout(Sample7_2_Activity.this);
                                                              //定义 LinearLayout
27                   ll.setOrientation(LinearLayout.VERTICAL);  //竖直排列
28                   ll.setPadding(5, 5, 5, 5);                 //留白
29                   TextView tv=new TextView(Sample7_2_Activity.this);
                                                              //初始化 TextView
30                   tv.setTextColor(Color.BLACK);           //设置字体颜色
31                   tv.setText(files[arg0].getName());      //添加文件名称
32                  tv.setGravity(Gravity.LEFT);             //左对齐
```

```
33                  tv.setTextSize(18);                          //字体大小
34                  ll.addView(tv);                  //LinearLayout 添加 TextView
35                   return ll;
36              }};
37      lv.setAdapter(ba);                           //设置适配器
38      lv.setOnItemClickListener(                   //添加监听
39         new OnItemClickListener(){                //匿名内部类
40            @Override
41            public void onItemClick(AdapterView<?> arg0, View arg1,int arg2, long
arg3) {//重写的方法
42              File f=new File(files[arg2].getPath());      //获得当前单击的文件对象
43                  if(f.isDirectory()){
44                      currentPath=files[arg2].getPath(); //设置当前的路径
45                      File[] fs=getFiles(currentPath);   //获取当前路径下所有子文件
46                      intoListView(fs,lv);             //将文件列表填入 ListView 中
47                  }else{                                 //弹出对话框填写新的文件名
48                      leavePath=f.getPath();             //设置路径
49                      showDialog(0);
50       }}});}}else{
51       File cf=new File(currentPath);                    //文件列表路径对应的文件
52            cf=cf.getParentFile();                       //获取父目录文件
53            currentPath=cf.getPath();                    //记录当前文件列表路径
54            Toast.makeText(Sample7_2_Activity.this,      //返回 Toast
55                           "该文件夹为空!! ",
56                           Toast.LENGTH_SHORT).show();
57   }}
```

其中：

- 第 11～36 行表示创建适配器，并设置适配器中各个控件的排列顺序。
- 第 37 行表示为 ListView 设置适配器。
- 第 38～50 行表示为 ListView 列表添加监听器，判断是否发生单击事件。

实例 3　删除手机中的文件

在本节中主要介绍如何在不进入 SD 卡的情况下，删除手机中的文件。

【实例描述】

运行本程序进入主界面。单击“文件搜索”按钮，查找 SD 卡中的所有文件，在该按钮上侧的文本框中显示查找的结果。在“文件搜索”按钮下面的文本框中输入要删除的文件名称，单击“删除”按钮，即可以实现删除文件的功能。

本实例的运行效果如图 7-8、图 7-9 和图 7-10 所示。

提示：在本程序的运行效果图 7-8、图 7-9 和图 7-10 中，图 7-8 为程序开始运行时的界面，图 7-9 为搜索结果的界面，图 7-10 为删除文件后的界面。

【实现过程】

在本程序的开发中，主要用到的依然是文件 I/O 操作，不过本程序主要是读取文件的名称。

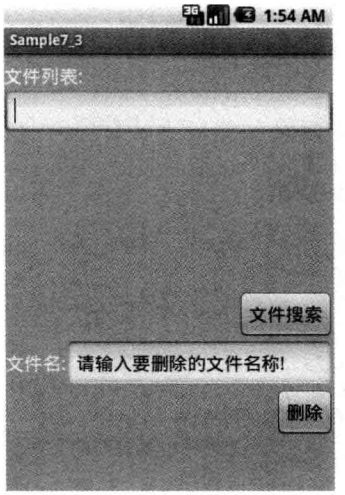


图 7-8　程序运行界面

图 7-9　文件列表

图 7-10　删除文件

【代码解析】

在本部分会详细介绍本程序的实现过程。首先介绍的是本程序中 mian.xml 文件的设置，代码如下。

代码位置：见随书光盘中源代码/第 7 章/Sample7_3/ res/layout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>                    <!--版本号和编码方式-->
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="vertical"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent"
6      android:background="#EDAB4A"
7      android:paddingTop="10dip">                            <!--LinearLayout-->
8      <LinearLayout
9       android:layout_width="fill_parent"
10      android:layout_height="wrap_content"
11      android:orientation="vertical">
12      <TextView
13              android:layout_width="wrap_content"
14              android:layout_height="wrap_content"
15              android:text="文件列表:"android:textSize="18dip"
16              android:textColor="#00FF00"/>                  <!--TextView 控件-->
17      </LinearLayout>                                        <!--LinearLayout-->
18      <LinearLayout
19          android:layout_width="fill_parent"
20          android:layout_height="200dip">                    <!--LinearLayout-->
21          <ScrollView
22              android:layout_width="fill_parent"
23              android:layout_height="fill_parent">           <!--ScrollView 控件-->
24              <EditText
25                  android:layout_width="fill_parent"
26                  android:layout_height="fill_parent"
27                  android:textSize="18dip"android:id="@+id/etwjlb"
28                  android:editable="false"/>                 <!--TextView 控件-->
29          </ScrollView>                                      <!--ScrollView 控件-->
30      </LinearLayout>
31     <LinearLayout
32      android:layout_width="fill_parent"
33      android:layout_height="wrap_content"
```

```
34      android:gravity="right">                              <!--LinearLayout-->
35      <Button
36          android:id="@+id/bss"
37          android:layout_width="wrap_content"
38          android:layout_height="wrap_content"
39          android:text="文件搜索"android:textSize="18dip"/><!--TextView 控件-->
40      </LinearLayout>                                       <!--LinearLayout-->
41      <LinearLayout
42          android:layout_width="fill_parent"
43          android:layout_height="wrap_content"
44          android:orientation="horizontal">                 <!--LinearLayout-->
45          <TextView
46              android:layout_width="wrap_content"
47              android:layout_height="wrap_content"
48              android:text="文件名:"android:textSize="18dip"
49              android:textColor="#00FF00"/>                 <!--TextView 控件-->
50          <EditText
51              android:id="@+id/etwjmc"
52              android:layout_width="260dip"
53              android:layout_height="wrap_content"
54              android:text="请输入要删除的文件名称!"
55              android:textSize="18dip"android:singleLine="true"/>
                                                              <!--EditText 控件-->
56      </LinearLayout>
57      <LinearLayout
58      android:layout_width="fill_parent"
59      android:layout_height="wrap_content"
60      android:gravity="right">                              <!--LinearLayout-->
61      <Button
62          android:id="@+id/bsc"
63          android:layout_width="wrap_content"
64          android:layout_height="wrap_content"
65          android:text="删除"android:textSize="18dip"/>    <!--Button 控件-->
66      </LinearLayout>                                       <!--LinearLayout-->
67  </LinearLayout>
```

提示：上述 main.xml 的代码主要是搭建程序的主界面。首先在 LinearLayout 中设置摆放方式为竖直摆放，然后添加相应的布局，并在每个布局中添加相应的控件。

上面已经介绍完本程序主界面的搭建。接下来介绍的是实现单击“删除”按钮，可以删除相应文件的功能，代码如下。

代码位置：见本书随书光盘中源代码/第 7 章/Sample7_3/src/com/bn/ex7c 目录下的 Sample7_3_Activity。

```
1   package com.bn.ex7b;                                      //声明包
2   ……//该处省略了部分类的导入代码，读者可自行查阅随书光盘源代码
3   import android.widget.EditText;                           //导入相关类
4   public class Sample7_2_Activity extends Activity{    //创建继承 Activity 的类
5      @Override
6      public void onCreate(Bundle savedInstanceState){       //继承类需重写的方法
7         super.onCreate(savedInstanceState);                 //调用父类
8         setContentView(R.layout.main);                      //显示主界面
9         final EditText etwjlb=(EditText)this.findViewById(R.id.etwjlb);
                                                              //文件列表显示文本框
10        final EditText etwjmc=(EditText)this.findViewById(R.id.etwjmc);
                                                              //文件名称输入框
11        Button bss=(Button)this.findViewById(R.id.bss);     //搜索按钮
12        Button bsc=(Button)this.findViewById(R.id.bsc);     //删除按钮
```

```
13          bss.setOnClickListener(                                  //为按钮添加监听器
14            new OnClickListener(){
15                    public void onClick(View v){                   //为按钮添加监听器
16                        String f=getFiles();                       //获取文件
17                        etwjlb.setText(f);                         //显示文件
18            }});
19          bsc.setOnClickListener(
20            new OnClickListener(){                                 //匿名内部类
21                    @Override
22                    public void onClick(View v){                   //为按钮添加监听器
23                        File f=new File("/sdcard/"+etwjmc.getText().toString().
trim());//获取当前文件
24                        f.delete();                                //删除该文件
25                        String ff=getFiles();                      //获取文件
26                        etwjlb.setText(ff);                        //显示文件
27        }});}
28       public String getFiles(){
29         String result="";                                         //自定义字符串
30         File[] files=new File("/sdcard").listFiles();             //得到文件名称列表
31         if(files==null){                                          //判断列表是否为空
32             result="当前没有文件，无法进行删除操作";                //字符串存入数据
33         }else{
34             for(File f:files){                                    //增强 for 循环
35                 result=result+f.getName()+"\n";                   //字符串存入数据
36             }
37         }
38         return result;                                            //返回字符串
39   }}
```

其中：

- 第 13～18 行表示为“文件搜索”按钮添加监听，单击该按钮查询 SD 卡中的文件名，并将得到的数据显示在文本框中。
- 第 19～27 行表示为“删除”按钮添加监听，单击该按钮，将在 SD 卡中删除相应的文件，以及更新上面显示的 SD 卡文件名的数据。
- 第 28～39 行表示搜索 SD 卡中所有文件的名称。

实例 4　访问 APK 包中的文件

很多应用程序不仅需要从手机上读取文件，有时也需要访问 APK 包中自带的文件。本节主要介绍如何访问 APK 中自带的文件。

【实例描述】

本程序主要用于访问程序 APK 包中存放的文件。在本程序中，用户可以直接输入文件名称，单击“打开”按钮，即可阅读当前文件。

本实例的运行效果如图 7-11 和图 7-12 所示。

提示：在本程序的运行效果图 7-11 和图 7-12 中，图 7-11 表示程序开始运行的界面，图 7-12 表示查看 APK 中的某一文件的界面。

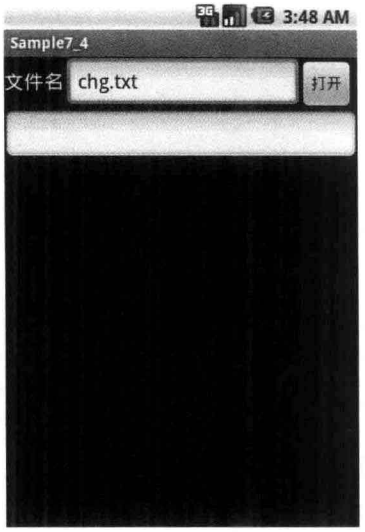


图 7-11　程序运行界面

图 7-12　访问文本文件

【实现过程】

在本程序中主要用了在 xml 中搭建界面的技术以及文件的 I/O 操作。

【代码解析】

在本部分会详细介绍本程序的实现过程。首先介绍本程序中 main.xml 文件的设置，其代码如下。

代码位置：见随书光盘中源代码/第 7 章/Sample7_4/ res/layout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>                    <!--版本号和编码方式-->
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="vertical"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent"
6      >                                                      <!--LinearLayout-->
7      <LinearLayout
8      android:orientation="horizontal"
9      android:layout_width="fill_parent"
10     android:layout_height="wrap_content"
11     >                                                      <!--LinearLayout-->
12     <TextView
13     android:textSize="18dip"
14     android:textColor="@color/white"
15     android:paddingRight="3dip"
16     android:layout_width="wrap_content"
17     android:layout_height="wrap_content"
18     android:text="文件名"
19     />                                                     <!--TextView 控件-->
20     <EditText
21     android:text="chg.txt"
22     android:id="@+id/EditText01"
23     android:layout_width="210dip"
24     android:layout_height="wrap_content">
25    </EditText>                                             <!--EditText 控件-->
26    <Button
27      android:text="打开"
```

```
    android:id="@+id/Button01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
  </Button>                                          <!--Button 控件-->
  </LinearLayout>                                    <!--LinearLayout-->
  <ScrollView
    android:id="@+id/ScrollView01"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">            <!--ScrollView 控件-->
    <EditText
        android:editable="false"
        android:id="@+id/EditText02"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">        <!--EditText 控件-->
     </EditText>                                     <!--EditText 控件-->
  </ScrollView>                                      <!--ScrollView 控件-->
</LinearLayout>
```

提示：上述 main.xml 的代码主要是搭建程序的主界面。首先在 LinearLayout 中设置摆放方式为竖直摆放，然后添加相应的布局以及相应的控件。

上面已经介绍完本程序主界面的搭建。接下来介绍的是实现单击“打开”按钮，打开 APK 中相应的文件并显示的功能，代码如下。

代码位置：见本书随书光盘中源代码/第 7 章/Sample7_4/src/com/bn/ex7d 目录下的 Sample7_4_Activity。

```
package com.bn.ex7d;                                          //声明包
……//该处省略了部分类的导入代码，读者可自行查阅随书光盘源代码
import android.widget.Toast;                                  //导入相关类
public class Sample7_4_Activity extends Activity{    //创建继承 Activity 的类
   @Override
   public void onCreate(Bundle savedInstanceState){ //继承类需重写的方法
      super.onCreate(savedInstanceState);                     //调用父类
      setContentView(R.layout.main);                          //显示主界面
      Button b=(Button)findViewById(R.id.Button01);
      b.setOnClickListener(                                   //对按钮添加监听
        new OnClickListener(){                                //匿名内部类
            @Override
            public void onClick(View v){                  //监听需要重写的方法
                EditText etFileName=(EditText)findViewById(R.id.
EditText01); //得到 EditText01 引用
                String contentStr=loadFromSDFile(etFileName.getText().
toString());
               EditText etContent=(EditText)findViewById(R.id.EditText02);
                                                          //得到 EditText02 引用
                etContent.setText(contentStr);            //设置内容
    }});}
   public String loadFromSDFile(String fname){
    String result=null;                                   //字符串 result 为空
    try{
        InputStream in=this.getResources().getAssets().open(fname);
                                                          //输入流
         int ch=0;                                        //变量 ch
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
                                                          //输出流
        while((ch=in.read())!=-1){                        //读取的值是否为-1
            baos.write(ch);                               //写入输出流
```

```
27                }
28                byte[] buff=baos.toByteArray();
29                baos.close();                                    //关闭输出流
30                in.close();                                      //关闭输入流
31            result=new String(buff,"UTF-8");                     //数组转换格式
32            result=result.replaceAll("\\r\\n","\n");             //调用 replace()方法
33        }catch(Exception e){                                     //捕获异常
34            Toast.makeText(this,"对不起,没有找到指定文件!",Toast.LENGTH_SHORT).show();
35        }
36        return result;                                           //返回 result 字符串
37    }}
```

其中：

- 第 10～18 行表示为“打开”按钮设置监听，同时得到要打开文件的内容，使该内容在 EditText02 代表的文本框中显示。
- 第 19～37 行表示得到文件名，在 APK 中查找相应的文件，并将文件中的内容输出，如果未查找到，则会弹出 Toast 提示信息。

实例 5　简单的学生信息管理

当应用程序需要处理的数据量比较大时，为了方便用户对数据的管理，通常需要使用数据库来存储数据。在本节中将要介绍的是在手机端使用数据库管理用户的数据。

【实例描述】

本程序主要是对学生信息的管理。在本程序中可以创建或者打开数据库，并对该数据库中的数据实现简单的增加、删除、修改与查询。同时可以关闭该数据库，使得在不打开数据库的情况下，无法对该数据库中的数据进行基本的操作。

本实例的运行效果如图 7-13、图 7-14 和图 7-15 所示。

图 7-13　程序运行界面

图 7-14　打开数据库

图 7-15　插入记录

在本程序的运行效果图 7-13、图 7-14 和图 7-15 中，图 7-13 为程序开始运行时的界面。图 7-14 为打开数据库的界面，图 7-15 为在数据库中插入记录的界面。

提示：在本程序其余的运行效果图 7-16、图 7-17 和图 7-18 中，图 7-16 为更新数据库中记录的界面，图 7-17 为删除数据库中记录的界面，图 7-18 为关闭数据库的界面。

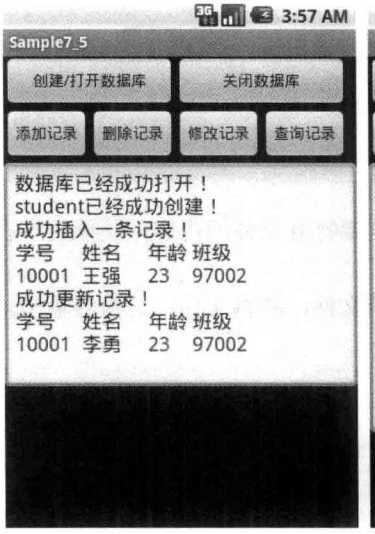


图 7-16 更新记录的界面

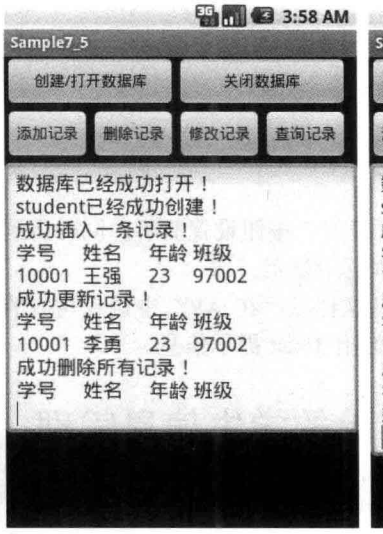


图 7-17 删除记录的界面

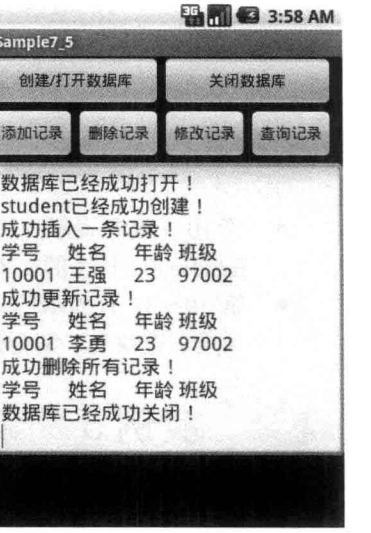


图 7-18 关闭数据库的界面

【实现过程】

在本程序中用到的数据库为 Android 系统自带的 Sqlite 数据库，在创建或者打开数据库时需要调用 SQLiteDatabase.openDatabase()方法。在执行增加、删除与修改语句时需要调用 execSQL()方法，在执行查询操作时需要调用 rawQuery()方法。

【代码解析】

在本部分会详细介绍本程序的实现过程。首先介绍的是本程序中 main.xml 文件的设置，代码如下。

代码位置：见随书光盘中源代码/第 7 章/Sample7_5/ res/layout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>                    <!--版本号和编码方式-->
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="vertical"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent"
6      >                                                      <!--LinearLayout-->
7      <LinearLayout
8      android:orientation="horizontal"
9      android:layout_width="fill_parent"
10     android:layout_height="wrap_content"
11     >                                                      <!--LinearLayout-->
12         <Button
13           android:text="创建/打开数据库"
14           android:id="@+id/Button01"
15           android:layout_width="156dip"
16           android:layout_height="wrap_content">
17         </Button>                                          <!--Button 控件-->
18         <Button
```

```
19          android:text="关闭数据库"
20          android:id="@+id/Button02"
21          android:layout_width="156dip"
22          android:layout_height="wrap_content">
23        </Button>                                    <!--Button 控件-->
24    </LinearLayout>
25    <LinearLayout
26    android:orientation="horizontal"
27    android:layout_width="fill_parent"
28    android:layout_height="wrap_content"
29    >                                                <!--LinearLayout-->
30        <Button
31          android:text="添加记录"
32          android:id="@+id/Button03"
33          android:layout_width="wrap_content"
34          android:layout_height="wrap_content">
35        </Button>                                    <!--Button 控件-->
36        <Button
37          android:text="删除记录"
38          android:id="@+id/Button04"
39          android:layout_width="wrap_content"
40          android:layout_height="wrap_content">
41        </Button>                                    <!--Button 控件-->
42            <Button
43          android:text="修改记录"
44          android:id="@+id/Button05"
45          android:layout_width="wrap_content"
46          android:layout_height="wrap_content">
47        </Button>                                    <!--Button 控件-->
48        <Button
49          android:text="查询记录"
50          android:id="@+id/Button06"
51          android:layout_width="wrap_content"
52          android:layout_height="wrap_content">
53        </Button>                                    <!--Button 控件-->
54    </LinearLayout>                                  <!--LinearLayout-->
55    <ScrollView
56      android:id="@+id/ScrollView01"
57      android:layout_width="fill_parent"
58      android:layout_height="wrap_content">          <!--ScrollView 控件-->
59      <EditText
60          android:id="@+id/EditText01"
61          android:layout_width="fill_parent"
62          android:layout_height="wrap_content">
63       </EditText>                                   <!--EditText 控件-->
64    </ScrollView>                                    <!--ScrollView 控件-->
65  </LinearLayout>
```

其中：

- 第 2～6 行表示设置界面中控件总体的排放顺序。
- 第 7～24 行表示设置一个 LinearLayout 布局，其控件摆放顺序为水平摆放，在其中添加两个 Button，并为这两个控件设置属性。
- 第 25～54 行表示再次设置 LinearLayout 布局，其控件摆放顺序为水平摆放，在其中添加四个 Button，并为这四个控件设置属性。
- 第 55～64 行表示为在界面添加 ScrollView 控件，并设置其属性，在其中添加 EditText 控件，并设置 EditText 控件的属性。

上面已经介绍了本程序主界面的搭建，接下来将要介绍的是本程序主控制类的框架，了解

该框架有助于读者以后学习本程序，代码如下。

代码位置：见本书随书光盘中源代码/第 7 章/Sample7_5/src/com/bn/ex7e 目录下的 Sample7_5_Activity。

```
1   package com.bn.ex7e;                                          //声明包
2   ……//该处省略了部分类的导入代码，读者可自行查阅随书光盘源代码
3   import android.widget.Toast;                                  //导入相关类
4   public class Sample7_5_Activity extends Activity{    //创建继承 Activity 的类
5       SQLiteDatabase sld;                                       //成员变量
6      @Override
7      public void onCreate(Bundle savedInstanceState){           //继承类需重写的方法
8         super.onCreate(savedInstanceState);                    //调用父类
9         setContentView(R.layout.main);                         //显示主界面
10        Button b=(Button)this.findViewById(R.id.Button01);
11        b.setOnClickListener(                                  //对按钮添加监听
12          new OnClickListener(){                               //匿名内部类
13               @Override
14               public void onClick(View v) {                   //监听需要重写的方法
15                    createOrOpenDatabase();
16           }});
17        b=(Button)this.findViewById(R.id.Button02);
18        b.setOnClickListener(                                  //对按钮添加监听
19          new OnClickListener(){                               //匿名内部类
20               @Override
21               public void onClick(View v){                    //监听需要重写的方法
22                    closeDatabase();
23           }});
24        b=(Button)this.findViewById(R.id.Button03);
25        b.setOnClickListener(                                  //对按钮添加监听
26          new OnClickListener(){                               //匿名内部类
27               @Override
28               public void onClick(View v){                    //监听需要重写的方法
29                    insert();
30           }});
31        b=(Button)this.findViewById(R.id.Button04);
32        b.setOnClickListener(                                  //对按钮添加监听
33          new OnClickListener(){                               //匿名内部类
34               @Override
35               public void onClick(View v) {                   //监听需要重写的方法
36                    delete();
37           }});
38        b=(Button)this.findViewById(R.id.Button05);
39        b.setOnClickListener(                                  //对按钮添加监听
40          new OnClickListener(){                               //匿名内部类
41               @Override
42               public void onClick(View v) {                   //监听需要重写的方法
43                    update();
44           }});
45        b=(Button)this.findViewById(R.id.Button06);
46        b.setOnClickListener(                                  //对按钮添加监听
47          new OnClickListener(){                               //匿名内部类
48               @Override
49               public void onClick(View v) {                   //监听需要重写的方法
50                    query();                                   //调用 query 方法
51     }});}
52     /*该处省略了对数据库操作的方法，将在下面给出*/
53     public void appendMessage(String msg){
54      EditText et=(EditText)this.findViewById(R.id.EditText01);
```

```
                                                    //创建 EditText01 对象
55        et.append(msg+"\n");                      //将 msg 添加到 et 中
56    }}
```

其中：

- 第 11～16 行表示为“创建或打开数据库”按钮添加监听器，调用 createOrOpenDatabase ()方法。
- 第 18～23 行表示为“关闭数据库”按钮添加监听器，主要调用 closeDatabase()方法。
- 第 25～30 行表示为“插入记录”按钮添加监听器，主要调用 insert()方法。
- 第 32～37 行表示为“删除记录”按钮添加监听器，主要调用 delete()方法。
- 第 39～44 行表示为“更新记录”按钮添加监听器，主要调用 update()方法。
- 第 46～51 行表示为“查找记录”按钮添加监听器，主要调用 query()方法。

上面已经介绍了主控制类的框架，接下来将为读者介绍的是本控制类核心部分的代码，将下列代码插入到上面主控制类框架的第 53 行。

代码位置：见本书随书光盘中源代码/第 7 章/Sample7_5/src/com/bn/ex7e 目录下的 Sample7_5_Activity。

```
1    public void createOrOpenDatabase(){                    //方法声明
2        try{
3            sld=SQLiteDatabase.openDatabase(               //创建或打开数据库
4                "/data/data/com.bn.ex7e/mydb",             //数据库所在路径
5                null,                                      //游标工厂
6                SQLiteDatabase.OPEN_READWRITE|SQLiteDatabase.CREATE_IF_NECESSARY
7            );
8            appendMessage("数据库已经成功打开！");          //调用 appendMessage 方法
9    String  sql="create  table  if  not  exists  student(sno  char(5),stuname
varchar(20),sage integer,sclass char(5))";
10           sld.execSQL(sql);                              //调用 execSQL()方法
11           appendMessage("student 已经成功创建！");        //调用 appendMessage 方法
12       }catch(Exception e){                               //捕获异常
13           Toast.makeText(this,"数据库错误："+e.toString(),Toast.LENGTH_SHORT).
show();
14       }}
15       public void closeDatabase(){                       //方法声明
16        try{
17           sld.close();                                   //关闭数据库
18           appendMessage("数据库已经成功关闭！");          //调用 appendMessage 方法
19        }catch(Exception e){                              //捕获异常
20           Toast.makeText(this,"数据库错误："+e.toString(),Toast.LENGTH_SHORT).
show();;
21        }}
22       public void insert(){                              //方法声明
23        try{
24           String sql="insert into student values('10001','王强',23,'97002')";
                                                           //插入数据的 sql 语句
25           sld.execSQL(sql);                              //调用 execSQL()方法
26           appendMessage("成功插入一条记录！");            //调用 appendMessage 方法
27        }catch(Exception e){                              //捕获异常
28           Toast.makeText(this,"数据库错误："+e.toString(),Toast.LENGTH_SHORT).
show();;
29        }}
30       public void delete(){                              //方法声明
31        try{
32           String sql="delete from student;";             //删除数据的 sql 语句
```

```
33            sld.execSQL(sql);                                    //调用 execSQL()方法
34            appendMessage("成功删除所有记录！");                  //调用 appendMessage 方法
35        }catch(Exception e){                                     //捕获异常
36            Toast.makeText(this,"数据库错误："+e.toString(),Toast.LENGTH_SHORT).
show();;
37        }}
38        public void update(){                                    //方法声明
39         try{
40            String sql="update student set stuname='李勇'"; //更新数据的 sql 语句
41            sld.execSQL(sql);
42            appendMessage("成功更新记录！");                      //调用 appendMessage 方法
43         }catch(Exception e){                                    //捕获异常
44            Toast.makeText(this,"数据库错误："+e.toString(),Toast.LENGTH_SHORT).
show();;
45         }}
46        public void query(){                                     //方法声明
47         try{
48            String sql="select * from student where sage>?";
                                                                   //查询数据的 sql 语句
49            Cursor cur=sld.rawQuery(sql, new String[]{"20"});
                                                          //调用 sld.rawQuery ()方法
50            appendMessage("学号\t\t 姓名\t\t 年龄\t 班级"); //调用 appendMessage 方法
51            while(cur.moveToNext()){
52                String sno=cur.getString(0);                     //得到学号
53                String sname=cur.getString(1);                   //得到姓名
54                int sage=cur.getInt(2);                          //得到年龄
55                String sclass=cur.getString(3);
56                appendMessage(sno+"\t"+sname+"\t\t"+sage+"\t"+sclass);
                                                                   //调用 appendMessage 方法
57            }
58            cur.close();                                         //关闭游标工厂
59        }catch(Exception e){                                     //捕获异常
60            Toast.makeText(this,"数据库错误："+e.toString(),Toast.LENGTH_SHORT).
show();;
61        }}
```

其中：

- 第 1～14 行表示创建或打开数据库的方法，如果数据库已经创建，则打开数据库。
- 第 15～21 行表示关闭数据库的方法，无论在任何情况下创建或者打开数据库，在不使用后，都必须将数据库关闭。
- 第 22～29 行表示在数据库中插入记录的方法。
- 第 30～37 行表示删除数据库中所有记录的方法。
- 第 38～45 行表示更新数据库中所有记录的方法。
- 第 46～61 行表示查找数据库中所有记录的方法。

上面已经介绍了对数据库进行操作的核心代码，接下来介绍如何实现在程序间共享该数据，其代码如下。

代码位置：见本书随书光盘中源代码/第 7 章/Sample7_5/src/com/bn/ex7e 目录下的 ContentProvider。

```
1   package com.bn.ex7e;                                         //声明包
2   ……//该处省略了部分类的导入代码，读者可自行查阅随书光盘源代码
3   import android.net.Uri;                                      //导入相关类
4   public class MyContentProvider extends ContentProvider{
                                                          //创建继承 ContentProvider 方法
```

```
    private static final UriMatcher um;                          //声明静态成员变量
    static{
        um=new UriMatcher(UriMatcher.NO_MATCH);                  //创建对象
        um.addURI("com.bn.ex7e.provider.student", "stu", 1);
                                                                 //为该对象添加 URI 地址
    }
    SQLiteDatabase sld;                                          //声明成员变量
    @Override
    public String getType(Uri uri){                              //得到 URI 地址
        return null;                                             //返回空
    }
    @Override
    public Cursor query(Uri uri, String[] projection, String selection,
            String[] selectionArgs, String sortOrder){    //查询方法
        switch(um.match(uri)){                                   //判断地址是否符合
          case 1:
            Cursor cur=sld.query(                                //调用数据库的 query 方法
                    "student",                                   //数据库名称
                    projection,                                  //projection
                    selection,                                   //selection
                    selectionArgs,                               //selectionArgs
                    null,                                        //为空
                    null,
                    sortOrder                                    //sortOrder
             );
            return cur;
        }
        return null;
    }
    @Override
    public int delete(Uri arg0, String arg1, String[] arg2){return 0;}
                                                                 //删除数据返回 0
    @Override
    public Uri insert(Uri uri, ContentValues values){return null;}
                                                                 //查询返回 null
    @Override
    public boolean onCreate(){                                   //声明方法
        sld=SQLiteDatabase.openDatabase(                         //创建或打开数据库
            "/data/data/com.bn.ex7e/mydb",                       //数据库所在路径
            null,                                                //游标工厂
            SQLiteDatabase.OPEN_READWRITE|SQLiteDatabase.CREATE_IF_
NECESSARY
    );
        return false;                                            //返回 false
    }
    @Override
    public int update(Uri uri, ContentValues values, String selection,
          String[] selectionArgs)                                //声明方法
    {return 0;}                                                  //返回 0
}
```

提示：当应用程序间需要共享数据时，可以利用 ContentProvider 为数据定义一个 URL。如果为当前应用程序的私有数据定义 URL，则需要创建继承 ContentProvider 的类，然后根据数据的不同，操作调用的方法，实现这些方法的功能。

实例 6　查看手机里面的相片

由于手机的内存有限，并且相片相对于其他文本文件都比较大，所以相片一般都被存放进 SD 卡。这就涉及如何读取 SD 卡中的相片。本节主要讲解的就是如何查看 SD 卡中的图片。

【实例描述】

运行本程序，在主界面单击“图片搜索”按钮，可以查看图片列表。在图片列表中单击任意图片名称，在该列表下面可以显示与图片名称对应的图片。

本实例的运行效果如图 7-19、图 7-20 和图 7-21 所示。

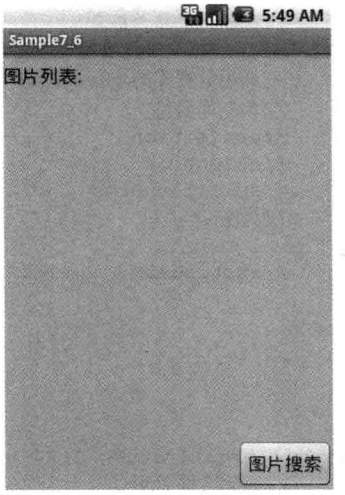


图 7-19　程序运行界面

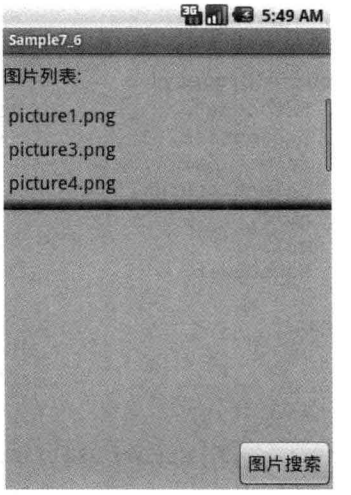


图 7-20　搜索结果

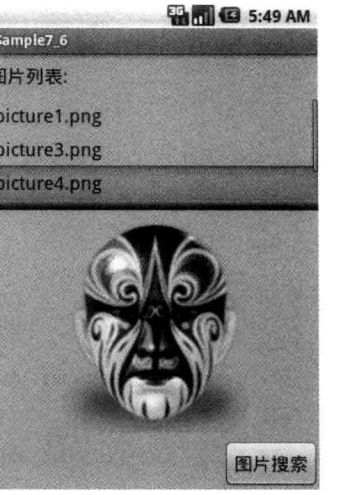


图 7-21　显示图片

提示：在本程序的运行效果图 7-19、图 7-20 和图 7-21 中，图 7-19 表示程序开始时的主界面，图 7-20 表示单击“图片搜索”按钮后的界面，图 7-21 表示显示图片的界面。

【实现过程】

在本程序中，首先通过 File 得到文件的名称列表，将其存进 ArrayList 中。然后为 ListView 设置适配器，将 ArrayList 中存储的数据添加到 ListView 列表中，并添加监听。

【代码解析】

在本部分会详细介绍本程序的实现过程。首先介绍本程序中 main.xml 文件的设置，其代码如下。

代码位置：见随书光盘中源代码/第 7 章/Sample7_6/ res/layout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>                    <!--版本号和编码方式-->
2   <FrameLayout
3        xmlns:android="http://schemas.android.com/apk/res/android"
4       android:orientation="vertical"
5       android:layout_width="fill_parent"
```

```
6       android:layout_height="fill_parent"
7       android:background="#EDAB4A"
8       android:paddingTop="10dip"
9       >                                                    <!--FrameLayout-->
10      <LinearLayout
11        android:layout_width="fill_parent"
12        android:layout_height="fill_parent"
13        android:gravity="bottom|right"
14        >                                                  <!--LinearLayout-->
15        <Button
16            android:id="@+id/btpss"
17            android:layout_width="wrap_content"
18            android:layout_height="wrap_content"
19            android:text="图片搜索"
20            android:textSize="18dip"
21            />                                             <!--Button 控件-->
22      </LinearLayout>
23      <LinearLayout
24        android:layout_width="fill_parent"
25        android:layout_height="wrap_content"
26        android:orientation="vertical"
27        >                                                  <!--LinearLayout-->
28        <LinearLayout
29            android:layout_width="fill_parent"
30            android:layout_height="wrap_content"
31            >                                              <!--LinearLayout-->
32            <TextView
33                    android:layout_width="wrap_content"
34                    android:layout_height="wrap_content"
35                    android:text="图片列表:"
36                    android:textSize="18dip"
37                    android:textColor="#000000".
38                    />                                     <!--TextView 控件-->
39            </LinearLayout>
40            <LinearLayout
41                android:layout_width="fill_parent"
42                android:layout_height="120dip"
43                android:paddingTop="10dip"
44                >                                          <!--LinearLayout-->
45                <ListView
46                    android:id="@+id/lvtplb"
47                    android:layout_width="fill_parent"
48                    android:layout_height="fill_parent"
49                    android:textSize="18dip"
50                    android:textColor="#000000"
51                    android:divider="#EDAB4A"
52                    />                                     <!-- ListView 控件-->
53            </LinearLayout>
54            <ImageView
55                android:layout_width="fill_parent"
56                android:layout_height="220dip"
57                android:paddingTop="10dip"
58                android:id="@+id/iv"
59                >                                          <!--ImageView-->
60            </ImageView>
61      </LinearLayout>                                      <!--LinearLayout-->
62  </FrameLayout>
```

其中：

- 第 2～9 行表示设置布局为帧布局，并且设置控件的排列方式为竖直排列。
- 第 10～22 行表示设置一个 LinearLayout 布局，其控件的摆放位置为右侧下方。在其中

添加一个 Button 控件，并设置其属性。

- 第 23～61 行表示再次设置一个 LinearLayout 布局，其排列方式为竖直排列，在其中添加两个 LinearLayout 布局，以及一个 ImageView 控件，同时设置其具体属性。

上面已经介绍了本程序主界面的搭建，接下来介绍读取图片名称并查看图片的功能，其代码如下。

代码位置：见本书随书光盘中源代码/第 7 章/Sample7_6/src/com/bn/ex7f 目录下的 Sample7_6_Activity。

```
1   package com.bn.ex7f;                                          //声明包
2   ……//该处省略了部分类的导入代码，读者可自行查阅随书光盘源代码
3   import android.widget.AdapterView.OnItemClickListener;  //导入相关类
4   public class Sample7_6_Activity extends Activity{    //创建继承 Activity 的类
5       Button b;                                                 //图片搜索按钮
6       ListView lv;                                              //图片名称存放 ListView
7       ImageView iv;                                             //ImageView 对象
8       ArrayList<File> al=new ArrayList<File>();         //文件存放列表
9      @Override
10     public void onCreate(Bundle savedInstanceState){ //继承类需重写的方法
11        super.onCreate(savedInstanceState);                    //调用父类
12        setContentView(R.layout.main);                         //显示主界面
13        b=(Button)this.findViewById(R.id.btpss);               //图片搜索按钮
14        lv=(ListView)this.findViewById(R.id.lvtplb);  //图片名称存放 ListView
15        iv=(ImageView)this.findViewById(R.id.iv);      //ImageView 对象
16          File[] files=new File("/sdcard").listFiles();   //得到路径
17          for(File f:files){                                     //增强 for 循环
18              String fileName=f.getName();                       //得到文件名称
19              String[] fs=fileName.split("\\.");
20              if(fs.length==2){                                  //判断 fs 的长度是否为 2
21                  String hz=fs[1];                               //得到后缀名
22                  if(hz.equals("jpg")||hz.equals("png")){
                                                                   //判断是否为 jpg 或 png 的图
23                      al.add(f);                                 //添加到 ArrayList 中
24          }}}
25        b.setOnClickListener(                                    //为按钮添加监听器
26          new OnClickListener(){                                 //匿名内部类
27                  @Override
28                  public void onClick(View v){            //实现监听器中的方法
29                      BaseAdapter ba=new BaseAdapter(){  //设置适配器
30                      @Override
31                      public int getCount(){return al.size();} //重写的方法
32                      @Override
33                      public Object getItem(int position){return null;}
                                                                   //重写的方法
34                      @Override
35                      public long getItemId(int position){return 0;}
                                                                   //重写的方法
36                      @Override
37                      public View getView(int arg0, View arg1, ViewGroup arg2)
{  //重写的方法
38                          LinearLayout ll=new LinearLayout(Sample7_6_
Activity.this);
39                          ll.setOrientation(LinearLayout.VERTICAL);
                                                                   //竖直排列
40                          ll.setPadding(5, 5, 5, 5);     //留白
41                          TextView tv=new TextView(Sample7_6_Activity.this);
```

```
                                                        //初始化 TextView
    42                          tv.setTextColor(Color.BLACK);       //设置字体颜色
    43                          tv.setText(al.get(arg0).getName());//添加文件名称
    44                        tv.setGravity(Gravity.LEFT);          //左对齐
    45                        tv.setTextSize(18);                   //字体大小
    46                        ll.addView(tv);                       //添加 TextView
    47                         return ll;
    48                    }
    49                };
    50                lv.setAdapter(ba);                  //为 ListView 添加适配器
    51                lv.setOnItemClickListener(
    52                    new OnItemClickListener(){          //匿名内部类
    53                        @Override
    54                        public void onItemClick(AdapterView<?> arg0,
View arg1,
    55                                int arg2, long arg3) {
                                                        //实现监听器中的方法
    56                            File curfile=al.get(arg2);
                                                        //获取单击图片文件
    57                            String filePath=curfile.getPath();
                                                        //得到路径
    58                            Bitmap bm=BitmapFactory.decodeFile
(filePath);
    59                            iv.setImageBitmap(bm);      //设置图片
    60   }});}});}}
```

其中：

- 第 13～16 行表示成员变量的初始化。
- 第 17～24 行表示循环遍历，查找图片名称，并将其存入 ArrayList 中。
- 第 29～49 行表示设置适配器，并设置适配器中内容的排列方式。
- 第 50 行表示为 ListView 添加适配器。
- 第 51～60 行表示为该 ListView 添加监听器，判断是否有单击事件发生。如果图片的名称被单击，则在 ImageView 控件中显示该图片。

实例 7　对数据库的简单操作

ContentResolver 类为用户应用程序提供了接入 Content 机制的方法。在获得该类的对象后就可以通过调用 query、insert 等方法来对数据库进行操作。本节主要介绍 ContentResolver 对数据库的操作。

【实例描述】

运行本程序，在主界面输入姓名，单击“查询”按钮，将查询得到的数据显示在 EditText 文本框中。本实例的运行效果如图 7-22 和图 7-23 所示。

提示：在程序的运行效果图 7-22 和图 7-23 中，图 7-22 表示程序开始时的运行效果界面，图 7-23 表示单击“查询”按钮显示查询的数据结果的界面。

【实现过程】

本程序主要是应用 xml 搭建程序的主界面，创建游标工厂，并调用 query 方法。

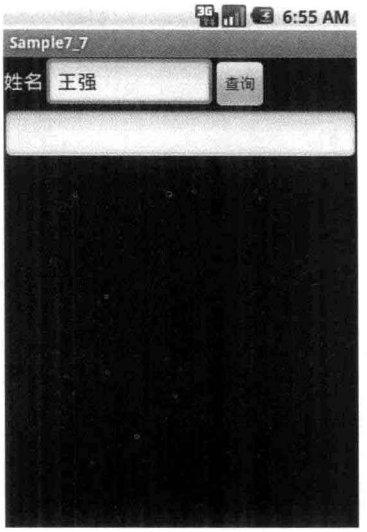


图 7-22　运行界面

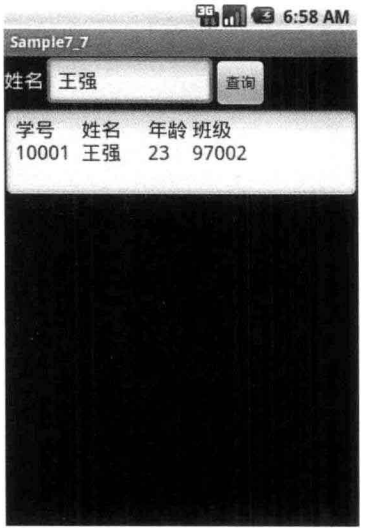


图 7-23　查询结果界面

【代码解析】

在本部分会详细介绍本程序的实现过程。首先介绍本程序中 main.xml 文件的设置，其代码如下。

代码位置：见随书光盘中源代码/第 7 章/Sample7_7/ res/layout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>                    <!--版本号和编码方式-->
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="vertical"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent"
6      >                                                      <!--LinearLayout-->
7      <LinearLayout
8      android:orientation="horizontal"
9      android:layout_width="fill_parent"
10     android:layout_height="wrap_content"
11     >                                                      <!--LinearLayout-->
12       <TextView
13         android:layout_width="wrap_content"
14         android:layout_height="wrap_content"
15         android:text="姓名"
16         android:textColor="@color/white"
17         android:textSize="18dip"
18         android:paddingRight="3dip"
19       />                                                   <!--TextView 控件-->
20       <EditText
21         android:text="王强"
22         android:id="@+id/EditText01"
23         android:layout_width="150dip"
24         android:layout_height="wrap_content">
25       </EditText>                                          <!--TextView 控件-->
26       <Button
27         android:text="查询"
28         android:id="@+id/Button01"
29         android:layout_width="wrap_content"
30         android:layout_height="wrap_content">
31       </Button>                                            <!--Button 控件-->
```

```
32     </LinearLayout>
33     <ScrollView
34       android:id="@+id/ScrollView01"
35       android:layout_width="fill_parent"
36       android:layout_height="wrap_content">          <!--ScrollView 控件-->
37      <EditText
38        android:id="@+id/EditText02"
39        android:layout_width="fill_parent"
40        android:layout_height="wrap_content">
41      </EditText>                                     <!--EditText 控件-->
42     </ScrollView>
43  </LinearLayout>
```

其中：

- 第 2～6 行表示设置 LinearLayout 布局，设置其排列方式为竖直排列。
- 第 7～32 行表示设置一个 Linearlayout 布局，设置其排列方式为竖直排列，并在该布局中添加一个 TextView 控件、EditText 控件与 Button 控件，并设置各个控件的属性。
- 第 33～42 行表示再次设置一个 ScrollView 布局，在该布局中添加一个 EditText 控件，并设置该控件的各个属性值。

上面已经介绍了本程序主界面的搭建，接下来介绍的是应用程序访问数据库，读取数据并显示的功能，代码如下。

代码位置：见本书随书光盘中源代码/第 7 章/Sample7_7/src/com/bn/ex7g 目录下的 Sample7_7_Activity。

```
1   package com.bn.ex7g;                                          //声明包
2   ……//该处省略了部分类的导入代码，读者可自行查阅随书光盘源代码
3   import android.widget.EditText;                               //导入相关类
4   public class Sample7_7_Activity extends Activity{    //创建继承 Activity 的类
5      ContentResolver cr;                                        //声明成员变量
6      @Override
7      public void onCreate(Bundle savedInstanceState){           //继承类需重写的方法
8         super.onCreate(savedInstanceState);                     //调用父类
9         setContentView(R.layout.main);                          //显示主界面
10        cr=this.getContentResolver();
11        Button b=(Button)this.findViewById(R.id.Button01);      //得到引用
12        b.setOnClickListener(                                   //设置监听
13          new OnClickListener(){                                //匿名内部类
14               @Override
15               public void onClick(View v){                     //重写的方法
16                   EditText et=(EditText)findViewById(R.id.EditText01);
                                                                   //得到引用
17                   String stuname=et.getText().toString().trim(); //得到字符串
18                   Cursor cur=cr.query(                        //调用 query 方法
19                     Uri.parse("content://com.bn.ex7e.provider.student/stu"),
20                     new String[]{"sno","stuname","sage","sclass"}, //数组
21                     "stuname=?",                              //名称
22                     new String[]{stuname},
23                     "sage ASC"                                //升序排列
24                   );
25                   appendMessage("学号\t\t 姓名\t\t 年龄\t 班级");
                                                          //调用 appendMessage 方法
26               while(cur.moveToNext()){                        //得到下一个
27                   String sno=cur.getString(0);                //得到学号
28                   String sname=cur.getString(1);              //得到姓名
29                   int sage=cur.getInt(2);                     //得到年龄
```

```
30                    String sclass=cur.getString(3);              //得到班级
31                    appendMessage(sno+"\t"+sname+"\t\t"+sage+"\t"+sclass);
                                                          //调用 appendMessage 方法
32                }
33                cur.close();                                      //关闭游标工厂
34        }});}
35      public void appendMessage(String msg){                          //定义方法
36        EditText et=(EditText)this.findViewById(R.id.EditText02);     //得到引用
37        et.append(msg+"\n");                                          //添加到 et 中
38  }}
```

其中：

- 第 12～34 行表示为“打开”按钮添加监听，在其中调用 query 方法得到数据，同时遍历得到的数据，将数据显示在 EditText 文本框中。
- 第 35～38 行表示将得到的数据添加到文本框中。

实例 8　记录访问程序的时间

有时程序需要的数据非常简单，此时动用复杂的技术就非常不值得，而在 Android 平台上提供了一种非常简单的数据存储方式——Preferences。本节将对 Preferences 的应用进行介绍。

【实例描述】

本程序主要是记录用户进入主界面的日期和时间。本实例的运行效果如图 7-24 和图 7-25 所示。

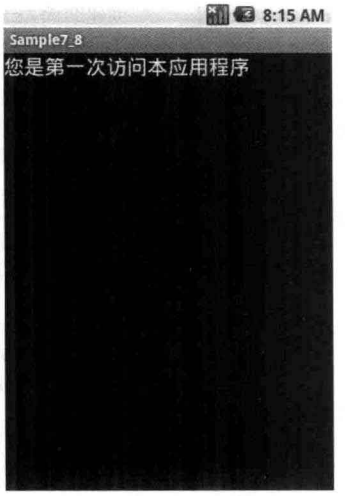


图 7-24　第一次进入程序的界面

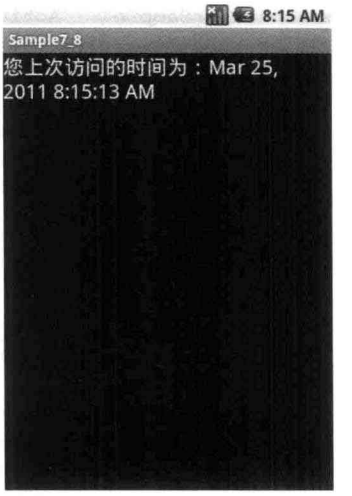


图 7-25　第二次进入程序的界面

提示：在两次程序运行效果图 7-24 和图 7-25 中，图 7-24 表示第一次进入该程序的界面，图 7-26 表示第二次进入程序的界面。

【实现过程】

Preferences 是一种轻量级的数据存储机制，将一些简单数据类型的数据，以键值对的形式

存储在应用程序的私有 Preferences 目录中。

【代码解析】

在本部分会详细介绍本程序的实现过程。首先介绍本程序中 main.xml 文件的设置，其代码如下。

代码位置：见随书光盘中源代码/第 7 章/Sample7_7/ res/layout 目录下的 main.xml。

```
<?xml version="1.0" encoding="utf-8"?>                    <!--版本号和编码方式-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   >                                                      <!--LinearLayout-->
<TextView
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:textColor="@color/white"
   android:textSize="20dip"
   android:id="@+id/TextView01"
   />                                                     <!--TextView 控件-->
</LinearLayout>
```

提示：上述 main.xml 的代码主要是搭建主界面。首先设置布局方式为 LinearLayout 布局，设置其排列方式为竖直排列，在该布局中添加 TextView 控件，并设置该控件属性。

上面已经介绍了本程序主界面的搭建，接下来介绍本程序实现 Preferences 存储的功能，代码如下。

代码位置：见本书随书光盘中源代码/第 7 章/Sample7_8/src/com/bn/ex7h 目录下的 Sample7_8_Activity。

```
package com.bn.ex7h;                                          //声明包
……//该处省略了部分类的导入代码，读者可自行查阅随书光盘源代码
import android.widget.TextView;                               //导入相关类
public class Sample7_8_Activity extends Activity{    //创建继承 Activity 的类
   @Override
   public void onCreate(Bundle savedInstanceState){       //继承类需重写的方法
      super.onCreate(savedInstanceState);                  //调用父类
      SharedPreferences sp=this.getSharedPreferences("actm", Context.MODE_
PRIVATE);
      String lastTimeStr=sp.getString(                   //读取键值为 it 的字符串数据
            "lt",                                          //键值
            null                                           //默认值
      );
      if(lastTimeStr==null){                               //lastTime 是否为空
        lastTimeStr="您是第一次访问本应用程序";                //赋值
      }else{
        lastTimeStr="您上次访问的时间为："+lastTimeStr;       //赋值
      }
      SharedPreferences.Editor editor=sp.edit();           //得到 editor 对象
      editor.putString("lt", new Date().toLocaleString());  //写入日期
      editor.commit();                                     //提交
      setContentView(R.layout.main);                       //显示主界面
      TextView tv=(TextView)this.findViewById(R.id.TextView01);  //得到引用
      tv.setText(lastTimeStr);                             //设置显示文本
}}
```

提示：本程序的主要作用是判断是否是第一次访问该程序。如果是第一次访问则在屏幕显示“你是第一次访问本程序”，否则显示用户上次访问的时间。主要是对 Preferences 的应用。

小结

本章主要介绍了手机文件 I/O 与数据库的应用，在手机文件 I/O 的学习中，通过创建 File 获取 Sdcard 目录中文件内容，并用 FileInputStream 输入流，读取文件内容存入结果集中，再按照一定的编码方式转换成用户需要的内容。

在数据库的开发中本章主要用了 Android SQllite 和 Preferences，Preferences 提供了一种存储简单数据的简便方式，是一种轻量级的数据存储机制。

第 8 章　手机网络应用

在本章节中主要向读者介绍 Android 平台下手机网络的应用。不用紧张，由于 Android 是使用 Java 的，因此网络部分的开发非常类似于 SE。

实例 1　网络连接检测软件

首先向读者介绍的是一款检测网络连接是否成功的软件，进入本软件的界面后，单击“网站连接”按钮，如果网络连接成功，将提示用户网络连接成功并显示内容。

【实例描述】

该软件主要用于检测网络的连通性，用户可以在本软件中输入所要检测的网址，然后单击“网站连接”按钮开始进行检测。当连接成功时，在软件中会显示网络连接成功的提示信息，并显示该网址的一些信息。当连接失败时，会提示用户网络连接失败。

本实例的运行效果如图 8-1 和图 8-2 所示。

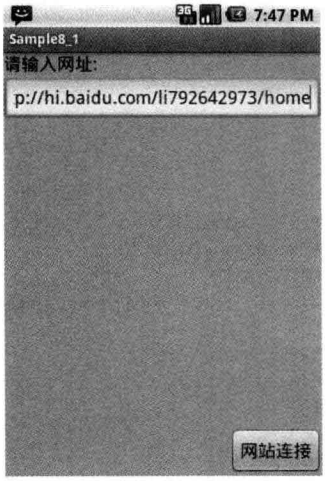


图 8-1　输入网址显示界面

图 8-2　网络连接成功界面

提示：当运行本软件后，首先显示的是如图 8-1 所示界面，在该界面中有默认的网址信息，用户可重新输入新的网址进行检测。当网络连接成功后，会显示如图 8-2 所示的界面。

【实现过程】

在本软件开发中，主要运用了 xml 界面显示技术，以及运用 HttpPost 方式建立 Http 网络连

接，并通过 HttpResponse 方式获取网络返回信息。通过 InputStreamReader 流获取返回的具体信息数据，然后显示在 TextView 控件中。

【代码解析】

本部分主要是为读者详细介绍本软件的开发过程。首先必须在 AndroidManifest.xml 中添加 Internet 网络权限，代码如下。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_1/AndroidManifest.xml

```
1   <uses-permission android:name="android.permission.INTERNET"/>        <!--
Internet 权限-->
```

接下来向读者介绍本软件 Main.xml 布局文件的开发，代码如下。

代码位置：见本书随书光盘中源代批码/第 8 章/Sample8_1/res/layout/main.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
3                        android:orientation="vertical"
4                        android:layout_width="fill_parent"
5                        android:layout_height="fill_parent"
6                        android:background="#EDAB4A"
7       >                                                    <!--总布局-->
8        <LinearLayout
9                        android:layout_width="fill_parent"
10                       android:layout_height="fill_parent"
11                       android:gravity="bottom|right"
12          >                                                <!-- LinearLayout 属性-->
13          <Button
14                       android:layout_width="wrap_content"
15                       android:layout_height="wrap_content"
16                       android:text="网站连接"
17                       android:textSize="18dip"
18                       android:textColor="#000000"
19                       android:id="@+id/b"
20          />
21      </LinearLayout>
22      <LinearLayout
23                       android:layout_width="fill_parent"
24                       android:layout_height="wrap_content"
25                       android:orientation="vertical"
26      >                                                    <!-- LinearLayout 属性-->
27          <TextView
28                       android:layout_width="wrap_content"
29                       android:layout_height="wrap_content"
30                       android:text="请输入网址:"
31                       android:textSize="18dip"
32                       android:textColor="#000000"
33                       android:id="@+id/tvts"
34          />                                               <!-- TextView 属性-->
35      <EditText
36                       android:layout_width="fill_parent"
37                       android:layout_height="wrap_content"
38                       android:text=" http://www.baidu.com/s?wd=html "
39                       android:textSize="18dip"
40                       android:textColor="#000000"
41                       android:singleLine="true"
42                       android:id="@+id/etwz"
43          />                                               <!-- EditText 属性-->
44         <LinearLayout
45                       android:layout_width="fill_parent"
46                       android:layout_height="wrap_content"
```

```
                android:paddingTop="10dip"
        >                                              <!-- LinearLayout 属性-->
        <ScrollView
                android:layout_width="fill_parent"
                android:layout_height="300dip"
            >                                          <!-- ScrollView 属性-->
            <TextView
                android:layout_width="fill_parent"
                   android:layout_height="fill_parent"
                   android:textSize="18dip"
                   android:textColor="#000000"
                   android:id="@+id/etxs"
                   android:singleLine="false"
                />                                     <!-- LinearLayout 属性-->
           </ScrollView>
       </LinearLayout>
   </LinearLayout>
</FrameLayout>
```

提示：通过上述 xml 文件编写设置，实现软件主界面的开发。其中总布局是 FrameLayout 帧布局并实现 Button 按钮控件、EditText 文本编辑框控件、TextView 文本提示控件及 ScrollView 控件的应用开发。

上述代码介绍的是本程序主界面的搭建，接下来为读者介绍本软件 Sample8_1_Activity 类的开发过程，代码如下。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_1/src/com.bn.ex8a/Sample8_1_Activity.java

```
package com.bn.ex8a;                                          //包名
import java.util.ArrayList;                                   //导入相关包
......// 该处省略了导入相关包的代码，读者可自行查阅随书光盘中源代码
import java.io.BufferedReader;                                //导入相关包
public class Sample8_1_Activity extends Activity {
    Button b;                                                 //网站连接按钮
    EditText etwz;                                            //网址输入文本
    TextView tvxs;                                            //回应字串显示文本
    TextView tvts;                                            //网址输入提示文本
    BufferedReader in = null;                                 //声明数据流缓冲
    public void onCreate(Bundle savedInstanceState){          //重写的 onCreate 方法
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        b=(Button)this.findViewById(R.id.b);                  //网站连接按钮
        tvts=(TextView)this.findViewById(R.id.tvts);          //网址输入提示文本
        etwz=(EditText)this.findViewById(R.id.etwz);          //网址输入文本
        tvxs=(TextView)this.findViewById(R.id.etxs);          //回应字串显示文本
        b.setOnClickListener (                                //为按钮添加监听器
          new OnClickListener(){
            public void onClick(View v) {
                String uri=etwz.getText().toString().trim();
                                                              //获取当前要连接的网址
                if(uri.equals("")){                  //如果网址输入为空，提示用户输入
                    Toast.makeText(Sample8_1_Activity.this, "请输入网址!",
                    Toast.LENGTH_SHORT).show();               //显示 Toast
                }
                else{
                    HttpPost postRequest=new HttpPost(uri);
                                                    //创建 Http 的 post 连接
                    ArrayList<NameValuePair> params=
```

```
29                          new ArrayList<NameValuePair>();          // Post 传送变量
30                          params.add(new BasicNameValuePair("name","Tom"));
                                                                     //传参数
31                          params.add(new BasicNameValuePair("password","123456"));
32                      try{
33                          postRequest.setEntity(new UrlEncodedFormEntity(params,
HTTP.UTF_8));
34                          HttpResponse response=new DefaultHttpClient().execute
(postRequest);
35                          if(response.getStatusLine().getStatusCode()==
HttpStatus.SC_OK){
36                      in = new BufferedReader(new InputStreamReader
37                          (response.getEntity().getContent())); //获取返回信息数据
38                          StringBuffer sb = new StringBuffer(""); //自定义缓冲
39                          String line = "";
40                          String NL = System.getProperty("line.separator");
                                                                     //为每行数据添加分隔符
41                          while((line = in.readLine()) != null){
42                          sb.append(line +NL);                     //获取各行数据信息
43                      }
44                          in.close();                              //关闭流
45                          String result = sb.toString(); //将缓冲中的数据转换成字符串
46                          tvxs.setText("网络连接成功! "+"\n"+result); //显示回应字串
47                      }
48                      else{
49                          tvxs.setText("网络连接失败! ");          //显示回应字串
50                      } }
51                          catch(Exception e){                      //捕获异常
52                          Toast.makeText(Sample8_1_Activity.this, "连接失败!
"+e.getMessage(),
53                          Toast.LENGTH_SHORT).show();              //提示信息
54  } } } } ); } }
```

其中：

- 第 6～10 行为对主界面应用控件对象的声明，及对数据流缓冲对象的声明。
- 第 11～54 行为 Button 按钮的单击事件监听器的开发。在该监听器开发中，通过 HttpPost 对象来创建 Http 的 post 连接并发出 Http 请求，通过 HttpResponse 对象来取得 Http 应答，用 InputStreamReader 对象来获取返回信息数据并存放到自定义缓冲中。

实例 2　制作简单网页浏览器

下面向读者介绍一个简单网页浏览器的开发，单击“Uri 打开网站”按钮为内置浏览器浏览页面，单击“WebView 打开网站”按钮为 WebView 控件显示页面。

【实例描述】

本节向读者介绍的是一款简单的网页浏览软件，并可以通过两种方式来进行网站的浏览，一种是 Uri 方式，另一种是 WebView 显示方式。

本实例的运行效果如图 8-3、图 8-4 和图 8-5 所示。

提示：软件开始运行，首先进入的是如图 8-3 所示的开始界面，其中的文本编辑框可供用户输入所要浏览的网站网址，当用户单击“Uri 打开网站”按钮时，界面跳转到如图 8-4 所示的界面；当用户单击“WebView 打开网站”按钮时，界面如图 8-5 所示。

图 8-3　开始界面

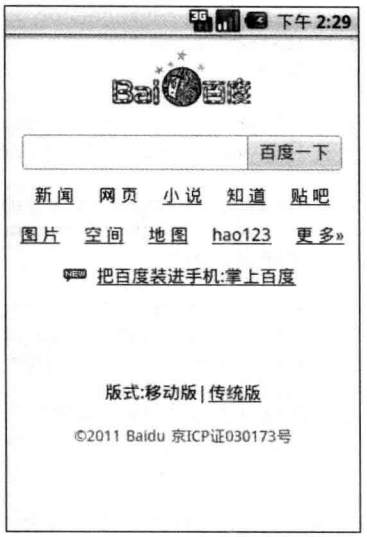


图 8-4　Uri 打开网站效果图

图 8-5　WebView 打开网站效图

【实现过程】

在本节中，主要向读者介绍 Uri 网络链接的应用，以及 WebView 控件的应用开发，通过单击“Uri 打开网站”按钮为内部浏览器显示页面，单击“WebView 打开网站”按钮为 WebView 控件显示界面。

【代码解析】

在本部分将为读者详细介绍本软件的开发过程。首先必须在 AndroidManifest.xml 中添加 Internet 网络权限，代码如下。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_2/AndroidManifest.xml

```
1    <uses-permission android:name="android.permission.INTERNET"/> <!--Internet 权限-->
```

上述代码介绍在 AndroidManifest.xml 中添加 Internet 网络权限，接下来向读者介绍本软件主界面的开发，代码如下。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_2/res/layout/main.xml

```
1    <?xml version="1.0" encoding="utf-8"?>
2    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3                  android:orientation="vertical"
4                  android:layout_width="fill_parent"
5                  android:layout_height="fill_parent"
6       android:background="#EDAB4A"    >                          <!--布局背景色-->
7        <TextView
8                  android:layout_width="wrap_content"
9                  android:layout_height="wrap_content"
10                 android:text="请输入网址:"
11                 android:textSize="18dip"
12                 android:textColor="#000000"
13       android:id="@+id/tvts"/>                                   <!--控件 id-->
14       <EditText
15                 android:layout_width="fill_parent"
16                 android:layout_height="wrap_content"
17                 android:text=http://www.baidu.com
18                 android:textSize="18dip"
```

```
                android:textColor="#000000"
                android:singleLine="true"
        android:id="@+id/et"/>                                  <!--控件 id-->
    <FrameLayout
                android:layout_width="fill_parent"
                android:layout_height="wrap_content">  <!--帧布局-->
     <LinearLayout
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
     android:gravity="bottom|right">              <!--控件在布局中的位置设置-->
         <Button
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="WebView 打开网站"
                android:textSize="18dip"
                android:textColor="#000000"
         android:id="@+id/b"/>                                  <!--控件 id-->
         </LinearLayout>
         <LinearLayout
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
         android:gravity="bottom|left">           <!--控件在布局中的位置设置-->
         <Button
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Uri 打开网站"
                android:textSize="18dip"
                android:textColor="#000000"
         android:id="@+id/b2"/>                                 <!--控件 id-->
         </LinearLayout>
    </FrameLayout>                                              <!--帧布局-->
     <LinearLayout
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:paddingTop="10dip"
     android:paddingBottom="10dip">                             <!-- 线性布局-->
     <ScrollView
                android:layout_width="fill_parent"
                android:layout_height="300dip">    <!-- ScrollView 控件-->
         <WebView
                android:layout_width="fill_parent"
                android:layout_height="fill_parent"
             android:id="@+id/wv"/>                             <!--控件 id-->
        </ScrollView>
      </LinearLayout>
</LinearLayout>
```

提示：通过上述 xml 文件的编写设置，实现了软件主界面的开发。其中总布局是线性布局并实现了 Button 按钮控件、EditText 文本编辑框控件、TextView 文本提示控件及 ScrollView 控件的应用开发。

上述 main.xml 代码介绍的是本程序主界面的开发，接下来向读者讲解该软件 Sample8_2_Activity 类的开发过程，代码如下。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_2/src/com.bn.ex8b/Sample8_2_Activity.java

```
1  package com.bn.ex8b;                                         //包名
2  import android.app.Activity;                                 //导入相关包
3  ……//该处省略了导入相关类的代码，读者可自行查阅随书光盘中源代码
```

```
4   import android.net.Uri;                                    //导入相关包
5   public class Sample8_2_Activity extends Activity {
6       Button b;                                              //WebView 打开网址按钮
7       Button b2;                                             //Uri 打开网址按钮
8       EditText et;                                           //网址编辑框
9       WebView wv;                                            //网页浏览控件
10      public void onCreate(Bundle savedInstanceState) {    //重写的 onCreate 方法
11          super.onCreate(savedInstanceState);
12          setContentView(R.layout.main);
13          b=(Button)this.findViewById(R.id.b);               //WebView 打开网址按钮
14          b2=(Button)this.findViewById(R.id.b2);             //Uri 打开网址按钮
15          et=(EditText)this.findViewById(R.id.et);           //网址编辑框
16          wv=(WebView)this.findViewById(R.id.wv);            //网页浏览控件
17          b.setOnClickListener(                          //WebView 方式按钮监听器事件
18              new OnClickListener(){                  //创建 OnClickListener 对象内部类
19              public void onClick(View v) {                  //重写 onClick()方法
20              String url = et.getText().toString();          //获取网址
21              wv.loadUrl(url);                               //在 WebView 中加载网页
22          } } );
23          b2.setOnClickListener(                             //Uri 方式按钮监听器事件
24              new  OnClickListener(){
25              public void onClick(View v) {
26              String url = et.getText().toString();          //获取网址
27              Uri uri = Uri.parse(url);                      //获取 Uri 对象
28              Intent intent = new Intent(Intent.ACTION_VIEW, uri);
                                                               //用 Intent 打开网址
29              startActivity(intent);                         //跳转界面
30  }  }   );  }   }
```

其中：

- 第 6～9 行为主界面控件对象声明。包括 Button 对象、EditText 对象和 WebView 对象。
- 第 10～30 行为重写 onCreate()方法，其中第 13～16 行为主界面控件对象的获取；第 17～22 行为 WebView 打开方式按钮监听事件；第 23～30 行为 Uri 打开方式按钮监听事件。

实例 3　自定义网页浏览器

下面向读者介绍一个自定义网页浏览器小软件的开发，进入此界面显示的是自定义网址，单击此自定义网址进入用户需要的网页界面。

【实例描述】

本节向读者介绍的是自定义开发的 HTML 网页，在该网页中给出了百度的网址链接，单击该链接可以自行跳转到百度首页。本实例的运行效果如图 8-6 和图 8-7 所示。

提示：软件开始运行，首先进入的是如图 8-6 所示的起始界面，在其中用 HTML 实现了一个网址链接，单击该链接，则界面跳转到如图 8-7 所示界面。

【实现过程】

在该小节中，主要使用了 WebView 控件和 WebView.loadData()方法以及 HTML 脚本语言知识，其中在 WebView 控件中自定义了一个网址，单击此网址进入用户需要的网页界面。

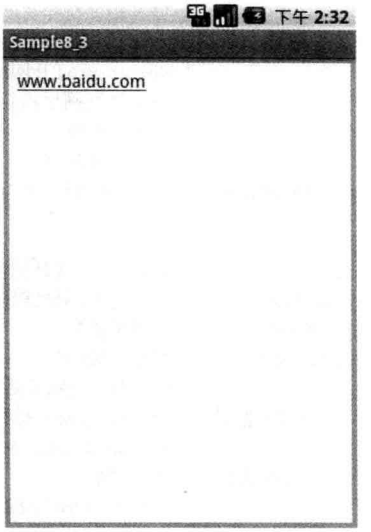


图 8-6　起始界面

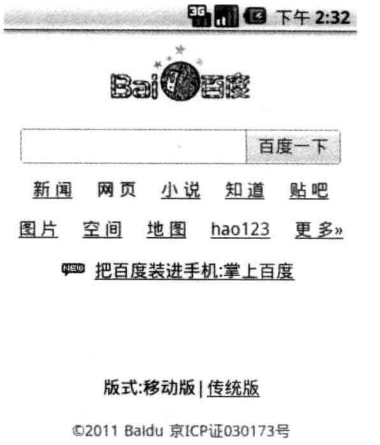


图 8-7　单击网址链接后跳转到的界面

【代码解析】

在本部分将为读者详细介绍本软件的开发，首先向读者介绍本软件主界面布局的开发，代码如下。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_3/res/layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>                    <!--版本号和编码方式-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
                        android:orientation="vertical"
                        android:layout_width="fill_parent"
                        android:layout_height="fill_parent"
                        android:background="#EDAB4A"
                        android:padding="5dip"
    >                                                     <!-- LinearLayout 属性-->
      <WebView
                        android:layout_width="fill_parent"
                        android:layout_height="fill_parent"
                        android:id="@+id/wv"
      />                                                  <!--显示网页的界面-->
</LinearLayout>
```

提示：通过上述 xml 文件的编写设置，实现了软件主界面的开发。其中总布局是线性布局，并实现了 WebView 控件的应用开发。

上述 main.xml 代码介绍的是本程序主界面的开发，接下来为读者介绍的是本程序的 Sample8_3_Activity 类的开发过程，代码如下。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_3/src/com.bn.ex8c /Sample8_3_Activity.java

```
package com.bn.ex8c;                                      //包名
import android.app.Activity;                              //导入相关包
import android.os.Bundle;                                 //导入相关包
import android.webkit.WebView;                            //导入相关包
public class Sample8_3_Activity extends Activity {
     WebView wv;                                          //网页浏览控件
```

```
7         public void onCreate(Bundle savedInstanceState) {
8             super.onCreate(savedInstanceState);
9             setContentView(R.layout.main);
10            wv=(WebView)this.findViewById(R.id.wv);     //网页浏览控件
11            wv.loadData(                                //在 WebView 中嵌套 HTML
12                "<html>"+
13                "<body test=#000000>"+
14                "<a href=\"http://www.baidu.com\">www.baidu.com</a>"+
15                "</body>"+
16                "</html>",                              //HTML TAG
17                "text/html",                            //MIME 类型
18                "utf-8"                                 //网页编码类型
19    ); } }
```

提示：在该软件 Activity 类的开发过程中，主要应用了 WebView 控件，第 10 行即为通过 findViewById()方法获取该控件对象。第 11～18 行为调用 loadData()方法，将 HTML 脚本语言嵌套到 WebView 中显示。

实例 4　网络图片浏览软件

接下来向读者介绍一款用于浏览网络图片的软件，本软件获取图片网址，然后单击“图片浏览”按钮连接网络获取图片，并把图片显示在界面上。

【实例描述】

这是一款用于浏览网络图片的小软件，在该软件中，用户可输入图片的网址信息，然后单击图片浏览，即可对该图片进行浏览。

本实例的运行效果如图 8-8 和图 8-9 所示。

图 8-8　开始界面图

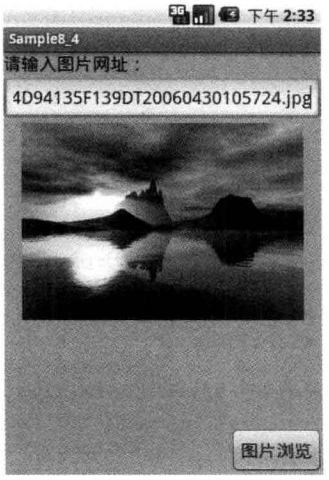


图 8-9　图片浏览界面图

提示：在本款软件运行开始后，首先进入的是如图 8-8 所示的开始界面，在该界面的文本框中可输入网络图片网址信息，然后通过单击图片浏览按钮，则会将网络图片显示在界面上，如图 8-9 所示。

【实现过程】

在本软件的开发过程中，主要应用了 HttpURLConnection 对象来取得网络连接，并应用该对象的 getInputStream()方法来获取图片流数据，最后将流数据转换成 Bitmap 对象并显示在 ImageView 中。

【代码解析】

在本部分主要是为读者详细介绍本软件的开发过程。首先必须在 AndroidManifest.xml 中添加 Internet 网络权限，代码如下。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_4/AndroidManifest.xml

```
    <uses-permission android:name="android.permission.INTERNET"/>
<!--Internet 权限-->
```

上述代码介绍的是在 AndroidManifest.xml 中添加 Internet 网络权限，接下来为读者介绍本软件布局文件的开发，代码如下。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_4/res/layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout                                    <!--总布局为 FrameLayout 帧布局-->
              xmlns:android="http://schemas.android.com/apk/res/android"
              android:orientation="vertical"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
              android:background="#EDAB4A"
>                                               <!--总布局为帧布局-->
<LinearLayout
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
              android:gravity="bottom|right"
>                                               <!--子布局为线性布局-->
<Button
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:id="@+id/b"
              android:text="图片浏览"
              android:textSize="18dip"
/>                                              <!--Button 按钮控件-->
</LinearLayout>                                 <!--线性布局子布局-->
<LinearLayout
              android:orientation="vertical"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
>                                               <!--线性布局子布局-->
<TextView
              android:layout_width="fill_parent"
              android:layout_height="wrap_content"
              android:text="请输入图片网址："
              android:textSize="18dip"
              android:textColor="#000000"
 />                                             <!--TextView 提示文本控件-->
<EditText
android:layout_width="fill_parent"
              android:layout_height="wrap_content"
              android:id="@+id/et"
              android:textSize="18dip"
              android:singleLine="true"

```

```
    android:text="http://image2.sina.com.cn/bj/t/2006-04-30/U1421P52T4D94135F139DT2
0060430105724.jpg"
41  />                                          <!--EditText 控件设置-->
42  <ImageView
43              android:layout_width="fill_parent"
44              android:layout_height="200dip"
45              android:id="@+id/iv"
46  />                                          <!--ImageView 控件设置-->
47  </LinearLayout>
48  </FrameLayout>
```

提示：通过上述 xml 文件的编写设置，实现了软件主界面的开发。其中总布局是 FrameLayout 帧布局并实现了 Button 控件、TextView 控件、EditText 控件，以及 ImageView 控件的应用开发。

上述 main.xml 代码介绍的是本程序主界面的开发，接下来为读者介绍的是本软件 Sample8_4_Activity 类的开发过程，代码如下。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_4/src/com.bn.ex8d/Sample8_4_Activity.java

```
1   package com.bn.ex8d;                                    //声明包
2   import java.io.IOException;                             //导入相关类
3   ......// 该处省略了导入相关包的代码，读者可自行查阅随书光盘中源代码
4   import android.view.View;
5   public class Sample8_4_Activity extends Activity {
6       Button b;                                           //图片浏览按钮
7       EditText et;                                        //图片网址编辑框
8       ImageView iv;                                       //图片显示控件
9       public void onCreate(Bundle savedInstanceState) {
10          super.onCreate(savedInstanceState);
11          setContentView(R.layout.main);                  //进入主界面
12          b=(Button)this.findViewById(R.id.b);            //图片浏览按钮
13          et=(EditText)this.findViewById(R.id.et);        //图片网址编辑框
14          iv=(ImageView)this.findViewById(R.id.iv);       //图片显示控件
15          b.setOnClickListener(                           //为图片浏览按钮添加监听器
16            new OnClickListener(){
17              public void onClick(View v) {               //添加监听需要重写的方法
18                  String uriPic =et.getText().toString(); //仅针对 Http 连接
19                  URL imageUrl = null;                    //声明 URL 对象
20                  Bitmap bitmap = null;                   //Bitmap 位图对象
21                  try{
22                      imageUrl = new URL(uriPic);    // new URL 对象将网址传入
23                  }catch (MalformedURLException e){ //捕获异常
24                      e.printStackTrace();            //打印堆栈信息
25                  }
26                  try{
27                      HttpURLConnection conn = (HttpURLConnection)
28                              imageUrl.openConnection();//打开连接
29                      conn.connect();                       //获取连接
30                      InputStream is = conn.getInputStream();
                                                    //取得返回的 InputStream
31                      bitmap = BitmapFactory.decodeStream(is);
                                                    //将 InputStream 变成 Bitmap
32                      is.close();                 //关闭 InputStream
33                  }catch (IOException e){         //捕获异常
34                      e.printStackTrace();        //打印信息
35                  }
```

```
36                          iv.setImageBitmap(bitmap);      //在 ImageView 中显示图片
37  }  }  );   }   }
```

其中：

- 第 6～8 行声明主界面所应用的控件对象，包括 Button 控件、EditText 控件和 ImageView 控件。
- 第 9～37 行为重写 onCreate()方法。其中，第 12～14 行为通过 findViewById()方法来获取控件对象。第 15～37 行为对 Button 按钮单击事件监听器的开发。

实例 5　网络图片相册集

接下来向读者介绍的是一款网络图片相册集的开发，首先获取图片的地址，单击“添加图片”按钮，把此图片加载到相册中，单击“浏览相册”按钮，跳转界面显示相册中的图片。

【实例描述】

本软件用于下载网络图片并存储到本地图片列表中，在文本框中输入图片网址信息，然后单击“添加图片”按钮可将网络图片下载到本地并通过 ImageView 显示。单击“浏览相册”按钮，可进行对图片列表的浏览，用户可以通过手指横向滑动手机屏幕进行图片的浏览。

本实例的运行效果如图 8-10 和图 8-11 所示。

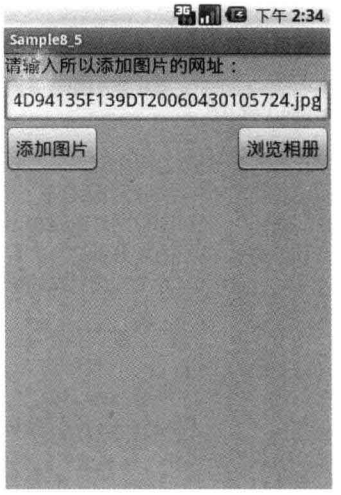


图 8-10　开始界面图

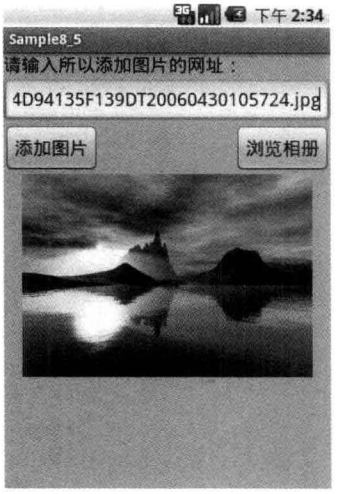


图 8-11　添加图片效果图

提示：当软件开始运行时，首先进入的是如图 8-10 所示的开始界面，在文本框中可以输入网络图片网址信息，单击“添加图片”按钮，将网络图片添加到图片列表中，效果如图 8-11 所示。

提示：当单击界面中的“浏览相册”按钮后，界面跳转到如图 8-12 和图 8-13 所示的网络图片相册界面 1 和网络图片相册界面 2，用户可以通过手指横向滑动手机屏幕来进行浏览操作。

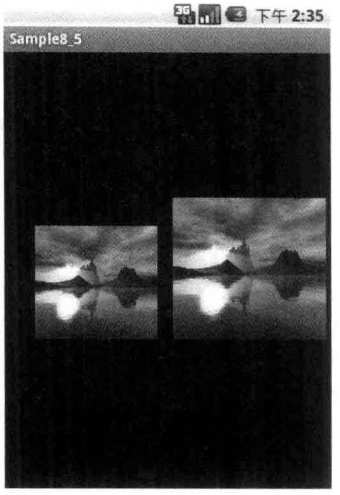


图 8-12　网络图片相册界面 1

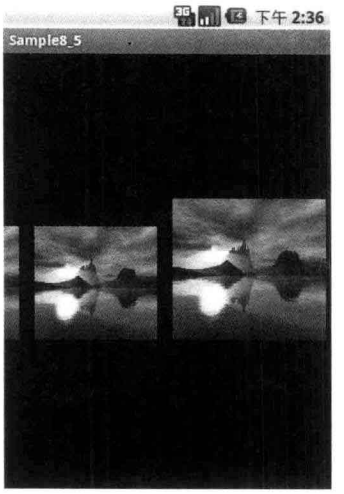


图 8-13　网络图片相册界面 2

【实现过程】

本软件中首先要求用户在 EditText 中输入图片网址，单击“添加图片”按钮把此图片加载到相册中，单击“浏览相册”按钮，跳转界面在 ImageView 中显示相册中的图片。

【代码解析】

下面向读者详细介绍本软件的开发过程。首先必须在 AndroidManifest.xml 中添加 Internet 网络权限，代码如下。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_5/AndroidManifest.xml

```
1    <uses-permission android:name="android.permission.INTERNET"/>
<!--Internet 权限-->
```

上述代码介绍的是在 AndroidManifest.xml 中添加 Internet 网络权限，接下来向读者介绍本软件主界面布局的开发，代码如下。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_5/res/layout/main.xml

```
1    <?xml version="1.0" encoding="utf-8"?>                    <!--版本号 编码方式-->
2    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3                          android:orientation="vertical"
4                          android:layout_width="fill_parent"
5                          android:layout_height="fill_parent"
6    android:background="#EDAB4A">                             <!--布局的背景色设置-->
7    <TextView
8                          android:layout_width="fill_parent"
9                          android:layout_height="wrap_content"
10                         android:text="请输入所添加图片的网址："
11                         android:textColor="#000000"
12   android:textSize="18dip" />                               <!--文本字体大小设置-->
13   <EditText                                                 <!--EditText 控件设置-->
14                         android:layout_width="fill_parent"
15                         android:layout_height="wrap_content"
16   android:text="http://image2.sina.com.cn/bj/t/2006-04-30/
U1421P52T4D94135F139DT20060430105724.jpg"
17                         android:textColor="#000000"
18                         android:textSize="18dip"
```

```
                    android:id="@+id/et"
android:singleLine="true"/>                          <!--文本内容单行显示设置-->
<FrameLayout
                    android:layout_width="fill_parent"
                    android:layout_height="wrap_content">  <!--帧布局-->
<LinearLayout
                    android:layout_width="fill_parent"
                    android:layout_height="wrap_content"
android:gravity="right">
<Button                                              <!--Button 控件设置-->
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="浏览相册"
                    android:textColor="#000000"
                    android:textSize="18dip"
android:id="@+id/bl"/>                               <!--Button 控件的 id-->
</LinearLayout>                                      <!-- 线性布局-->
<LinearLayout
                    android:layout_width="fill_parent"
                    android:layout_height="wrap_content"
android:gravity="left">                              <!--布局中控件摆放位置的设置-->
<Button
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="添加图片"
                    android:textColor="#000000"
                    android:textSize="18dip"
android:id="@+id/bt"/>                               <!--控件 id-->
</LinearLayout>                                      <!-- 线性布局-->
</FrameLayout>                                       <!--帧布局-->
<LinearLayout
                    android:layout_width="fill_parent"
                    android:layout_height="wrap_content">  <!-- 线性布局-->
<ImageView
                    android:layout_width="fill_parent"
                    android:layout_height="200dip"
                    android:id="@+id/iv">            <!--控件 id-->
                </ImageView>                         <!--ImageView 控件设置-->
        </LinearLayout>
</LinearLayout>                                      <!-- 线性布局-->
```

提示：通过上述 xml 文件的编写设置，实现了软件主界面的开发。其中总布局是线性布局并实现了 Button 控件、TextView 控件、EditText 控件，以及 ImageView 控件的应用开发。

上述 main.xml 代码介绍的是本程序主界面的开发，接下来为读者介绍的是本软件 Sample8_5_Activity 类的开发过程，代码如下。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_5/src/com.bn.ex8e/Sample8_5_Activity.java

```
package com.bn.ex8e;                                          //声明包
import java.io.InputStream;                                   //导入相关包
......// 该处省略了导入相关包的代码，读者可自行查阅随书光盘中源代码
import android.widget.ImageView;                              //导入相关包
public class Sample8_5_Activity extends Activity {
    SurfaceViewTag mySurface;                                 //界面
    ArrayList<Bitmap> al=new ArrayList<Bitmap>();   //创建 Bitmap 位图存放列表
    Bitmap bm=null ;                                          //位图
```

```
9       EditText et;                                          //文本编辑框
10      Button bt;                                            //添加图片按钮
11      Button bl;                                            //图片浏览按钮
12      ImageView iv;                                         //图片控件
13      public void onCreate(Bundle savedInstanceState) {//继承 Activity 重写的方法
14          super.onCreate(savedInstanceState);
15          setContentView(R.layout.main);                    //跳转到主界面
16          mySurface=new SurfaceViewTag(this);               //界面的对象
17          et=(EditText)this.findViewById(R.id.et);          //文本编辑框
18          bt=(Button)this.findViewById(R.id.bt);            //添加图片按钮
19          bl=(Button)this.findViewById(R.id.bl);            //图片浏览按钮
20          iv=(ImageView)this.findViewById(R.id.iv);         //图片控件
21          bt.setOnClickListener (                           //添加图片按钮监听器
22              new OnClickListener(){
23                  public void onClick(View v) {             //监听需要重写的方法
24                      String url =et.getText().toString() ;  //获得 URL 地址
25                      try{
26                        URL aryURI = new URL(url);              //创建 URL 对象
27                        URLConnection conn = aryURI.openConnection();
                                                                //打开连接
28                          conn.connect();
29                          InputStream is = conn.getInputStream();
                                                                //获取 InputStream
30                          bm = BitmapFactory.decodeStream(is);
                                                                //从输入流构建 Bitmap
31                          is.close();                         //关闭 InputStream
32                       }catch(Exception e) {
33                          StackTrace();                       //异常处理
34                      }
35                      al.add(bm);                           //将位图添加到列表
36                      iv.setImageBitmap(bm);                //在 ImageView 中显示
37          }   }   );
38          bl.setOnClickListener (                           //图片浏览按钮监听器
39              new OnClickListener(){                        //匿名内部类
40                  public void onClick(View v) {       //监听事件需要重写的方法
41                      Sample8_5_Activity.this.setContentView(mySurface);
42                      mySurface.setFocusableInTouchMode(true);
                                                  //触控设置应该在获得焦点之前
43                      mySurface.requestFocus();             //获得焦点
44  }   }   );  }   }
```

其中：

- 第 6～12 行为成员变量的声明，包括主界面中应用的控件对象以及位图存放列表对象。
- 第 13～44 行为重写 onCreate()方法。其中，第 16～20 行为通过 findViewById()方法来获取主界面应用控件对象。第 11～37 行为添加图片按钮的单击事件监听器部分。第 38～44 行为浏览相册按钮的单击事件监听器部分。

上述代码介绍的是本程序开始时的控制类，接下来向读者介绍的是图片相册浏览界面的开发过程。首先给出的是该界面类的框架结构，代码如下。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_5/src/com.bn.ex8e/SurfaceViewTag.java

```
1   package com.bn.ex8e;                                      //声明包
2   import java.util.ArrayList;                               //导入相关包
3   ......// 该处省略了导入相关包的代码，读者可自行查阅随书光盘中源代码
4   import android.view.SurfaceView;                          //导入相关包
5   public class SurfaceViewTag extends SurfaceView implements SurfaceHolder.Callback{
```

```
6       public static final int SCREEN_WIDTH=480;                    //屏幕宽
7       ......// 该处有所省了部分成员变量的声明，读者可自行查看源代码
8       public static final int MENU_SLIDESPAN=30;                   //滑动阈值
9       Sample8_5_Activity father;                                   /声明 acitvity 对象
10      Paint paint;                                                 //声明画笔
11      ArrayList<Bitmap> al;                                       //创建 Bitmap 位图存放列表
12      int currentIndex=1;                                         //当前选中的菜单数组索引
13      float changePercent=0;                                       //动画进行的百分比
14      int anmiState=0;                                             //动画的索引值
15      float currentSelectWidth;float currentSelectHeight;float currentSelectX;float
currentSelectY;
16      float leftWidth;float leftHeight;float leftX;float leftY;
                                                                     //当前选中菜单，左侧菜单的宽与高
17      float rightWidth; float rightHeight;float rightX;float rightY;
                                                                     //当前选中菜单，右侧菜单的宽与高
18      float previousX; float previousY;                           //上次触控的 X,Y 坐标
19      public SurfaceViewTag(Sample8_5_Activity father) {
20          super(father);this.father=father;                        //获取当前 Activity 对象
21          this.getHolder().addCallback(this);                      //获得 Callback 接口
22          Paint=new Paint();                                       //创建画笔
23          paint.setAntiAlias(true);                                //抗锯齿设置
24          al=father.al;                                            //获取位图列表
25          initMenu();                                              //调用 initMenu()方法
26      }
27      /*该处省略了部分方法，将在下面给出*/
28      public void onDraw(Canvas canvas){... ...}                   //重写的 onDraw()方法
29      public void surfaceChanged(SurfaceHolder holder, int format, int width,int
height) {}
30      public void surfaceCreated(SurfaceHolder holder) {  //重写的 onCreate 方法
31          repaint();                                               //调用 repaint()方法，重绘界面
32      }
33      public void surfaceDestroyed(SurfaceHolder holder) {} //重写的 onDestory 方法
34      public void repaint(){                                       //重绘界面的方法
35          SurfaceHolder holder=this.getHolder(); //获取 SurfaceHolder 对象
36          Canvas canvas=holder.lockCanvas();                       //锁画布
37          try{
38              synchronized(holder){                                //同步获取画布
39                  onDraw(canvas);                                  //绘制界面
40          } }catch(Exception e){
41              e.printStackTrace();                                 //异常处理
42          }finally{
43              if(canvas!=null) {
44                  holder.unlockCanvasAndPost(canvas);              //解开画布
45      }   }    }
46      public void initMenu(){                                      //初始化菜单方法
47        urrentSelectWidth=BIGWIDTH; currentSelectHeight=BIGHEIGHT;
                                                                     //初始化当前菜单大小的位置
48        currentSelectX=SELECT_X; currentSelectY=SELECT_Y;
49        leftWidth=SMALLWIDTH; leftHeight=SMALLHEIGHT;              //左侧菜单的大小
50        leftX=currentSelectX-MENU_SPAN-leftWidth;                  //左侧菜单的横坐标
51        leftY=currentSelectY+currentSelectHeight-leftHeight;//左侧菜单的纵坐标
52        rightWidth=SMALLWIDTH; rightHeight=SMALLHEIGHT;
                                                                     //紧邻当前菜单，右侧菜单的大小
53        rightX=currentSelectX+currentSelectWidth+MENU_SPAN; //右侧菜单的横坐标
54        rightY=currentSelectY+currentSelectHeight-rightHeight; //右侧菜单的纵坐标
55  } }
```

其中：

- 第 6～18 行为声明所要用到的常量，以及该类的成员变量。
- 第 19～26 行为该类的构造器方法。在其中创建了画笔对象并初始化了界面所要应用的位图资源的初始位置。
- 第 34～45 行为 repaint()重绘方法，用于同步绘制更新界面。

接下来为读者介绍的是相册浏览界面类中屏幕触控事件的实现方法，以及界面绘制方法的开发过程。将代码插入到上述代码的第 27 行。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_5/src/com.bn.ex8e /SurfaceViewTag.java

```
1   public boolean onTouchEvent(MotionEvent e){          //屏幕触控事件
2       if(anmiState!=0){                                //若动画播放中则触控无效
3           return true;                                 //返回值为 true
4       }
5       float x=e.getX();float y=e.getY();               //获得当前触控点的 X,Y 坐标
6       switch(e.getAction()){                           //判断动作
7           case MotionEvent.ACTION_DOWN:                //动作为按下
8               previousX=x;previousY=y;                 //记录 X，Y 坐标
9           break;
10          case MotionEvent.ACTION_UP:                  //动作为抬起
11              float dx=x-previousX;                    //计算 X 位移
12              if(dx<-MENU_SLIDESPAN){                 //若 X 位移小于负阈值，则向左移
13                  if(currentIndex<al.size()-1){        // 若当前菜单项不是最后一个
14                  int afterCurrentIndex=currentIndex+1;  //新的当前菜单项索引
15                      anmiState=2;                     //动画状态设为 2，表向左移
16                      new ThreadMenu(this,afterCurrentIndex).start();
17                  } }
18              else if(dx>MENU_SLIDESPAN){              //判断 dx 的值
19                  if(currentIndex>0){                  //若当前菜单项不是第一个
20                      int afterCurrentIndex=currentIndex-1; //新的当前菜单项索引
21                      anmiState=1;                     //动画状态设为 1，表向右移
22                      new ThreadMenu(this,afterCurrentIndex).start();
23          } }
24          break;                                       //退出
25      }
26   return true;                                        //返回值为 true
27  }
28  public void onDraw(Canvas canvas){                   //绘制的方法
29      super.onDraw(canvas);                            //调用父类
30      canvas.drawColor(Color.argb(255, 0, 0, 0));      //擦空界面
31      Bitmap selectbm=al.get(currentIndex);            //获得当前选中的位图
32      selectbm=Bitmap.createScaledBitmap(              //创建新位图
33          selectbm,(int)currentSelectWidth,            //这里一定要是 int 的
34          (int)currentSelectHeight,false);
35      canvas.drawBitmap(selectbm, currentSelectX, currentSelectY, paint);
36      if(currentIndex>0){                              //currentIndex 是否大于 0
37          Bitmap leftbm=Bitmap.createScaledBitmap(     //创建新位图
38          al.get(currentIndex-1),(int)leftWidth,(int)leftHeight,false);
39          canvas.drawBitmap(leftbm, leftX, leftY, paint);//绘制当前位图左侧的位图
40      }
41      if(currentIndex<al.size()-1){
42          Bitmap rightbm=Bitmap.createScaledBitmap(        //创建新位图
43          al.get(currentIndex+1),(int)rightWidth,(int)rightHeight,false);
44          canvas.drawBitmap(rightbm, rightX, rightY, paint);//绘制当前位图右侧的位图
45      }
46      for(int i=currentIndex-2;i>=0;i--){     //对当前位图左侧的位图对象进行绘制
```

```
47            float tempx=leftX-(MENU_SPAN+SMALLWIDTH)*((currentIndex-1)-i);
48            if(tempx<-SMALLWIDTH){                          //判断 tempx 值的大小
49                break;                                      //退出
50            }
51            float tempy=SELECT_Y+(BIGHEIGHT-SMALLHEIGHT);
52            Bitmap tempbm=Bitmap.createScaledBitmap(    //创建新位图
53                al.get(i),(int)SMALLWIDTH,(int)SMALLHEIGHT,false);
54            canvas.drawBitmap(tempbm, tempx, tempy, paint); //绘制位图
55        }
56        for(int i=currentIndex+2;i<al.size();i++){ //对当前位图右侧的位图进行绘制
57            float tempx=rightX+(SMALLWIDTH+MENU_SPAN)*(i-(currentIndex+1));
58            if(tempx>SCREEN_WIDTH){                         //判断 tempx 值的大小
59                break;                                      //退出
60            }
61            float tempy=SELECT_Y+(BIGHEIGHT-SMALLHEIGHT);
62            Bitmap tempbm=Bitmap.createScaledBitmap(            //创建新位图
63                al.get(i),(int)SMALLWIDTH,(int)SMALLHEIGHT,false);
64            canvas.drawBitmap(tempbm, tempx, tempy, paint);     //绘制位图
65    } }
```

其中：

- 第 1～27 行为屏幕触控事件 onTouchEvent()方法的开发，用于进行图片浏览控制，并绘制当前界面。
- 第 28～65 行为界面绘制方法 onDraw()方法的开发，该方法中主要进行对界面各个图片的绘制。

上述代码介绍的是本程序触控事件的实现，接下来介绍的是本软件中实时监测界面，此界面为相册界面，并通过调用 onDraw()方法进行界面绘制的线程类的开发过程，代码如下。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_5/src/com.bn.ex8e/ThreadMenu.java

```
1   package com.bn.ex8e;                                    //声明包
2   public class ThreadMenu extends Thread {                //继承自 Thread 类
3       public static final int SCREEN_WIDTH=480;           //屏幕宽
4       public static final int SCREEN_HEIGHT=320;          //屏幕高
5       public static int BIGWIDTH=150;                     //选中菜单项的宽
6       public static int BIGHEIGHT=140;                    //选中菜单项的高
7       public static int SELECT_X=SCREEN_WIDTH/2-BIGWIDTH/2;
                                                            //选中菜单项左上角的坐标
8       public static float SELECT_Y=SCREEN_HEIGHT/1.5f-BIGHEIGHT/2;
9       public static int SMALLWIDTH=120;                   //未选中菜单项的宽与高
10      public  static  int  SMALLHEIGHT=(int)(((float)SMALLWIDTH/BIGWIDTH)*
BIGHEIGHT);//关注细节
11      public static final int MENU_SPAN=15;                   //菜单间的间距
12      public static final int MENU_SLIDESPAN=30;              //滑动阈值
13      public static final int TOTAL_STEPS=10;                 //动画总步数
14      public static final float PERCENT_STEP=1.0f/TOTAL_STEPS;
                                                                //每一步动画的百分比
15      public static final int ANMI_TIMESPAN=20;               //每一步动画的间隔时间
16      SurfaceViewTag mv;
17      int afterCurrentIndex;                          //播放完后，菜单图片的数组索引
18      public ThreadMenu(SurfaceViewTag mv,int afterCurrentIndex)   {
19          this.mv=mv;this.afterCurrentIndex=afterCurrentIndex;     //初始化 mv
20      }
21      public void run(){
22          for(int i=0;i<=TOTAL_STEPS;i++){                //循环指定的步数完成动画
23              mv.changePercent=PERCENT_STEP*i;            //计算此步占动画的百分比
24              mv.initMenu();                              //初始化各个位置的值
```

```
25                if(mv.anmiState==1){                          //如果状态是 1，则向右移
26                  mv.currentSelectX=mv.currentSelectX+
27                  (BIGWIDTH+MENU_SPAN)*mv.changePercent;     //当前的 X 的值
28                  mv.currentSelectY=mv.currentSelectY+
29                  (BIGHEIGHT-SMALLHEIGHT)*mv.changePercent; //当前的 Y 的值
30                  mv.currentSelectWidth=BIGWIDTH-
31                  (BIGWIDTH-SMALLWIDTH)*mv.changePercent;    //当前图片的宽度
32                  mv.currentSelectHeight=BIGHEIGHT-
33                  (BIGHEIGHT-SMALLHEIGHT)*mv.changePercent; //当前图片的高度
34                  mv.leftWidth=SMALLWIDTH+
35                  (BIGWIDTH-SMALLWIDTH)*mv.changePercent;    //左侧图片的宽度
36                  mv.leftHeight=SMALLHEIGHT+
37                  (BIGHEIGHT-SMALLHEIGHT)*mv.changePercent; //左侧图片的高度
38                }
39                else if(mv.anmiState==2){                     //如果状态是 2，则向左移
40                  mv.currentSelectX=mv.currentSelectX-
41                  SMALLWIDTH+MENU_SPAN)*mv.changePercent;    //当前的 X 的值
42                  mv.currentSelectY=mv.currentSelectY+
43                  (BIGHEIGHT-SMALLHEIGHT)*mv.changePercent; //当前的 Y 的值
44                  mv.currentSelectWidth=BIGWIDTH-
45                  (BIGWIDTH-SMALLWIDTH)*mv.changePercent;    //当前图片的宽度
46                  mv.currentSelectHeight=BIGHEIGHT-
47                  (BIGHEIGHT-SMALLHEIGHT)*mv.changePercent; //左侧图片的高度
48                  mv.rightWidth=SMALLWIDTH+
49                  (BIGWIDTH-SMALLWIDTH)*mv.changePercent;    //右侧图片的宽度
50                  mv.rightHeight=SMALLHEIGHT+
51                  (BIGHEIGHT-SMALLHEIGHT)*mv.changePercent; //右侧图片的高度
52                }
53                  mv.leftX=mv.currentSelectX-MENU_SPAN-mv.leftWidth;
                                                              //左侧菜单的位置
54                  mv.leftY=mv.currentSelectY+
55                          (mv.currentSelectHeight-mv.leftHeight);
                                                              //左侧菜单的位置
56                  mv.rightX=mv.currentSelectX+
57                          mv.currentSelectWidth+MENU_SPAN;  //右侧菜单的位置
58                  mv.rightY=mv.currentSelectY+
59                          (mv.currentSelectHeight-mv.rightHeight);
                                                              //右侧菜单的位置
60                    mv.repaint();
61                try{
62                    Thread.sleep(ANMI_TIMESPAN);            //线程休眠
63                }catch(Exception e){
64                    e.printStackTrace();                    //打印信息
65            } }
66            mv.anmiState=0;                                 //anmiState 值为 0
67            mv.currentIndex=afterCurrentIndex;
68            mv.initMenu();                                  //调用 initMenu 方法
69            mv.repaint();                                   //调用 repaint 方法
70    } }
```

其中：

- 第 3～17 行为应用常量的声明。第 18～20 行为该类的构造器方法，在其中传入了 SurfaceViewTag 对象和 afterCurrentIndex 变量。
- 第 21～70 行为重写的 run()方法。第 22～65 行为循环指定的步数完成动画。当状态为 1 时，向右移，当状态为 2 时，向左移。

实例 6　手机查看实时卫星云图

当今越来越多的人在悠闲时喜欢外出旅行，但天气情况是外出旅行的人们最担心的。在本节将要介绍的是在手机端查看实时卫星云图，可以方便地了解天气情况。

【实例描述】

运行本程序进入主界面，单击“下载实时卫星云图”按钮打开卫星云图，可以下载当前的卫星云图，方便观察近几天的天气。本实例的运行效果如图 8-14 和图 8-15 所示。

图 8-14　程序开始主界面

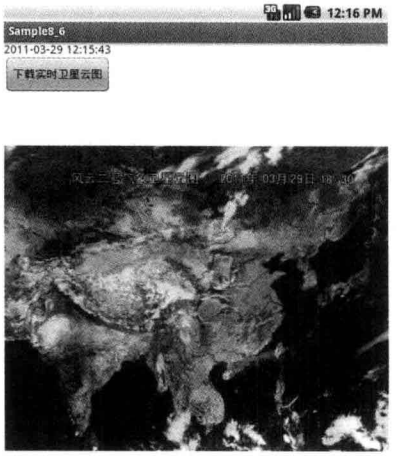


图 8-15　卫星云图界面

提示：在本程序的运行效果图 8-14 和图 8-15 中，图 8-14 表示的是程序开始时的运行效果图，图 8-15 为单击“下载实时卫星云图”按钮后的卫星云图界面。

【实现过程】

在本程序的开发过程中，主要用到在 xml 文件中对主界面搭建，文件的 I/O 操作，通过 URL 联机并获得链接，开启新线程时更新数据，同时需要在 AndroidManifest.xml 中声明网络访问权限。

【代码解析】

在本部分会详细介绍该软件的实现过程。首先介绍的是该软件中 main.xml 文件的设置，代码如下。

代码位置：见随书光盘中源代码/第 8 章/Sample8_6/ res/layout 目录下的 main.xml。

```
1    <?xml version="1.0" encoding="utf-8" ?>          <!--版本号和编码方式-->
```

```
<LinearLayout
              xmlns:android="http://schemas.android.com/apk/res/android"
              android:background="@drawable/white"
              android:orientation="vertical"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
    >                                          <!-- LinearLayout 的属性-->
  <TextView
                  android:id="@+id/TextView01"
                  android:layout_width="fill_parent"
                  android:text="日期、时间显示"
                  android:textColor="@drawable/blue"
                  android:layout_height="wrap_content"
    />                                         <!-- TextView 的属性-->
    <Button
                  android:id="@+id/Button01"
                  android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:text="下载实时卫星地图"
    />                                         <!-- Button 的属性-->
    <ImageView
                  android:id="@+id/ImageView01"
                  android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:layout_gravity="center"
    />                                         <!-- ImageView 的属性-->
</LinearLayout>
```

提示：上述 main.xml 的代码主要是搭建程序开始运行的主界面。首先在 LinearLayout 中设置摆放方式为竖直摆放，然后添加 TextView、Button、ImageView 控件，并设置其具体的属性。

上述代码已经介绍完本程序主界面的搭建。接下来将要介绍如何在手机端打开卫星云图，代码如下。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_6/src/com/bn/ex8f 目录下的 Sample8_6_Activity。

```
package com.bn.ex8f;                                          //声明包
……//该处省略了导入相关包的代码，读者可自行查阅随书光盘中源代码
import android.widget.TextView;                               //导入相关包
public class Sample8_6_Activity extends Activity{             //创建继承 Activity 类
      ......// 该处省略了部分私有成员变量代码，读者可自行查阅随书光盘中源代码
       @Override
       public void onCreate(Bundle savedInstanceState){
                                                   //继承 Activity 类要重写的方法
            super.onCreate(savedInstanceState);               //调用父类
            setContentView(R.layout.main);                    //设置布局
            sdf = new java.text.SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
            mButton = (Button) findViewById(R.id.Button01); //连接实时卫星云图
            mTextView = (TextView) findViewById(R.id.TextView01); //日期、时间显示
            mImageView = (ImageView) findViewById(R.id.ImageView01);//图片显示
            mButton.setOnClickListener(                   //为 Button 按钮添加监听器
               new Button.OnClickListener(){              //匿名内部类
                   @Override
                   public void onClick(View arg0){        //需要重写的方法
                       getWeatherPic();                   //调用 getWeatherPic 方法
```

```
19              }});
20              new Thread(mTks).start();                    //开启线程
21          }
22          @Override
23          protected void onDestroy(){                      //重写的方法
24                flag = false;                              //flag 为 false
25                super.onDestroy();                         //调用父类
26          }
27          private void getWeatherPic(){                    //声明 getWeatherPic 方法
28                try{
29                     String uriAPI="http://image.weather.gov.cn/product"+
                                                             //得到网址
30                                    +"/2011/201103/20110329/WXCL/medium/"+
31                     +"SEVP_NSMC_WXCL_ASC_E99_ACHN_LNO_PY_20110329103000000.JPG";
32                     URL objURL = new URL(uriAPI);         //得到 URL
33                     URLConnection conn = objURL.openConnection();  //建立联机
34                     conn.connect();                       //创建链接
35                     BufferedReader in = new BufferedReader(
36                                  new InputStreamReader(conn.getInputStream()));
37                     String inputLine;                     //声明字符串引用
38                     String uriPic = "";                   //uriPic 长度为零
39                     while ((inputLine = in.readLine()) != null){ //while 方法
40                      uriPic += inputLine;                 //改变 uriPic 字符串
41                     }
42                     objURL = new URL(uriPic);             //创建新的 URL
43                     HttpURLConnection  conn2  =  (HttpURLConnection)  objURL.
openConnection();
44                     conn2.connect();                      //建立连接
45                     InputStream is = conn2.getInputStream();   //得到输入流
46                     Bitmap bm = BitmapFactory.decodeStream(is); //得到 Bitmap 对象
47                     is.close();                           //关闭输入流
48                     mImageView.setImageBitmap(bm);        //设置图片
49                     mTextView.setText(sdf.format(new java.util.Date()));
                                                             //设置 TextView 的显示值
50                }catch (MalformedURLException e){                 //捕获异常
51                     e.printStackTrace();                  //打印堆栈信息
52                } catch (IOException e){                   //捕获 IOException 异常
53                     e.printStackTrace();                  //打印堆栈信息
54                }
55          }
56          private Runnable mTks = new Runnable(){          //内部类
57            public void run(){                             //重写的 run()方法
58              while (flag){                                //判断 flag 是否为 true
59                try{
60                     Thread.sleep(IntervalSec * 1000);     //休息
61                     hd.sendMessage(hd.obtainMessage());   //发送消息
62                }catch (InterruptedException e){           //捕获异常
63                   e.printStackTrace();                    //打印堆栈信息
64          }}}};
65          Handler hd = new Handler(){                      //内部类
66            public void handleMessage(Message msg){        //接收消息的方法
67              super.handleMessage(msg);                    //调用父类
68              getWeatherPic();                           //调用 getWeatherPic()方法
69    }};}
```

其中：

- 第 14-19 行表示为“下载实时卫星云图”按钮添加监听器，调用 getWeatherPic 方法。

- 第 23-26 行表示程序结束时进入销毁状态需要重写的方法。
- 第 27-55 行表示得到卫星云图的具体方法，主要是对 URL 与文件 I/O 的操作。
- 第 56-62 行表示实现 Runnable 接口的内部类方法。
- 第 65-69 行表示接收消息重写的 Handler 方法。

实例 7　Google 天气客户端

下面向读者介绍一款天气预报软件的开发过程，进入此软件首先显示的是城市的列表，单击城市名称可以查询此城市的天气情况。

【实例描述】

在本款软件中，可以查看不同城市的天气情况，包括当前温度、当前湿度、当前风向、当前天气以及当前风速情况。本软件提供了北京、天津、广州、上海、重庆和唐山的天气情况显示。

本实例的运行效果如图 8-16 和图 8-17 所示。

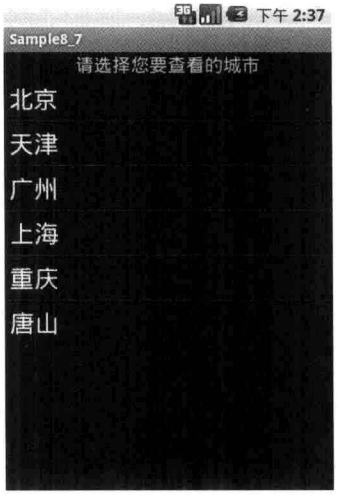


图 8-16　城市列表界面

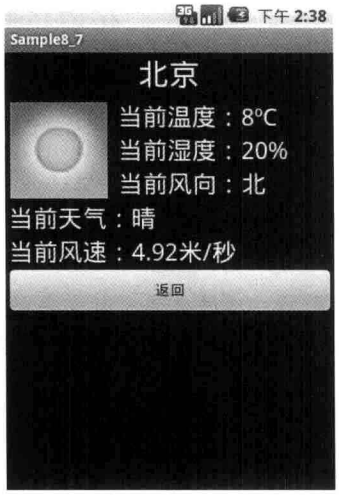


图 8-17　北京天气情况

提示：运行软件后首先进入的是城市列表界面，如图 8-16 所示。当单击城市列表中的城市名称后，界面会跳转到天气的具体信息显示界面，如图 8-17 所示为北京市的天气情况介绍。

【实现过程】

很多大的互联网内容提供商都提供天气预报的服务，如 Google、Yahoo 等。使用时一般是由客户端发送指定格式的 XML 文档。客户端收到 XML 文档后再在本地解析呈现。例如，Google 获取北京天气预报的 URL 为：http://www.google.com/ig/api?hl=en&weather=beijing,china。

【代码解析】

在本部分将要为读者详细介绍本软件的开发过程。首先必须在 AndroidManifest.xml 中添加

Internet 网络权限，代码如下。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_7/AndroidManifest.xml

```
1   <uses-permission android:name="android.permission.INTERNET"/>
<!--Internet 权限-->
```

上述代码介绍的是在 AndroidManifest.xml 中添加 Internet 网络权限，接下来向读者介绍本软件主界面布局的开发，代码如下。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_7/res/layout/main.xml

```
1   <?xml version="1.0" encoding="utf-8"?>          <!--版本号和编码方式-->
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3                 android:orientation="vertical"
4                 android:layout_width="fill_parent"
5                 android:layout_height="fill_parent"
6       >                                                <!-- LinearLayout 控件设置-->
7       <TextView
8                     android:layout_width="fill_parent"
9                     android:layout_height="wrap_content"
10                    android:text="请选择您要查看的城市"
11                    android:textColor="@color/green"
12                    android:textSize="18dip"
13                    android:gravity="center_horizontal"
14      />                                               <!--TextView 控件设置-->
15       <ListView
16                android:id="@+id/ListView01"
17                android:layout_width="fill_parent"
18                android:layout_height="wrap_content">  <!--ListView 的大小-->
19       </ListView>                                     <!--ListView 控件设置-->
20  </LinearLayout>
```

提示：通过上述 xml 文件的编写设置，实现了软件主界面的开发。其中总布局是线性布局，并实现了 TextView 控件和 ListView 控件的应用开发。

上述代码介绍的是本程序主界面的开发，接下来介绍的是天气详细信息界面布局的开发，代码如下。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_7/res/layout/detail.xml

```
1   <?xml version="1.0" encoding="utf-8"?>              <!--版本号和编码方式-->
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3                     android:orientation="vertical"
4                     android:layout_width="fill_parent"
5                     android:layout_height="fill_parent">
6       <TextView
7                     android:id="@+id/TextView06" android:gravity="center"
8                     android:layout_width="fill_parent"
9                     android:layout_height="wrap_content"
10        android:textColor="@color/white" android:textSize="30dip"/>
                                                          <!--控件内容字体设置-->
11      <LinearLayout
12                    android:id="@+id/LinearLayout01"
13                    android:orientation="horizontal"android:gravity="center"
14                    android:paddingTop="5dip"
15                    android:layout_width="fill_parent"
16                    android:layout_height="wrap_content">
                                                      <!-- LinearLayout 的属性-->
17          <ImageView
18                    android:id="@+id/ImageView01"
19                    android:layout_width="100dip"  android:paddingLeft="5dip"
```

```
        android:layout_height="100dip">                    <!-- ImageView 的大小-->
      </ImageView>                                         <!-- ImageView 的属性-->
    <LinearLayout
            android:id="@+id/LinearLayout02"
            android:orientation="vertical"
            android:paddingLeft="10dip"  android:gravity="left"
            android:layout_width="fill_parent"
      android:layout_height="wrap_content"> <!-- LinearLayout 的属性-->
        <TextView
                android:id="@+id/TextView02"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:textColor="@color/white"
            android:textSize="24dip" android:gravity="left"/>
                                                    <!--TextView 控件设置-->
         <TextView
                android:id="@+id/TextView03"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:textColor="@color/white"
            android:textSize="24dip"android:gravity="left"/>
                                          <!--控件文本字体大小及位置设置-->
         <TextView
                android:id="@+id/TextView04"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:textColor="@color/white"
            android:textSize="24dip" android:gravity="left"/>
                                              <!--TextView 控件设置-->
      </LinearLayout>
  </LinearLayout>                                          <!-- LinearLayout 的属性-->
  <TextView
    android:id="@+id/TextView01"  android:gravity="left"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:textColor="@color/white"
            android:paddingLeft="3dip"+
     android:textSize="24dip"/>                            <!--字体大小设置-->
  <TextView
            android:id="@+id/TextView05" android:gravity="left"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:textColor="@color/white"
    android:paddingLeft="3dip"android:textSize="24dip"/> <!--控件留白设置-->
  <Button
            android:layout_width="fill_parent"
            android:layout_height="wrap_content" android:text="返回"
      android:id="@+id/ButtonBack"/>                  <!--按钮控件 id-->
</LinearLayout>
```

提示：通过上述 xml 文件的编写设置，实现了软件子界面的开发。其中总布局是线性布局，并实现了 Button 控件、TextView 控件，以及 ImageView 控件的应用开发。

上述代码介绍的是天气详细信息界面布局的开发，接下来为读者介绍的是本软件 Sample8_7_Activity 类的开发过程，代码如下所示。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_7/src/com.bn.ex8g /Sample8_7_Activity.java

```
1   package com.bn.ex8g;                                    //声明包
2   import java.io.ByteArrayOutputStream;                   //导入相关包
```

```
3   ......// 该处省略了导入相关包的代码，读者可自行查阅随书光盘中源代码
4   import android.widget.Toast;                                    //导入相关包
5   public class Sample8_7_Activity extends Activity {
6       String[] cityStrArray;                                      //城市名称数组
7       int cityCount;                                              //城市数量
8       public void onCreate(Bundle savedInstanceState) {
9           super.onCreate(savedInstanceState);
10          setContentView(R.layout.main);                          //主界面显示
11          initCityList();                                         //初始化城市列表信息
12      }
13      /*该处省略了初始化列表的方法，将在后面给出*/
14      public String getCondition(String enStr) {      //翻译天气情况字符串的方法
15          if(enStr.trim().equals("Cloudy")){return "多云";}          //多云
16          else if(enStr.trim().equals("Clear")){return "晴";}        //晴
17          else if(enStr.trim().equals("Overcast")){return "阴";}     //阴
18          else if(enStr.trim().equals("Rain Showers")){return "阵雨";} //阵雨
19          else if(enStr.trim().equals("Rain")){return "雨";}         //雨
20          else if(enStr.trim().equals("Heavy Rain")){return "大雨";}    //大雨
21          else if(enStr.trim().equals("Partly Sunny")){return "晴间多云";}
                                                                       //晴间多云
22          else if(enStr.trim().equals("Chance of Rain")){return "可能有雨";}
23          else if(enStr.trim().equals("Light rain")){return "小雨";}    //小雨
24          else if(enStr.trim().equals("Mostly Sunny")){return "以晴为主";}
                                                                       //以晴为主
25          else if(enStr.trim().equals("Mostly Cloudy")){return "以多云为主";}
26          else if(enStr.trim().equals("Partly Cloudy")){return "局部多云";}
                                                                       //局部多云
27          return enStr;
28      }
29      public String getDirection(String direction) {  //翻译当前风向字符串的方法
30          String result="";
31          for(int i=0;i<direction.trim().length();i++){   //循环获取风向表示字符
32              char c=direction.charAt(i);                 //获得当前字符
33              switch(c){
34                  case 'E':                               //c 的值为 E
35                      result=result+"东";                 //为 result 赋值
36                  break;
37                  case 'W':                               //c 的值为 W
38                      result=result+"西";                 //为 result 赋值
39                  break;                                  //退出
40                  case 'N':                               //c 的值为 N
41                      result=result+"北";                 //为 result 赋值
42                  break;
43                  case 'S':                               //c 的值为 S
44                      result=result+"南";                 //为 result 赋值
45                  break;                                  //退出
46              } }
47          if(result.length()==2){                         //判断 result 的长度
48              result=result.substring(1)+result.substring(0,1); //为 result 赋值
49          }
50          return result;
51      }
52      public String getSpeed(String speed){               //翻译风速字符串的方法
53          double d=Double.parseDouble(speed.trim());      //获取当前风速字符串
54          d=d*1609.344/3600;String result=(Math.round(d*100)/100.00)+"米/秒";
55          return result;
56      }
```

```
57      public Document getXMLContent(String cityHYPY){ //从网络上获取天气 XML 文档
58          Document doc=null;
59          try{
60              URL url=new URL("http://www.google.com/ig/api?"+
61                              "hl=en&weather="+cityHYPY+",china"); //获得 URL
62              DocumentBuilderFactory factory = DocumentBuilderFactory.
newInstance();
63              DocumentBuilder builder=factory.newDocumentBuilder();
64              doc=builder.parse(url.openStream());    //获取 XML 文档对象
65          }catch(Exception e){                        //捕获异常
66              e.printStackTrace();                    //打印堆栈信息
67          }
68          return doc;
69      }
70      public byte[] getImgBytesInfo(String subPath) { //从网络上获取图片数据的方法
71          byte[] result=null;                         //声明 byte 数组
72          try{
73             URL url=new URL("http://www.google.com"+subPath);//URL 对象创建
74             URLConnection uc=url.openConnection();          //打开链接
75             InputStream in=uc.getInputStream();int ch=0;    //获取流数据
76             ByteArrayOutputStream baos = new ByteArrayOutputStream();
77             while((ch=in.read())!=-1) {baos.write(ch);}     //将流数据写入
78             result=baos.toByteArray(); baos.close();in.close();  //关闭流
79           }catch(Exception e){                          //捕获异常
80              e.printStackTrace();                       //异常处理
81           }
82          return result;                                 //返回值
83  } }
```

其中：

- 第 6～12 行为声明成员变量，以及该类的构造器方法的开发，并在构造器方法中调用 initCityList()方法来获取城市信息列表。
- 第 14～28 行为翻译天气情况字符串的方法，主要是翻译成中文。
- 第 29～51 行为翻译当前风向字符串的方法，主要是翻译成中文。
- 第 52～56 行为翻译风速字符串的方法，将风速精确到小数点后两位。
- 第 57～69 行为从网络上获取天气 XML 文档的方法，主要是获得 URL 地址。
- 第 70～83 行为从网络上获取图片数据的方法。

接下来向读者讲解 Activity 类中 initCityList()方法实现过程，该方法主要用于初始化城市列表，将代码插入到上述代码的第 13 行。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_7/src/com.bn.ex8g/Sample8_7_Activity.java

```
1   public void initCityList() {                    //初始化城市列表
2       String citysInfo=this.getResources().getString(R.string.citylist);
                                                    //获取城市列表信息字符串
3       cityStrArray=citysInfo.split("\\|");cityCount=cityStrArray.length/2;
                                                    //切分城市列表字符串获取列表
4           BaseAdapter ba=new BaseAdapter() {      //为 ListView 准备内容适配器
5           public int getCount() {
6               return cityCount;                   //总共 cityCount 个选项
7           }
8           public Object getItem(int arg0) { return null; }//重写的 getItem 方法
9           public long getItemId(int arg0) { return 0; } //重写的 getItemId 方法
10          public View getView(int arg0, View arg1, ViewGroup arg2) {
11              TextView tv=new TextView(Sample8_7_Activity.this);
                                                        //初始化 TextView
```

```
    12                tv.setText(cityStrArray[arg0*2]);              //设置内容
    13                tv.setTextSize(24);                            //设置字体大小
    14                tv.setTextColor(Sample8_7_Activity.this.getResources().
getColor(R.color.white));
    15                tv.setPadding(5,5,5,5);                        //设置四周留白
    16                tv.setGravity(Gravity.LEFT);                   //设置摆放位置
    17                return tv;                                     //返回 TextView
    18        }   };
    19        ListView lv=(ListView)this.findViewById(R.id.ListView01);
                                                                     //得到 ListView 的引用
    20        lv.setAdapter(ba);                                  //为 ListView 设置内容适配器
    21        lv.setOnItemClickListener(                             //设置选项被单击的监听器
    22            new OnItemClickListener(){
    23                public void onItemClick(AdapterView<?> arg0, View arg1, int
arg2,long arg3) {
    24                    try{
    25                        String cityHYPY=cityStrArray[arg2*2+1];
                                                                     //获取选中城市的汉语拼音
    26                        Document doc=getXMLContent(cityHYPY);
                                                                     //天气预报的 XML 文档对象
    27                        NodeList   links   =doc.getElementsByTagName("current_
conditions");
    28                        Element  currentConditions=(Element)links.item(0);
                                                                     //得到 Element 对象
    29                        setContentView(R.layout.detail);   //切换到天气明细 View
    30                        String cStr=currentConditions.getElementsByTagName
("condition")
    31                        .item(0).getAttributes().item(0).getNodeValue();
                                                                     //设置当前天气
    32                        TextView tv=(TextView)findViewById(R.id.TextView01);
    33                        tv.setText("当前天气: "+getCondition(cStr));
                                                                     //设置当前温度
    34                        cStr=currentConditions.getElementsByTagName("temp_c")
    35                        .item(0).getAttributes().item(0).getNodeValue();
                                                                     //得到 cStr
    36                        tv=(TextView)findViewById(R.id.TextView02);
    37                        tv.setText("当前温度: "+cStr+"\u00baC");//设置 TextView 内容
    38                        cStr=currentConditions.getElementsByTagName("humidity")
    39                        .item(0).getAttributes().item(0).getNodeValue();
                                                                     //设置当前湿度
    40                        tv=(TextView)findViewById(R.id.TextView03);
    41                        tv.setText(cStr.replaceFirst("Humidity: ", "当前湿度:
"));/设置 TextView 内容
    42
    cStr=currentConditions.getElementsByTagName("wind_condition")
    43                        .item(0).getAttributes().item(0).getNodeValue();
                                                                     //设置当前风向风速
    44                        String directionStr=cStr.replaceAll("Wind:|at.*", "").
trim();
    45                        String speed=cStr.replaceAll("Wind[A-Za-z\\s:]*| mph",
"").trim();
    46                        tv=(TextView)findViewById(R.id.TextView04);
    47                        tv.setText("当前风向: "+getDirection(directionStr));
                                                                     //设置 TextView 内容
    48                        tv=(TextView)findViewById(R.id.TextView05);
    49                        tv.setText("当前风速: "+getSpeed(speed));
    50                        cStr=currentConditions.getElementsByTagName("icon").
item(0).getAttributes()
    51                        .item(0).getNodeValue();       //获取并设置天气预报图标
```

```
52                          byte[] imgData=getImgBytesInfo(cStr);  //存入数组
53                          Bitmap broadcastBmp=
54                          BitmapFactory.decodeByteArray(imgData,0,imgData.length);
55                          ImageView iv=(ImageView)findViewById(R.id.ImageView01);
56                          iv.setImageBitmap(broadcastBmp);        //设置图片
57                          tv=(TextView)findViewById(R.id.TextView06);
                                                    //设置城市名称
58                          tv.setText(((TextView)arg1).getText());
                                                    //给返回按钮添加监听器
59                          Button bBack=(Button)Sample8_7_Activity.this.
findViewById(R.id.ButtonBack);
60                          bBack.setOnClickListener(      //按钮监听方法
61                          new OnClickListener(){              //匿名内部类
62                              public void onClick(View v) {  //重写的 onClick 方法
63                              setContentView(R.layout.main);initCityList();
64                          } } ); }catch(Exception eq){        //提示信息
65                          String msg="当前此城市天气信息不可用！\n 你可以选择相近的城市查
询！";
66                          Toast.makeText(Sample8_7_Activity.this, msg, Toast.
LENGTH_LONG).show();
67                          setContentView(R.layout.main);          //跳转到主界面
68                          initCityList();                 //调用 initCityList 方法
69  }   }   }   );   }
```

其中：

- 第 3 行为获取城市列表信息。
- 第 4～20 行为 Adapter 适配器的开发，并为 ListView 设置城市名称内容适配器。
- 第 21～69 行为 ListView 列表项的单击事件监听器开发。实现城市具体天气信息界面的显示，并获取风速、风向、湿度等具体天气内容。

实例 8　旅游城市的介绍

接下来向读者介绍一款城市介绍软件，进入本软件首先显示的是城市名称的列表，用户可以单击城市名称，触发城市名称的监听事件，进入城市的详细介绍界面。

【实例描述】

该款小软件用于显示某些城市的相关介绍信息，包括历史、文化等内容。当用户需要对某一城市进行了解时，可以通过本软件选择相应的城市，对其进行一定的了解，包括代表性的图片及相应文字说明。

本实例的运行效果如图 8-18 和图 8-19 所示。

提示：当软件运行时，首先进入的是如图 8-18 所示的城市列表界面。当用户单击某一城市名称时，界面会跳转到城市信息介绍界面，如图 8-19 所示为北京介绍信息界面。

【实现过程】

在本软件的开发过程中，主要应用了 ListView 实现技术，Adapter 适配器实现技术，并实现了 ListView 项的单击监听器事件。

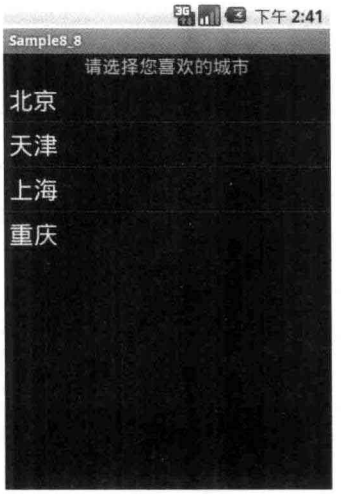


图 8-18　城市列表界面

图 8-19　北京介绍信息界面

【代码解析】

在本部分将为读者详细介绍本软件的开发过程。首先必须在 AndroidManifest.xml 中添加 Internet 网络权限，代码如下。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_8/AndroidManifest.xml

```
1   <uses-permission android:name="android.permission.INTERNET"/>  <!--Internet 权
限-->
```

上述代码介绍的是在 AndroidManifest.xml 中添加 Internet 网络权限，接下来为读者介绍的是本软件主界面布局的开发，代码如下。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_8/res/layout/main.xml

```
1   <?xml version="1.0" encoding="utf-8"?>      <!--版本号 编码方式-->
2   <LinearLayout
3       xmlns:android="http://schemas.android.com/apk/res/android"
4               android:orientation="vertical"
5               android:layout_width="fill_parent"
6               android:layout_height="fill_parent"
7     >                                          <!-- LinearLayout 的属性-->
8     <TextView
9               android:layout_width="fill_parent"
10              android:layout_height="wrap_content"
11              android:text="请选择您喜欢的城市"
12              android:textColor="@color/green"
13              android:textSize="18dip"
14    android:gravity="center_horizontal"      >
15    />                                         <!--TextView 控件设置-->
16    <ListView
17                  android:id="@+id/ListView01"
18                  android:layout_width="fill_parent"
19              android:layout_height="wrap_content"
20                  android:choiceMode="singleChoice"
21    >                                          <!--ListtView 控件设置-->
22    </ListView>
23  </LinearLayout>
```

提示：通过上述 xml 文件的编写设置，实现了软件子界面的开发。其中总布局是线性布局，并实现了 TextView 控件和 ListView 控件的应用开发。

上述代码介绍的是本程序主界面的开发，接下来为读者介绍的是城市信息介绍界面 xml 布局，代码如下所示。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_8/res/layout/detail.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
              android:id="@+id/ScrollView01"
          android:layout_width="fill_parent"
      android:layout_height="wrap_content">        <!-- LinearLayout 的属性-->
       <LinearLayout
              android:orientation="vertical"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
              android:gravity="center_horizontal"
           >                                       <!-- LinearLayout 的属性-->
         <ImageView
               android:id="@+id/ImageViewDetail"
                    android:layout_width="wrap_content"
                    ight="wrap_content"
         >                                         <!-- ImageView 的属性-->
         </ImageView>
         <TextView
                    android:id="@+id/TextViewDetail"
                    android:layout_width="fill_parent"
                    android:layout_height="wrap_content"
                    android:textColor="@color/white"
                    android:textSize="18dip"
                    android:gravity="left"
         />                                        <!--TextView 控件设置-->
         <Button
                    android:layout_width="fill_parent"
                    android:layout_height="wrap_content"
                    android:text="返回"
                    android:id="@+id/ButtonBack"   <!--控件 id-->
         />                                        <!-- Button 控件设置-->
       </LinearLayout>
</ScrollView>
```

提示：通过上述 xml 文件的编写设置，实现了软件子界面的开发。其中总布局是线性布局，并实现了 Button 控件、TextView 控件，以及 ImageView 控件的应用开发。

上述代码介绍的是城市信息的界面，接下来为读者介绍本软件 Sample8_8_Activity 类的开发过程，代码如下所示。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_8/src/com.bn.ex8h/Sample8_8_Activity.java

```
package com.bn.ex8h;                                    //声明包
import java.io.ByteArrayOutputStream;                   //导入相关包
......// 该处省略了导入相关包的代码，读者可自行查阅随书光盘中源代码
import android.widget.TextView;                         //导入相关包
public class Sample8_8_Activity extends Activity {
    String[] citylistStrArray;                          //城市列表信息
    String cityListStr;                                 //城市列表信息
```

```
8       public void onCreate(Bundle savedInstanceState) {    //重写onCreate()方法
9           super.onCreate(savedInstanceState);
10          setContentView(R.layout.main);                    //显示主界面
11          cityListStr=getCityStringInfo("list_contents.txt"); //获取城市简介内容
12          initListView(cityListStr);                        //初始化城市列表
13      }
14      public byte[] getCityBytesInfo(String subPath){       //获取城市简介内容
15          byte[] result=null;                              //声明byte数组数据流缓冲
16          try{
17              URL url=new URL(Constant.ADD_PRE+subPath); //创建URL对象0
18              URLConnection uc=url.openConnection()       //建立连接
19              InputStream in=uc.getInputStream();     //获取InputStream流对象
20              int ch=0;                                   //局部变量ch为0
21              ByteArrayOutputStream baos = new ByteArrayOutputStream();
22              while((ch=in.read())!=-1) {                 //判断是否符合条件
23                  baos.write(ch);                         //写入输出流
24              }
25              result=baos.toByteArray();              //读取信息存入result中
26              baos.close();                           //关闭流对象
27              in.close();                             //关闭InputStream流对象
28          }catch(Exception e){                        //捕获异常
29              e.printStackTrace();                    //异常处理
30          }
31          return result;                              //返回result
32      }
33      public String getCityStringInfo(String subPath){ //获取指定路径的信息字符串
34          String result=null;                         //指定字符串为空
35          try{
36              URL url=new URL(Constant.ADD_PRE+subPath); //URL对象创建
37              URLConnection uc=url.openConnection();      //建立连接
38              InputStream in=uc.getInputStream();         //获取流数据
39              int ch=0;                                   //ch值为0
40              ByteArrayOutputStream baos = new ByteArrayOutputStream();
41              while((ch=in.read())!=-1) {
42                  baos.write(ch);                         //将数据写入流缓冲
43              }
44              byte[] bb=baos.toByteArray();           //将流缓冲转换成byte数组
45              baos.close();                           //关闭流缓冲
46              in.close();                             //关闭流数据
47              result=new String(bb,"UTF-8");          //将byte数组转换成字符串
48              result=result.replaceAll("\\r\\n","\n");    //回车换行符设置
49          }catch(Exception e){
50              e.printStackTrace();                    //异常处理
51          }
52          return result;                              //返回结果
53      }
54      /*该处省略了初始化城市列表的方法，将在后面给出*/
55  }
```

其中：

- 第 14～32 行为获取城市简介内容，通过 URL 连接，从 inputStream 流中读取数据并写入缓冲，进而获取数据字符串信息。
- 第 33～53 行为获取指定路径的信息字符串，首先是获得 URL 连接，然后写入输出流，最后再将流缓冲中的数据转换到 byte 数组。

上述代码介绍的是本程序的 Activity 的框架，接下来向读者介绍的是 ininListView()方法的

具体开发过程，将下列代码插入到上述代码的第 54 行。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_8/src/com.bn.ex8h/Sample8_8_Activity.java

```
1   public void initListView(String cityListInfo) {          //初始化城市列表
2       citylistStrArray=cityListInfo.split("\\|");
3       final int count=citylistStrArray.length/3;           //城市数量
4       BaseAdapter ba=new BaseAdapter() {          //为 ListView 准备内容适配器
5           public int getCount() {
6           return count;                                    //总共 5 个选项
7       }
8       public Object getItem(int arg0) { return null; }     //重写的 getItem 方法
9       public long getItemId(int arg0) { return 0; }   //重写的 getItemId 方法
10      public View getView(int arg0, View arg1, ViewGroup arg2) {
11          TextView tv=new TextView(Sample8_8_Activity.this); //初始化 TextView
12          tv.setText(citylistStrArray[arg0*3]);                //设置内容
13          tv.setTextSize(24);                                  //设置字体大小
14          tv.setTextColor(Sample8_8_Activity.this.getResources().getColor
(R.color.white));
15          tv.setPadding(5,5,5,5);                          //设置四周留白
16          tv.setGravity(Gravity.LEFT);                     //设置字体对齐方式
17          return tv;
18      } };
19      ListView lv=(ListView)this.findViewById(R.id.ListView01);
                                                             //获取 ListView 对象
20      lv.setAdapter(ba);                          //为 ListView 设置内容适配器
21      lv.setOnItemClickListener(                       //设置选项被单击的监听器
22          new OnItemClickListener(){                       //匿名内部类
23              public void onItemClick(AdapterView<?> arg0, View arg1, int
arg2,long arg3) {
24              String txtPath=citylistStrArray[arg2*3+1];
25              String imgPath=citylistStrArray[arg2*3+2];
26              String cityDetailTxt=getCityStringInfo(txtPath);
                                                             //获取城市介绍文本信息
27              byte[] imgData=getCityBytesInfo(imgPath);    //获取城市图片
28              Bitmap
cityBmp=BitmapFactory.decodeByteArray(imgData,0,imgData.length);
29              setContentView(R.layout.detail);             //显示子界面
30              TextView
tv=(TextView)Sample8_8_Activity.this.findViewById(R.id.TextViewDetail);
31              tv.setText(cityDetailTxt);                   //设置显示内容
32              ImageViewiv=(ImageView)Sample8_8_Activity.this.findViewById
(R.id.ImageViewDetail);
33              iv.setImageBitmap(cityBmp);             //设置 ImageView 显示位图
34              Button
bBack=(Button)Sample8_8_Activity.this.findViewById(R.id.ButtonBack);
35              bBack.setOnClickListener(                    //设置返回按钮监听器
36                  new OnClickListener(){                   //匿名内部类
37                      public void onClick(View v) {        //重写的 onClick 方法
38                          setContentView(R.layout.main);//显示城市列表界面
39                      initListView(cityListStr);           //初始化城市列表
40  } }  ); } } ); }
```

其中：

- 第 2～3 行为通过字符串切分获取城市列表中的城市名称，获取城市列表中的城市数量。
- 第 4～20 行为设置 ListView 的内容适配器，每一项都是一个 TextView 控件显示。
- 第 21～40 行为城市列表项单击触控事件，当单击城市列表项时，完成城市天气的具体信息显示。

实例 9　网络音乐播放

下面向读者介绍的是一款用于网络音乐播放的小软件的开发过程，进入本软件首先需要用户输入音乐的网络地址，然后单击“播放”按钮进行音乐的播放。

【实例描述】

本款小软件用于播放网络歌曲，在文本编辑框中输入音乐网络链接，单击“播放”按钮，即可播放当前链接下的网络歌曲；单击“重播”按钮，歌曲重新播放；单击“停止”按钮，歌曲停止播放。

本实例的运行效果如图 8-20 所示。

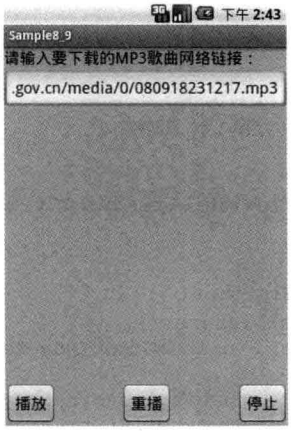


图 8-20　起始界面图

提示：当软件开始运行时，首先进入的是如图 8-20 所示的起始界面，单击“播放”按钮进行音乐的播放，单击“重播”按钮可以重新进行音乐的播放，单击“停止”按钮，将停止音乐的播放。

【实现过程】

在本款软件的开发过程中，主要运用了 URL 网络连接技术、InputStream 流数据读/写技术及 MediaPlayer 音乐播放技术。

【代码解析】

下面向读者详细介绍本软件的开发过程。首先必须在 AndroidManifest.xml 中添加 Internet 网络权限。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_9/AndroidManifest.xml

```
1    <uses-permission android:name="android.permission.INTERNET"/>
<!--Internet 权限-->
```

接下来向读者介绍本软件主界面的开发。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_9/res/layout/main.xml

```
1    <?xml version="1.0" encoding="utf-8"?>
```

```
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:background="#EDAB4A">                    <!--FramLayout-->
   <LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:gravity="bottom|right">
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/bzt"
        android:text="停止"
        android:textColor="#000000"
        android:textSize="18dip"/>                  <!--Button 按钮->
   </LinearLayout>
   <LinearLayout                                    <!--LinearLayout-->
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:gravity="bottom|left">
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/bbf"
        android:text="播放"
        android:textColor="#000000"
        android:textSize="18dip"/>                  <!--Button 按钮-->
   </LinearLayout>
     <LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:gravity="center_horizontal|bottom">
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/bcb"
        android:text="重播"
        android:textColor="#000000"
        android:textSize="18dip"/>                  <!--Button 按钮-->
   </LinearLayout>
   <LinearLayout
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="请输入要下载的 MP3 歌曲网络链接："
        android:textColor="#000000"
        android:textSize="18dip"/>                  <!--TextView-->
       <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/etlj"
        android:text="http://gzw.zj.gov.cn/media/0/080918231217.mp3"
        android:textColor="#000000"
        android:textSize="18dip"
        android:singleLine="true"/>                 <!--EditText 控件-->
   </LinearLayout>
</FrameLayout>
```

提示：通过上述 xml 文件的编写设置，实现了软件主界面的开发。其中总布局是 FrameLayout 帧布局，并实现了 TextView 控件和 EditText 控件，以及 Button 控件的应用开发。

接下来向读者讲解该软件 Sample8_9_Activity 类的开发过程。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_9/src/com.bn.ex8i /Sample8_9_Activity.java

```
1   package com.bn.ex8i;                                          //包名
2   import java.io.File;                                          //导入相关包
3   public class Sample8_9_Activity extends Activity {
4       Button bbf;                                               //播放按钮
5       Button bcb;                                               //重播按钮
6       Button bzt;                                               //停止按钮
7       EditText et;                                              //网址编辑框
8       MediaPlayer mp;                                           //播放器
9       public void onCreate(Bundle savedInstanceState) {         //重写的 onCreate 方法
10          super.onCreate(savedInstanceState);
11          setContentView(R.layout.main);
12          bbf=(Button)this.findViewById(R.id.bbf);              //播放按钮
13          bcb=(Button)this.findViewById(R.id.bcb);              //重播按钮
14          bzt=(Button)this.findViewById(R.id.bzt);              //停止按钮
15          et=(EditText)this.findViewById(R.id.etlj);            //网址编辑框
16          final String wangZhi=et.getText().toString();         //得到网址
17          bbf.setOnClickListener (                              //播放按钮监听器
18            new OnClickListener(){                              //匿名内部类
19                public void onClick(View v) {
20                    new Thread(){                               //创建线程
21                        public void run(){                      //重写的 run()方法
22                            mp = MediaPlayer.create(Sample8_9_Activity.this,
23                                    Uri.parse(wangZhi));//创建播放器
24                            try {
25                                URL myURL=new URL(wangZhi);  //创建 URL 对象
26                                URLConnection conn=myURL.openConnection();
27                                conn.connect();                 //建立连接
28                                InputStream is=conn.getInputStream();
                                                                  //获取流数据
29                                File myTempFile=File.createTempFile("temp",
".mp3");
30                                FileOutputStream  fos=new  FileOutputStream
(myTempFile);
31                                byte[] buffer=new byte[1024];     //创建缓冲
32                                int length=-1;                    //声明长度
33                                while((length=is.read(buffer))!=-1){
                                                                  //while 循环
34                                    fos.write(buffer, 0, length); //写入文件
35                                }
36                                is.close();                       //关闭数据流
37                                fos.close();                      //关闭文件输出流
38                                mp.setDataSource(myTempFile.
getAbsolutePath());
39                                mp.prepare();                     //播放器准备
40                                mp.start();                       //开始播放
41                            } catch (IllegalArgumentException e1) {
                                                                  //捕获异常
42                                e1.printStackTrace();             //异常处理
```

```
43                              }catch (IllegalStateException e1) {   //捕获异常
44                                  e1.printStackTrace();             //异常处理
45                              } catch (IOException e1) {
46                                  e1.printStackTrace();             //异常处理
47                              }
48                      }   }.start();                                //开启线程
49              }   }   );
50          bcb.setOnClickListener(                                   //按钮单击事件监听器
51              new OnClickListener(){                                //匿名内部类
52                  public void onClick(View v) {                     //重写 onClick()方法
53                  if(mp.isPlaying()){
54                      mp.reset();                                   //播放器重新播放
55                  }
56          } } );
57      bzt.setOnClickListener(                                   //停止按钮单击事件监听器
58          new OnClickListener(){
59              public void onClick(View v) {                     //重写 onClick()方法
60                  mp.pause();                                   //播放器停止播放
61              }
62  } ); } }
```

其中：

- 第 4～8 行为声明和创建主界面的应用控件对象，并声明了 MediaPlayer 播放器。
- 第 17～49 行为“播放”按钮单击事件监听器的开发，使其实现了线程内部类的开发，主要用到 URL 对网络歌曲的下载，使用输入、输出流获取歌曲，在使用完后需要关闭输入、输出流。
- 第 50～56 行为“重播”按钮单击事件监听器的开发，通过 MediaPlayer.reset()方法实现了播放器的重播功能。
- 第 57～62 行为“停止”按钮单击事件监听器的开发，通过 MediaPlayer.pause()方法实现了播放器的停止功能。

实例 10　网络歌曲下载软件

下面向读者介绍的是一款用于网络歌曲下载的软件的开发过程，本款软件要求用户输入歌曲网络地址，然后用户输入歌曲存储的名字，单击“下载”按钮，下载歌曲。

【实例描述】

本节所介绍的软件，主要用于将网络歌曲下载到本地并存放在 SD 卡中，同时显示当前 SD 卡中音频文件列表。本实例的运行效果如图 8-21、图 8-22 所示。

提示：软件运行，首先进入是如图 8-21 所示的开始界面，在第一个文本编辑框中，用户可输入 MP3 网络歌曲链接，在第二个文本编辑框中，用户可输入当前网络歌曲的名称。在界面下半部分显示当前 SD 卡中的 MP 3 文件。单击“下载”按钮后，当歌曲下载完成时，软件界面会自动更新，如图 8-22 所示。

【实现过程】

在本款软件的开发过程中，主要应用到 URL 网络链接技术、inputStream 数据流读/写技术

和 ListView 的实现，用户单击“下载”按钮触动监控事件，创建 URL 获取歌曲网络地址，下载此歌曲。

图 8-21　开始界面图

图 8-22　下载完成界面图

【代码解析】

下面向读者详细介绍本款软件的开发过程。首先必须在 AndroidManifest.xml 中添加 Internet 网络权限。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_10/AndroidManifest.xml

```
1    <uses-permission android:name="android.permission.INTERNET"/>
<!--Internet 权限-->
```

接下来向读者介绍本软件主界面的开发。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_10/res/layout/main.xml

```
1    <?xml version="1.0" encoding="utf-8"?>
2    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3       android:orientation="vertical"
4       android:layout_width="fill_parent"
5       android:layout_height="fill_parent"
6       android:background="#EDAB4A"
7       >                                                    <!--LiearLayout 布局-->
8       <TextView
9        android:layout_width="wrap_content"
10       android:layout_height="wrap_content"
11       android:text="请输入要下载的 MP3 歌曲网络链接："
12       android:textColor="#000000"
13       android:textSize="18dip"
14       />                                                  <!--TextView 控件设置-->
15      <EditText
16       android:layout_width="fill_parent"
17       android:layout_height="wrap_content"
18       android:id="@+id/etlj"
19       android:text="http://gzw.zj.gov.cn/media/0/080918231217.mp3"
20       android:textColor="#000000"
21       android:textSize="18dip"
22       android:singleLine="true"
23       />                                                  <!--EditText 控件-->
24      <LinearLayout
25       android:layout_width="fill_parent"
```

```
26      android:layout_height="wrap_content"
27      android:orientation="horizontal"
28      >                                                    <!--LinearLayout-->
29      <EditText
30          android:layout_width="260dip"
31          android:layout_height="wrap_content"
32          android:id="@+id/etmc"
33          android:textColor="#000000"
34          android:textSize="18dip
35          android:singleLine="true"
36          />
37         <Button                                           <!--EditText 控件设置-->
38          android:id="@+id/bxz"
39          android:layout_width="wrap_content"
40          android:layout_height="wrap_content"
41          android:text="下载"
42          android:textColor="#000000"
43          android:textSize="18dip"
44      />                                                   <!--Button 控件设置-->
45    </LinearLayout>
46    <ListView
47      android:paddingTop="10dip"
48      android:id="@+id/lvgq"
49      android:layout_width="fill_parent"
50      android:layout_height="wrap_content"
51      android:divider="#EDAB4A"
52      />                                                   <!--ListView 控件设置-->
53  </LinearLayout>
```

提示：通过上述 xml 文件的编写设置，实现了软件主界面的开发。其中总布局为线性布局，并实现了 TextView 控件、EditText 控件、Button 控件和 ListView 控件的应用开发。

然后向读者讲解该软件 Sample8_10_Activity 类的开发过程，代码如下。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_10/src/com.bn.ex8j /Sample8_10_Activity.java

```
1   package com.bn.ex8j;                                          //包名
2   import java.io.File;                                          //导入相关包
3   ......// 该处省略了导入相关包的代码，读者可自行查阅随书光盘中源代码
4   import android.widget.TextView;                               //导入相关包
5   public class Sample8_10_Activity extends Activity {
6       Button bxz;                                               //下载
7       EditText etwz;                                            //网址链接编辑框
8       EditText etmc;                                            //存储歌曲的自定义名称
9       ListView lv;                                              //歌曲显示列表
10      String uri;                                               //存放网址链接
11      String name;                                              //歌曲自定义名称
12      ArrayList<File> alFiles=new ArrayList<File>();  //用于存放歌曲文件的列表
13      MediaPlayer mediaPlayer;                                  //播放器
14      public void onCreate(Bundle savedInstanceState) {   //重写的 onCreate 方法
15          super.onCreate(savedInstanceState);
16          setContentView(R.layout.main);
17          bxz=(Button)this.findViewById(R.id.bxz);              //下载
18          etwz=(EditText)this.findViewById(R.id.etlj);      //网址链接编辑框
19          etmc=(EditText)this.findViewById(R.id.etmc);      //存储歌曲的自定义名称
20          lv=(ListView)this.findViewById(R.id.lvgq);        //歌曲显示列表
21          uri=etwz.getText().toString();                        //获取网址链接
```

```
22            alFiles.clear();                                    //清空歌曲列表
23            getList(alFiles);                                 //获取当前 SD 卡的音频文件列表
24            showCurrentList();                                  //屏幕显示当前歌曲列表
25            bxz.setOnClickListener(                             //下载按钮监听器
26                new OnClickListener(){                          //匿名内部类
27                    public void onClick(View v) {               //重写的 onClick 方法
28                    name=etmc.getText().toString();             //获取歌曲自定义名称
29                    byte[] bname=name.getBytes();               //创建 byte 数组
30                    try {
31                        name=new String(bname,"utf-8");         //为 name 赋值
32                    }catch (UnsupportedEncodingException e1) {  //捕获异常
33                        e1.printStackTrace();                     //打印堆栈信息
34                    }
35                    new Thread(){                                 //线程
36                        public void run(){                      //重写的 run()方法
37                        try {
38                            URL url=new URL(uri);               //获取链接
39                            HttpURLConnection conn = (HttpURLConnection) url.
openConnection();
40                            conn.setConnectTimeout(6* 1000);    //设置时间缓冲
41                            if (conn.getResponseCode() != 200)
42                            throw new RuntimeException("请求 url 失败");
                                                                  //判断链接是否失败
43                            InputStream is = conn.getInputStream();
                                                                  //获取 InputStream 流对象
44                            readAsFile(is,new File("/sdcard/"+name+".mp3"));
45                            alFiles.clear();                    //清空歌曲列表
46                            getList(alFiles);                 //获取当前 SD 卡的音频文件列表
47                            showCurrentList();                  //屏幕显示当前歌曲列表
48                        }catch (Exception e) {                  //捕获异常
49                            e.printStackTrace();                //异常处理
50                    } } }.start();                              //开启线程
51            } } );
52    }
53    /*该处省略了部分方法的代码，将在后面给出*/
54}
```

其中：

- 第 6～20 行为主界面应用控件对象的声明，及通过调用 findViewById()方法的对象获取。
- 第 22～24 行为显示当前 SD 卡音频文件列表，首先应当调用 clear()方法来清空旧的文件列表，再获取新的文件列表并通过 ListView 显示在界面上。
- 第 25～51 行为“下载”按钮单击事件监听器的应用开发，在其中通过 URL 连接及流数据的读/写来实现网络歌曲的下载。

接下来介绍的是网络流数据的读/写方法及文件列表显示方法，将代码插入到上述代码的第 53 行。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_10/src/com.bn.ex8j/Sample8_10_Activity.java

```
1   public static void readAsFile(InputStream inStream, File file) throws Exception{
2       FileOutputStream outStream = new FileOutputStream(file);
                                                                  //自定义 FileOutputStream
3       byte[] buffer = new byte[1024];                           //自定义 byte[]缓冲
4       int len = -1;
5       while( (len = inStream.read(buffer)) != -1 ){             //判断是否完全写入
6           outStream.write(buffer, 0, len);                      //写入文件
7       }
```

```
        outStream.close();                                  //关闭 FileOutputStream 流
        inStream.close();                                   //关闭 InputStream 流
    }
    public ArrayList<File> getList(ArrayList<File> fileList){
                                                            //用于获取当前 SD 卡音频文件
        File[] sdFiles=new File("/sdcard").listFiles();     //SD 卡中文件列表
        for(File f:sdFiles){                                //增强的 for 循环
            if(f.getName().endsWith(".mp3")){               //判断文件名的结尾
                fileList.add(f);                            //添加到音频文件列表中
        } }
        return fileList;                                    //返回集合
    }
    public void showCurrentList(){                          //屏幕显示当前歌曲列表
        BaseAdapter songsListAdapter=new BaseAdapter(){
        public int getCount() {
            return alFiles.size();                      //BaseAdapter 的操作个数
        }
        public Object getItem(int position) {               //重写的 getItem 方法
            return null;                                    //返回值为空
        }
        public long getItemId(int position) {               //重写的 getItem 方法
            return 0;                                       //返回值为 0
        }
        public View getView(int position, View convertView,ViewGroup parent) {
            LinearLayout ll=new LinearLayout(Sample8_10_Activity.this);
                                                        //用于存放每条操作中的控件
            ll.setOrientation(LinearLayout.HORIZONTAL);//控件存放顺序为水平放置
            ll.setPadding(15, 5, 5, 5);                 //设置 LinearLayout 的留白
            TextView tv=new TextView(Sample8_10_Activity.this);
            tv.setText(alFiles.get(position).getName());    //设置内容
            tv.setTextSize(18);                             //设置字体大小
            tv.setTextColor(Color.BLACK);
            tv.setPadding(5,5,5,5);                         //设置四周留白
            tv.setGravity(Gravity.LEFT);                    //左对齐
            ll.addView(tv);                             //添加到 LinearLayout 中
            return ll;
        } };
        lv.setAdapter(songsListAdapter);                    //锁定 Adapter 设置
        lv.setOnItemClickListener(                      //为 ListView 各个子项添加监听器
            new OnItemClickListener(){                      //匿名内部类
                public void onItemClick(AdapterView<?> arg0,
                View arg1,int arg2, long arg3) {        //重写的 onItemClick 方法
                    File currentFile=new File(alFiles.get(arg2).getPath());
                    if(mediaPlayer!=null){              //判断播放器是否正在使用
                        mediaPlayer.release();          //释放播放器资源
                    }
                    mediaPlayer = new MediaPlayer();    //自定义播放器
                    try {
                        mediaPlayer.setDataSource(currentFile.
getAbsolutePath());
                        mediaPlayer.prepare();                  //准备播放
                        mediaPlayer.start();                    //开始播放
                    } catch (IllegalArgumentException e) {      //捕获异常
                        e.printStackTrace();                    //异常处理
                    } catch (IllegalStateException e) {         //捕获异常
                        e.printStackTrace();                    //异常处理
                    } catch (IOException e) {                   //捕获异常
```

```
62                          e.printStackTrace();                    //打印信息
63   } } } ); }
```

其中：

- 第 1～10 行为读取网络 InputStream 数据写入到 byte 数组中。
- 第 11～18 行为获取当前 SD 卡中的音频文件列表，并判断是否以“mp3”结尾，如果是则存入音频文件列表中。
- 第 19～63 行为屏幕显示当前歌曲列表的实现方法，其中运用了 ListView 内容适配器，并实现了 ListView 列表项的单击事件，当单击某一列表项时，播放相应的歌曲。

实例 11　下载网络歌曲制作手机铃声

下面向读者介绍手机铃声的设置过程，用户首先输入铃声的网络地址，然后输入存储的铃声的名称，单击“下载手机铃声”按钮下载用户需要的铃声。

【实例描述】

本软件实现了网络歌曲的下载和手机铃声的设置功能。用户可以在第一个编辑框中输入所要设置为手机铃声的歌曲的网络链接，在第二个文本编辑框中输入铃声的名称，当单击“下载手机铃声”按钮后开始下载，单击“设置手机铃声”则弹出对话框供用户设置铃声。

本实例的运行效果如图 8-23 和图 8-24 所示。

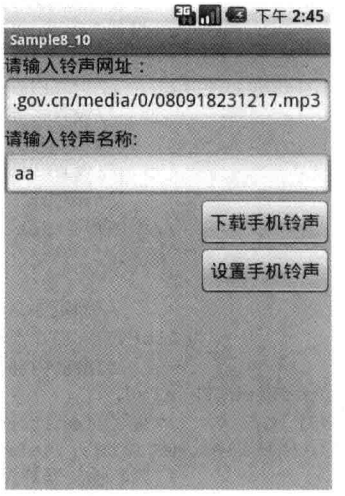


图 8-23　起始界面图

图 8-24　铃声设置界面图

提示：当软件开始运行时，首先进入的是如图 8-23 所示的开始界面，当单击“设置手机铃声”按钮时，弹出对话框供用户设置铃声，如图 8-24 所示。

【实现过程】

在该软件开发过程中，主要应用了 URL 网络链接技术、inputStream 数据流读/写技术，并通过 Intent 应用来调用系统的铃声设置，以实现手机铃声的设置。

【代码解析】

下面向读者详细介绍本软件的开发过程。

首先必须在 AndroidManifest.xml 中添加 Internet 网络权限和手机铃声设置权限。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_11/AndroidManifest.xml

```
    <uses-permission android:name="android.permission.INTERNET"/>
        <uses-permission  android:name="android.permission.MOUNT_UNMOUNT_
FILESYSTEMS"/>   <!--手机铃声设置权限  -->
```

接下来向读者介绍本软件主界面的开发。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_11/res/layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:background="#EDAB4A">                          <!--LinearLayout-->
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="请输入铃声网址："
        android:textColor="#000000"
        android:textSize="18dip"  />                      <!--TextView 控件设置-->
   <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="http://gzw.zj.gov.cn/media/0/080918231217.mp3"
        android:textColor="#000000"
        android:textSize="18dip"
        android:singleLine="true"
        android:id="@+id/etwz"/>                          <!--EditText 控件设置-->
   <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#000000"
        android:textSize="18dip"
        android:singleLine="true"
        android:text="请输入铃声名称:"/>                  <!--TextView 控件设置-->
   <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:textColor="#000000"
        android:textSize="18dip"
        android:singleLine="true"
        android:id="@+id/etmc"/>                          <!--EditText 控件设置-->
   <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:gravity="right">
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/bxz"
            android:text="下载手机铃声"
            android:textColor="#000000"
            android:textSize="18dip"/>                    <!--Button 控件设置-->
   </LinearLayout>
   <LinearLayout
        android:layout_width="fill_parent"
```

```
49          android:layout_height="wrap_content"
50          android:gravity="right">
51          <Button
52              android:layout_width="wrap_content"
53              android:layout_height="wrap_content"
54              android:id="@+id/bsz"
55              android:text="设置手机铃声"
56              android:textColor="#000000"
57              android:textSize="18dip"/>              <!--Button 控件设置-->
58      </LinearLayout>
59  </LinearLayout>
```

提示：通过上述 xml 文件的编写设置，实现了软件主界面的开发。其中总布局为线性布局并实现了 TextView 控件、EditText 控件和 Button 控件的应用开发。

然后向读者讲解该软件 Sample8_11_Activity 类的开发过程，代码如下。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_11/src/com.bn.ex8k/Sample8_11_Activity.java

```
1   package com.bn.ex8k;                                          //声明包
2   import java.io.File;                                          //导入相关包
3   ......// 该处省略了导入相关包的代码，读者可自行查阅随书光盘中源代码
4   import android.widget.Toast;                                  //导入相关包
5   public class Sample8_11_Activity extends Activity {
6       public static final int RINGTONE_PICKED=0;                //静态常量
7       private static final String FileRingtone =
8               Environment.getExternalStorageDirectory()+ "/music/ringtones";
9       Button bxz;                                               //下载按钮
10      Button bsz;                                               //设置按钮
11      EditText etwz;                                            //网址编辑框
12      EditText etmc;                                            //铃声名称编辑框
13      public void onCreate(Bundle savedInstanceState) {         //重写的 onCreate 方法
14          super.onCreate(savedInstanceState);
15          setContentView(R.layout.main);
16          bxz=(Button)this.findViewById(R.id.bxz);              //下载按钮
17          bsz=(Button)this.findViewById(R.id.bsz);              //设置按钮
18          etwz=(EditText)this.findViewById(R.id.etwz);          //网址编辑框
19          etmc=(EditText)this.findViewById(R.id.etmc);          //铃声名称编辑框
20          bxz.setOnClickListener (                              //下载按钮监听器
21              new OnClickListener(){
22                  public void onClick(View v) {                 //重写的 onClick 方法
23                      String wzStr=etwz.getText().toString();
                                                                  //获取当前输入框中的网址
24                      String nameStr=etmc.getText().toString();
                                                                  //获取当前输入框中的铃声名称
25                      if(isFileExist(FileRingtone)){            //判断是否符合条件
26                          File  newFile=new  File("/sdcard/music/ringtones/
"+nameStr+".mp3");
27                          InputStream is=getConnect(wzStr);
                                                                  //获取网址链接，得到数据流
28                      try {
29                          readAsFile(is, newFile);              //调用 readAsFile 方法
30                      } catch (Exception e) {                   //捕获异常
31                          e.printStackTrace();                  //异常处理
32                      }
33          }}});
34          bsz.setOnClickListener(                               //按钮监听方法
35              new OnClickListener(){                            //匿名内部类
```

```
36                  public void onClick(View v) {          //重写的 onClick 方法
37                      if(isFileExist(FileRingtone)){     //捕获异常
38                          setPhoneMusic();               //设置手机铃声
39                      }
40  } } ); }
41  /*该处省略了部分方法的代码，将在后面给出*/
42 }
```

提示：第 4～16 行为成员变量的声明和创建，第 17～29 行为下载按钮的单击事件监听器开发，第 30～35 行为设置按钮的单击事件监听器的开发。

接下来向读者介绍的是判断当前文件是否存在，不存在则创建方法、获取网址链接，得到数据流方法、读取网络 InputStream 方法、设置当前文件存放音乐为手机铃声方法和界面跳转方法的开发。将下列代码插入到上面代码的第 41 行。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_11/src/com.bn.ex8k/Sample8_11_Activity.java

```
1   public boolean isFileExist(String filePath){ //判断当前文件是否存在，不存在则创建
2       boolean result=false;                              //设置 boolean 的返回值
3       File f=new File(filePath);
4       if(!f.exists()){                                   //如果当前文件不存在
5           if(f.mkdirs()){                                //创建当前目录文件
6               result=true;                               //返回值为 true
7           }else{                                         //文件创建失败
8               result=false;                              //返回值为 false
9           }
10      }else{                                             //当前文件存在
11          result=true;                                   //返回值为 true
12      }
13      return result;                                     //返回返回值
14  }
15  public InputStream getConnect(String uri) {            //获取网址链接，得到数据流
16      InputStream is=null;                               //输入流为空
17      try {
18          URL myUrl=new URL(uri);                        //获取 URL
19          URLConnection conn=myUrl.openConnection();     //开启链接
20          conn.connect();                                //连接链接
21          is=conn.getInputStream();                      //获得链接数据流
22      }catch (Exception e) {                             //捕获异常
23          e.printStackTrace();                           //异常处理
24      }
25      return is;
26  }
27  public static void readAsFile(InputStream inStream, File file) throws Exception{
28      FileOutputStream outStream = new FileOutputStream(file);
                                                           //自定义 FileOutputStream 对象
29      byte[] buffer = new byte[1024];                    //自定义 byte[]缓冲
30      int len = -1;                                //用于记录 InputStream 流的读入字节长度
31      while( (len = inStream.read(buffer)) != -1 ){
32          outStream.write(buffer, 0, len);               //写入文件
33      }
34      outStream.close();                                 //关闭 FileOutputStream 流
35      inStream.close();                                  //关闭 InputStream 流
36  }
37  public void setPhoneMusic(){                           //设置当前文件存放音乐为手机铃声
38      Intent intent =
39      new Intent(RingtoneManager.ACTION_RINGTONE_PICKER); //打开系统铃声设置
```

```
40        intent.putExtra(RingtoneManager.EXTRA_RINGTONE_TYPE,
41        RingtoneManager.TYPE_RINGTONE);                    //在 Intent 中写入数据
42        intent.putExtra(RingtoneManager.EXTRA_RINGTONE_TITLE,
43         "设置来电铃声");                                    //设置显示的标题
44        startActivityForResult(intent,
45         RINGTONE_PICKED);                                 //跳转界面，显示铃声设置对话框
46    }
47    public void onActivityResult(int requestCode,int resultCode,Intent data){
48        super.onActivityResult(requestCode, resultCode, data);   //调用父类
49        if(resultCode!=RESULT_OK){                         //resultCode 是否为 RESULT_OK
50            return;                                        //返回
51        }
52        switch(requestCode){                               //判断 requestCode 的值
53            case RINGTONE_PICKED:                          //开启手机铃声
54            try{
55                Uri pickedUri=
56
data.getParcelableExtra(RingtoneManager.EXTRA_RINGTONE_PICKED_URI);
57                if(pickedUri!=null){                       //判断 pickedUri 是否为空
58                    RingtoneManager.setActualDefaultRingtoneUri( //设置铃声链接
59                    Sample8_11_Activity.this,
60                    RingtoneManager.TYPE_RINGTONE,         //类型
61                    pickedUri);                            //URL 地址
62                }
63            }catch(Exception e){                           //捕获异常
64                e.printStackTrace();                       //异常处理
65            }
66            break;                                         //退出
67    } }
```

其中：

- 第 1～14 行为判断当前文件是否存在，不存在则创建方法。在该方法中调用了 File.mkdirs()方法来创建文件并判断创建是否成功。
- 第 15～26 行为获取网址链接，得到数据流方法，在该方法中首先获得 URL，并获得连接，之后获得连接的数据流。
- 第 27～36 行表示通过输出流，将输入的信息写入 byte 数组中，最后需要关闭输入、输出流。
- 第 37～46 行设置当前文件存放音乐为手机铃声的方法，主要是通过 Intent 打开系统铃声设置，并在 Intent 中写入数据，同时设置显示的标题。
- 第 47～67 行为界面跳转方法的开发，首先需要判断 resultCode 是否为 RESULT_OK，如果不为 RESULT_OK 则返回。如果 resultCode 与 RESULT_OK 相等则得到 URI，并判断 pickedUri 是否为空，不为空则设置铃声。

实例 12　下载网络图片制作手机背景

下面向读者介绍的是一款手机背景图设置的小软件的开发，本软件要求用户输入背景图片的网络地址，然后单击“设置手机背景”按钮获取图片，显示在界面中，同时背景设置成功。

【实例描述】

本软件实现了网络图片的下载和将网络图片设置为手机背景图的功能。本实例的运行效果如图 8-25、图 8-26 和图 8-27 所示。

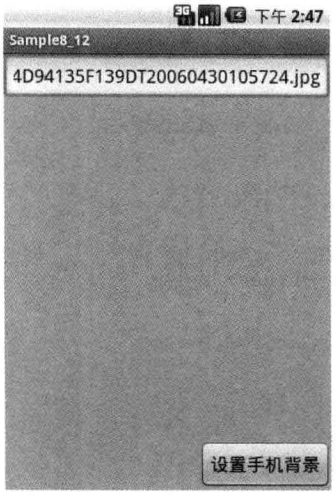


图 8-25　开始界面

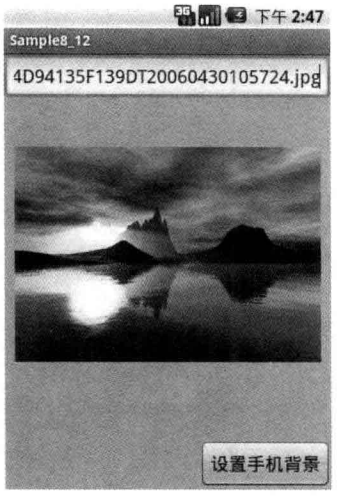


图 8-26　下载手机背景图

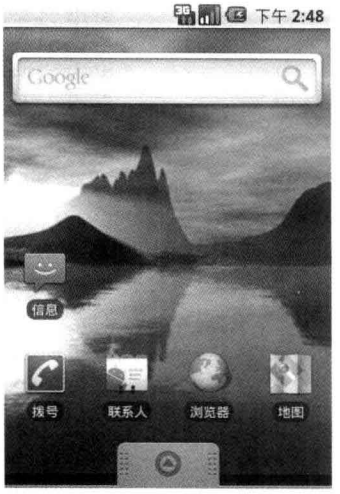


图 8-27　手机背景显示

提示：当软件开始运行后，首先进入如图 8-25 所示的开始界面，单击“设置手机背景”按钮，界面效果如图 8-26 所示，图 8-27 为手机背景设置效果图。

【实现过程】

本软件开发过程中，主要应用了 URL 网络链接技术获取网络背景图片，单击“设置手机背景”按钮，获取背景图片显示在界面中，并通过调用 setWallpaper()方法来设置当前手机背景图。

【代码解析】

下面向读者详细介绍本软件的开发过程。首先必须在 AndroidManifest.xml 中添加 Internet 网络权限。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_12/AndroidManifest.xml

```
1   <uses-permission android:name="android.permission.INTERNET"/>   <!-- 添加设置
手机背景图权限 -->
2   <uses-permission android:name="android.permission.SET_WALLPAPER"/>
```

接下来向读者介绍本软件主界面的开发。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_12/res/layout/main.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <FrameLayout
3       xmlns:android="http://schemas.android.com/apk/res/android"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent"
6      android:background="#EDAB4A"
7      >
8      <LinearLayout
9       android:layout_width="fill_parent"
10      android:layout_height="fill_parent"
11      android:gravity="bottom|right"
12      >                                          <!--LinearLayout-->
13      <Button
14          android:layout_width="wrap_content"
15          android:layout_height="wrap_content"
```

```
16          android:id="@+id/bbjt"
17          android:text="设置手机背景"
18          android:textSize="18dip"
19          android:textColor="#000000"
20          />                                          <!--Button 按钮设置-->
21      </LinearLayout>
22      <LinearLayout
23       android:layout_width="fill_parent"
24       android:layout_height="fill_parent"
25       android:orientation="vertical"
26       >                                              <!--LinearLayout -->
27       <EditText
28          android:layout_width="fill_parent"
29          android:layout_height="wrap_content"
30          android:id="@+id/etwz"
31  android:text="http://image2.sina.com.cn/bj/t/2006-04-30/
U1421P52T4D94135F139DT20060430105724.jpg"
32          android:singleLine="true"
33          android:textSize="18dip"
34          android:textColor="#000000"
35          />                                          <!--EditText 设置-->
36       <ImageView
37          android:layout_width="fill_parent"
38          android:layout_height="300dip"
39          android:padding="10dip"
40          android:id="@+id/imtp"
41          >                                           <!--ImageView 设置-->
42       </ImageView>
43      </LinearLayout>
44  </FrameLayout>
```

提示：通过上述 xml 文件的编写设置，实现了软件主界面的开发。其中总布局为 FrameLayout 帧布局，并实现了 EditText 控件、Button 控件和 ImageView 控件的应用开发。

然后向读者讲解该软件 Sample8_12_Activity 类的开发过程，代码如下所示。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_12/src/com.bn.ex8l/Sample8_12_Activity.java

```
1   package com.bn.ex8l;                                    //包名
2   import java.io.InputStream;                             //导入相关包
3   ......// 该处省略了导入相关包的代码，读者可自行查阅随书光盘中源代码
4   import android.widget.EditText;                         //导入相关包
5   public class Sample8_12_Activity extends Activity {
6       Button bbjt;                                        //设置背景图按钮
7       EditText etwz;                                      //网站编辑框
8       ImageView imtp;                                     //背景图显示
9       Bitmap bm;                                          //图片对象
10      public void onCreate(Bundle savedInstanceState) {   //重写的onCreate方法
11          super.onCreate(savedInstanceState);
12          setContentView(R.layout.main);
13          bbjt=(Button)this.findViewById(R.id.bbjt);      //设置背景图按钮、
14          etwz=(EditText)this.findViewById(R.id.etwz);    //网站编辑框
15          imtp=(ImageView)this.findViewById(R.id.imtp);   //背景图显示
16          bbjt.setOnClickListener(                        //设置手机背景图按钮单击监听器
17              new OnClickListener(){
18                  public void onClick(View v) {
19                  String url = etwz.getText().toString();//获取当前图片链接信息
20                      try{
21                          URL aryURI = new URL(url);      //创建 URL 对象
```

```
22                          URLConnection conn = aryURI.openConnection();
23                          conn.connect();                    //打开连接
24                          InputStream is = conn.getInputStream();
                                                               //获取 InputStream
25                          bm = BitmapFactory.decodeStream(is);
                                                               //从输入流构建 Bitmap
26                          is.close();                        //关闭 InputStream
27                          imtp.setImageBitmap(bm);           //显示下载图片
28                          Sample8_12_Activity.this.setWallpaper(bm);
29                      }
30                      catch(Exception e){                    //捕获异常
31                          e.printStackTrace();               //异常处理
32          } } } );
33  } }
```

其中：

- 第 6～9 行表示对本程序的成员变量的声明，该成员变量主要为控件的引用。
- 第 13～15 行为通过调用 findViewById()方法来获取控件对象。
- 第 17～32 行为设置手机背景图按钮单击监听器事件。其中第 26 行为通过调用 setWallpaper()方法来将该图片设置为手机背景图，需要添加设置手机背景图权限。

实例 13　制作 RSS 阅读器

在日常生活中经常需要浏览多个网站，从而获得自己需要的信息，但是这样不仅浪费了时间，而且不利于信息的查找。RSS 阅读器很好地解决了这些问题，其可以管理多个网站，并且可以显示网站的更新信息。本节主要介绍的是在 Android 端 RSS 阅读器的制作。

【实例描述】

运行本程序进入主界面，在主界面单击“开始阅读”按钮，跳转到信息更新的界面，在信息更新界面单击其中的任意一项，进入该信息的详细说明界面。本实例的运行效果如图 8-28、图 8-29 和图 8-30 所示。

图 8-28　程序开始运行的界面

图 8-29　更新信息的界面

图 8-30　更新信息的详细介绍界面

提示：在本程序的运行效果如图 8-28、图 8-29、图 8-30 所示，图 8-28 表示程序开始运行时的界面，图 8-29 表示更新信息的界面，图 8-30 表示更新信息的详细介绍界面。

【实现过程】

在本程序的开发过程中，主要用到在公共类中对私有数据的存储和读取数据，URL 的使用，自定义的适配器，以及集合框架中 List 的应用。

【代码解析】

RSS 阅读器是针对网站更新，所以在开发本程序之前需要获取网页内容，代码如下。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_13/src/com/bn/ex8m 目录下的 Sample8_13_Activity。

```
1   package com.bn.ex8m;                                    //声明包
2   public class News{                                      //创建类
3       private String _title="";                           //私有的成员变量
4       private String _link="";                            //私有的成员变量
5       private String _desc="";                            //私有的成员变量
6       private String _date="";                            //私有的成员变量
7       public String getTitle(){                           //得到标题
8           return _title;                                  //返回标题
9       }
10      public void setTitle(String title){                 //设置标题
11          _title=title;
12      }
13      public String getLink(){                            //得到链接
14          return _link;                                   //返回链接
15      }
16      public void setLink(String link){                   //设置连接
17          _link=link;
18      }
19      public String getDesc(){                            //得到排序方式
20          return _desc;                                   //返回排序方式
21      }
22      public void setDesc(String desc){                   //设置排序
23          _desc=desc;
24      }
25      public String getDate(){                            //得到日期
26          return _date;                                   //返回日期
27      }
28      public void setDate(String date){                   //设置日期
29          _date=date;
30  }       }
```

提示：上面的类主要是通过调用 get 与 set 方法对私有的数据进行更改。其中 get 方法为得到信息，调用 set 方法为设置相应的信息。使用私有的成员变量可以增加安全性。

上面已经介绍了读取信息的方法，接下来将要介绍的是怎样解析 xml 文件，代码如下。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_13/src/com/bn/ex8m 目录下的

Sample8_13_Activity。

```
package com.bn.ex8m;                                          //声明包
 ……//该处省略了导入相关包的代码，读者可自行查阅随书光盘中源代码
import org.xml.sax.helpers.DefaultHandler;                    //导入相关包
public class MyHandler extends DefaultHandler{  //创建继承 DefaultHandler 的类
    ......//该处省略部分成员变量的声明，读者可自行查阅随书光盘中源代码
    public List<News> getParsedData(){                        //得到数据存进 List 中
        return list;                                          //返回 List
    }
    public String getRssTitle(){                          //得到该 Rss 阅读器的标题
        return title;                                         //返回标题
    }
    @Override
    public void startDocument() throws SAXException{          //文件开始方法
        list = new ArrayList<News>();                         //为 list 赋新值
    }
    @Override
    public void endDocument() throws SAXException{}           //文件结束方法
    @Override
    public void startElement(String namespaceURI, String localName,
                    String qName, Attributes atts) throws SAXException{
        if (localName.equals("item")){                        //名称为 item
            this.item_flag = true;                            //item_flag 为 true
            news=new News();                                  //创建新 News 对象
        }else if (localName.equals("title")){                 //名称为 title
            if(this.item_flag){
                this.title_flag = true;                       // item_flag 为 true
            }else{
                this.mainTitle_flag = true;          // mainTitle_flag 为 true
            }
        }else if (localName.equals("link")){                  //名称为 link
            if(this.item_flag){
                this.link_flag = true;                        // link_flag 为 true
            }
        }else if (localName.equals("description")){           //名称为 description
            if(this.item_flag){
                this.desc_flag = true;                        // desc_flag 为 true
            }
        }else if (localName.equals("pubDate")){               //名称为 pubDate
            if(this.item_flag){
                this.date_flag = true;                        //date_flag 为 true
        }
    }}
    @Override
    public void endElement(String namespaceURI, String localName,
                    String qName) throws SAXException{  //重写的方法
        if (localName.equals("item")){                        //名称为 item
            this.item_flag = false;                           //item_flag 为 true
            list.add(news);                                   //list 中添加信息
        }else if (localName.equals("title")){                 //名称为 title
            if(this.item_flag){
                news.setTitle(buf.toString().trim());  //设置该 Rss 的标题
                buf.setLength(0);                             //设置长度
                this.title_flag = false;             //title_flag 为 false
            }else{
                title=buf.toString().trim();                  //得到标题
                buf.setLength(0);                             //设置长度
```

```
57                  this.mainTitle_flag = false;   // mainTitle_flag 为 false
58              }
59          }else if (localName.equals("link")){              //名称为 link
60              if(this.item_flag){
61                  news.setLink(buf.toString().trim());      //设置 Link
62                  buf.setLength(0);                         //设置长度为 0
63                  this.link_flag = false;              //设置 link_flag 为 false
64              }
65          }else if (localName.equals("description")){       //名称为 description
66              if(item_flag){
67                  news.setDesc(buf.toString().trim());      //设置 desc
68                  buf.setLength(0);                         //设置长度
69                  this.desc_flag = false;                   //desc_flag 为 false
70              }
71          }else if (localName.equals("pubDate")){           //名称为 pubDate
72              if(item_flag){
73                  news.setDate(buf.toString().trim());      //设置日期
74                  buf.setLength(0);                         //设置长度为零
75                  this.date_flag = false;                   //date_flag 为 false
76          }
77      }   }
78      @Override
79      public void characters(char ch[], int start, int length){  //重写的方法
80          if(this.item_flag||this.mainTitle_flag){
                                    //如果 item_flag 或 mainTitle_flag 为 true
81              buf.append(ch,start,length);                      //添加进数据
82  }  }   }
```

其中：

- 第 13～15 行表示重写的 startDocument 方法，并将数据存入 List 中。
- 第 19～42 行表示重写的 startElement 方法，文档开始解析时被调用，主要是判断 localName 与 item、title、link、description、pubDate 其中的哪个相等。
- 第 44～77 行表示重写的 endElement 方法，文档结束解析时被调用，依然是判断 localName 与上面提到的几项做比较，并判断出相等的，同时做出处理。

上面介绍了解析文档用的方法，接下来介绍的是自定义适配器的类，代码如下。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_13/src/com/bn/ex8m 目录下的 Sample8_13_Activity。

```
1   package com.bn.ex8m;                                      //声明包
2   ……//该处省略了导入相关包的代码，读者可自行查阅随书光盘中源代码
3   import android.widget.TextView;                           //导入相关包
4   public class MyBaseAdapter extends BaseAdapter{ //创建继承 BaseAdapter 的类
5       private LayoutInflater myInflater;                    //私有的成员变量
6       private List<News> list;
7       public MyBaseAdapter(Context context,List<News> list){   //构造器
8           myInflater=LayoutInflater.from(context);          //为 myInflater 赋值
9           this.list=list;                                   //为 list 赋值
10      }
11      @Override
12      public int getCount(){                                //重写的方法
13          return list.size();                               //设置长度
14      }
15      @Override
16      public Object getItem(int position){                  //重写的方法
17          return list.get(position);                        //返回值
```

```
    }
    @Override
    public long getItemId(int position){                          //重写的方法
        return position;                                          //返回值
    }
    @Override
    public View getView(int position,View convertView,ViewGroup par){
        ViewHolder holder;                                  //得到 ViewHolder 的引用
        if(convertView==null){                                    //判断是否为空
            convertView=myInflater.inflate(R.layout.news_row, null);
            holder=new ViewHolder();                              //为 holder 创建对象
            holder.text=(TextView) convertView.findViewById(R.id.news);
            convertView.setTag(holder);     //将 holder 添加进 convertView 中
        }else{
            holder=(ViewHolder) convertView.getTag();   //为 holder 创建对象
        }
        News tmpNews=(News)list.get(position);                    //得到 News 对象
        holder.text.setText(tmpNews.getTitle());                  //设置文本
        return convertView;                                       //返回 convertView
    }
    private class ViewHolder{                                     //创建的私有方法
        TextView text;
}}
```

提示：本方法的主要目的是为创建列表时添加适配器。要创建自定义的适配器必须继承系统的 BaseAdapter 类，并重写需要重写的方法。

上面已经介绍了自定义的适配器的类，下面主要介绍程序开始时的控制类，代码如下。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_13/src/com/bn/ex8m 目录下的 Sample8_13_Activity。

```
package com.bn.ex8m;                                              //声明包
……//该处省略了导入相关包的代码，读者可自行查阅随书光盘中源代码
import android.widget.EditText;                                   //导入相关包
public class Sample8_13_Activity extends Activity{  //创建继承 Activity 的类
    private Button myButton;                                      //私有的成员变量
    private EditText myEditText;
    @Override
    public void onCreate(Bundle savedInstanceState){
                                                    //继承 Activity 需要重写的方法
        super.onCreate(savedInstanceState);                       //调用父类
        setContentView(R.layout.main);                            //跳转到主界面
        myEditText=(EditText) findViewById(R.id.EditText);
                                                          //为 myEditText 创建对象
        myButton=(Button) findViewById(R.id.Button);     //为 myButton 创建对象
        myButton.setOnClickListener(new Button.OnClickListener(){//添加监听
            @Override
            public void onClick(View v){                          //匿名内部类
                String path=myEditText.getText().toString();//得到路径
                if(path.equals("")){                      //判断得到的路径是否为空
                    showDialog("请输入网址！");                    //显示对话框
                }else{
                    Intent intent=new Intent();                   //创建 Intent
                    intent.setClass(Sample8_13_Activity.this,Reader_1.
class);
                    Bundle b=new Bundle();                        //创建 Bundle 对象
```

```
23                          b.putString("path",path);          //绑定数据
24                          intent.putExtras(b);               //发送数据
25                          startActivityForResult(intent,0);
26      } } } ); }
27      @Override
28      protected void onActivityResult(int requestCode,int resultCode,Intent
data){//重写的方法
29          switch (resultCode){                               //判断 resultCode
30              case 8080:                                     //resultCode 为 8080
31                  Bundle b=data.getExtras();                 //得到绑定的数据
32                  String error=b.getString("error");         //得到错误
33                  showDialog(error);                         //显示对话框
34              break;                                         //退出
35              default:                                       //默认
36              break;                                         //退出
37      } }
38      private void showDialog(String mess){                  //显示对话框方法
39          new AlertDialog.Builder(Sample8_13_Activity.this).setTitle("错误提示")
                                                               //创建 Dialog
40          .setMessage(mess)                                  //设置信息
41          .setNegativeButton("返回", new DialogInterface.OnClickListener(){
42              public void onClick(DialogInterface dialog, int which){}
                                                               //重写的方法
43          }).show();                                         //显示对话框
44  } }
```

其中：

- 第 13-26 行表示为“开始阅读”按钮添加监听器，重写其中的方法，同时判断得到的路径。
- 第 28-37 行表示重写的 onActivityResult 方法，主要是为了判断是否发生错误。
- 第 38-44 行表示显示对话框的方法，设置了其中的信息，并为该对话框添加按钮。

上面已经介绍了程序的控制类，接下来要介绍的是 RSS 阅读器中显示网页更新内容的类，代码如下。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_13/src/com/bn/ex8m 目录下的 Sample8_13_Activity。

```
1   package com.bn.ex8m;                                       //声明包
2   ……//该处省略了导入相关包的代码，读者可自行查阅随书光盘中源代码
3   import android.widget.TextView;                            //导入相关包
4   public class Reader_1 extends ListActivity{//继承 ListActivity 需要重写的方法
5       private TextView myTextView;                           //私有的成员变量
6       private String myTitle;                                //私有的成员变量
7       private List<News> list=null;                          //私有的成员变量
8       @Override
9       public void onCreate(Bundle savedInstanceState){
                                          //继承 ListActivity 类要重写的方法
10          super.onCreate((android.os.Bundle) savedInstanceState); //调用父类
11          setContentView(R.layout.newslist);                 //跳转到更新界面
12          myTextView=(TextView)this.findViewById(R.id.myText);
                                                   //创建 myTextView 对象
13          Intent intent=this.getIntent();                    //创建 Intent 对象
14          Bundle b=intent.getExtras();                       //创建 Bundle 对象
15          String path=b.getString("path");                   //得到路径
16          list=getRss(path);
17          myTextView.setText(myTitle);                       //设置内容
```

```
18          setListAdapter(new MyBaseAdapter(this,list));   //设置适配器
19      }
20      @Override
21      public void onListItemClick(ListView l,View v,int position,long id){
                                                    //重写的 onListItemClick 方法
22          News news=(News)list.get(position);             //得到 news 对象
23          Intent intent=new Intent();                     //创建 Intent 对象
24          intent.setClass(Reader_1.this, Reader_2.class); //
25          Bundle ba=new Bundle();                         //创建 Bundle 对象
26          ba.putString("title", news.getTitle());         //在 ba 对象中添加数据
27          ba.putString("desc", news.getDesc());           //在 ba 对象中添加数据
28          ba.putString("link", news.getLink());           //在 ba 对象中添加数据
29          intent.putExtras(ba);
30          startActivity(intent);                      //调用 startActivity 方法
31      }
32      private List<News> getRss(String path){             //获取 RSS 方法
33          List<News> result=new ArrayList<News>();        //得到 List 对象
34          URL url=null;                                   //URL
35          try{
36              url=new URL(path);                          //得到 URL 对象
37              SAXParserFactory spf=SAXParserFactory.newInstance();
                                                    //得到 SAXParserFactory 对象
38              SAXParser sp=spf.newSAXParser();            //得到 SAXarser 对象
39              XMLReader xr = sp.getXMLReader();           //得到 XMLReader 对象
40              MyHandler myExampleHandler = new MyHandler();
                                                    //得到 MyHandler 对象
41              xr.setContentHandler(myExampleHandler); //设置 ContentHandler
42              xr.parse(new InputSource(url.openStream())); //添加进输入流
43              result=myExampleHandler.getParsedData();    //得到 result
44              myTitle=myExampleHandler.getRssTitle();     //得到标题
45          }catch(Exception e){                            //捕获异常
46               Intent intent=new Intent();                //创建 Intent 对象
47               Bundle b = new Bundle();                   //得到 Bundle 对象
48              b.putString("error",""+e);                  //在 b 中添加信息
49              intent.putExtras(b);
50              Reader_1.this.setResult(8080, intent);      //调用 setResult 方法
51              Reader_1.this.finish();                     //调用 finish 方法
52          }
53          return result;                                  //返回 result
54      }
55  }
```

其中：

- 第 5～7 行表示该类成员变量的声明。
- 第 9～19 行表示继承 ListActivity 类需要重写的方法。
- 第 21～31 行表示继承 ListActivity 重写的 onListItemClick 方法，该方法主要是为创建继承 ListActivity 的类添加监听器。
- 第 32～54 行表示获取 RSS 的方法，该方法主要是通过 URL 获取 RSS 更新的内容。

上面已经介绍了显示 RSS 阅读器内容更新的类，接下来介绍读取更新内容的类，代码如下。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_13/src/com/bn/ex8m 目录下的 Sample8_13_Activity。

```
1   package com.bn.ex8m;                                    //声明包
2   ……//该处省略了导入相关包的代码，读者可自行查阅随书光盘中源代码
```

```
3   import android.widget.TextView;                          //导入相关包
4   public class Reader_2 extends Activity{                  //继承 Activity 需要重写的方法
5       private TextView myTitle;                            //私有的成员变量
6       private TextView myDesc;                             //私有的成员变量
7       private TextView myLink;                             //私有的成员变量
8       @Override
9       public void onCreate(Bundle savedInstanceState){
                                                             //继承 Activity 需要重写的方法
10          super.onCreate(savedInstanceState);              //调用父类
11          setContentView(R.layout.newscontent);            //跳转到具体内容界面
12          myTitle=(TextView) findViewById(R.id.myTitle);   //得到 title 对象
13          myDesc=(TextView) findViewById(R.id.myDesc);     //得到 desc 对象
14          myLink=(TextView) findViewById(R.id.myLink);     //得到 link 对象
15          Intent intent=this.getIntent();                  //创建 Intent 对象
16          Bundle b=intent.getExtras();                     //创建 Bundle 对象
17          myTitle.setText(b.getString("title"));           //设置文本
18          myDesc.setText(b.getString("desc"));             //设置文本
19          myLink.setText(b.getString("link"));             //设置文本
20          Linkify.addLinks(myLink,Linkify.WEB_URLS);       //Linkify 中添加数据
21  } }
```

提示：上面主要介绍的是在 RSS 阅读器中显示网页更新的具体内容。主要是得到 Bundle 对象中的内容，并将得到的内容添加进与之对应的各个 TextView 中。

实例 14　远程下载与安装 Android 程序

Google 公司自从推出 Android 平台到现在获得了迅速的发展，其开发者与使用者也逐渐增多。这就涉及如何下载与安装程序。本节主要介绍如何通过网络下载 Android 的应用程序到手机中，并通过打开 application installer 安装下载的软件。

【实例描述】

运行本程序进入主界面，单击“开始安装”按钮下载相应的程序，在程序下载完成后打开 application installer 安装软件。在该界面单击“Install”按钮将安装该软件，单击“Cancel”按钮退出该安装程序界面，并且在安装完该程序后会自动删除下载的 APK 软件包。

本实例的运行效果如图 8-31、图 8-32 和图 8-33 所示。

提示：在程序的运行效果图 8-31、图 8-32 和图 8-33 中，图 8-31 表示的是程序开始时运行的界面图，图 8-32 表示的是下载程序的界面，图 8-33 表示的是程序下载完成的界面。

【实现过程】

在开发本程序的过程中，主要用到在 xml 文件中对主界面的搭建，打开 application installer 的方法，文件的 I/O 操作，通过 URL 联机并获得链接，同时需要在 AndroidManifest.xml 中声明一系列的权限。

搭建服务器：由于本程序需要连接网络下载 APK 安装包，因此读者需要自己搭建 Tomcat 服务器。本节为读者提供 Tomcat 压缩文件，其位于随书光盘中第 8 章的 Sample8_14_Tomcat

文件夹。解压该 Tomcat 压缩文件，配置 Tomcat 环境即可。本程序用到的 APK 安装文件位于压缩文件的 webapps 目录下的 apkname 文件夹中。

图 8-31　程序开始运行的界面

图 8-32　下载程序的界面

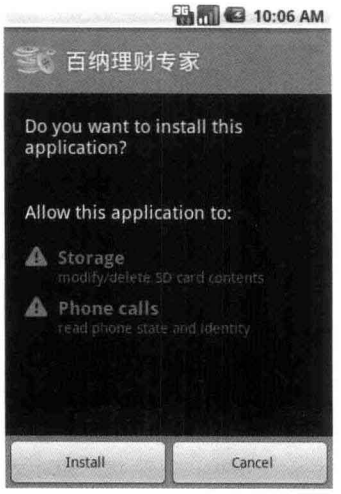


图 8-33　下载完成的界面

在解压完 Tomcat 压缩文件后，如果读者不清楚 Tomcat 服务器的配置或者使用方法，可以查阅有关 Tomcat 服务器的详细资料。

【代码解析】

在本部分会详细介绍该软件的实现过程。首先介绍的是该软件中 main.xml 文件的设置，代码如下。

代码位置：见随书光盘中源代码/第 8 章/Sample8_14/ res/layout 目录下的 main.xml。

```
<?xml version="1.0" encoding="utf-8"?>          <!--版本号和编码方式-->
<LinearLayout
  android:id="@+id/widget27"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
>                                               <!--LinearLayout-->
  <TextView
    android:id="@+id/TextView01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="请输入远程安装程序的 URL"
  >                                             <!--TextView 控件设置-->-->
  </TextView>
  <EditText
    android:id="@+id/EditText01"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="http://192.168.9.103:8080/apkname/LcSoftware.apk"
    android:textSize="18sp"
  >                                             <!--EditText 控件设置-->
  </EditText>
  <Button
    android:id="@+id/Button01"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
```

```
    android:text="开始安装"
  >                                                    <!-- Button 的属性-->
  </Button>
</LinearLayout>
```

提示： 上述 main.xml 的代码主要是搭建程序开始运行的主界面。首先在 LinearLayout 中设置摆放方式为竖直摆放，然后添加 TextView、EditText、Button 控件，并设置其具体的属性。

上面已经介绍完本程序主界面的搭建。接下来将要介绍的是下载网络上的 APK 安装包，并在下载完后打开 application installer 安装软件的框架，代码如下。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_14/src/com/bn/ex8n 目录下的 Sample8_14_Activity。

```
package com.bn.ex8n;                                          //声明包
……//该处省略了导入相关包的代码，读者可自行查阅随书光盘中源代码
import android.widget.TextView;                               //导入相关包
public class Sample8_14_Activity extends Activity{ //继承 Activity 需要重写的方法
      ....../// 该处省略了部分私有成员变量代码，读者可自行查阅随书光盘中源代码
      @Override
      public void onCreate(Bundle savedInstanceState){
                                                    //继承 Activity 类要重写的方法
            super.onCreate(savedInstanceState);               //实现父类的方法
            setContentView(R.layout.main);       //显示界面为 main 中设置的界面
            mTextView = (TextView)findViewById(R.id.TextView01);
                                                    //获取控件的 Id
            mButton = (Button)findViewById(R.id.Button01);    //获取控件的 Id
            mEditText =(EditText)findViewById(R.id.EditText01);
                                                    //获取控件的 Id
            mButton.setOnClickListener(
                  new Button.OnClickListener(){               //为控件添加监听
                      public void onClick(View v) {           //当单击控件时
                         mTextView.setText("下载中...");  //显示下载中
                         strURL = meText.getText().toString();  //获取 URL 地址
                         fileEx = sUrl.substring(sUrl.lastIndexOf(".")+1,
                            strURL.length()).toLowerCase();  //取得 Apk 的路径
                         fileNa = sUrl.substring(sUrl.lastIndexOf("/")+1,
                            strURL.lastIndexOf(".")); //取得欲安装程序之文件名称
                         getFile(strURL);                  //调用 getFile 方法
             } } );
            mEditText.setOnClickListener(                     //添加监听事件
                  new EditText.OnClickListener(){             //匿名内部类
                      @Override
                      public void onClick(View arg0){     //重写的方法
                        mEditText.setText("");          //清空 EditText 中的内容
                        mTextView.setText("请输入 URL 地址");
                                                    //提示用户输入 url 地址
             } } );
      }
      private void getFile(final String strPath){    //getFile 方法的声明
            try{
                  if (strPath.equals(currentFilePath)){ //路径为 currentFilePath
                        getDataSource (strPath);      //调用 getSource 方法
                  }
                  currentFilePath = strPath;          //不满足条件，获取路径
```

```
38                    Runnable r = new Runnable(){              //开启新的线程
39                          public void run(){                  //重写的方法
40                                try{
41                                   getDataSource (strPath); //调用 getSource 方法
42                                }catch (Exception e){          //捕获异常
43                                  e.printStackTrace();         //打印堆栈信息
44                           } } };
45                          new Thread(r).start();               //开始线程
46                    }catch(Exception e){                       //捕获异常
47                         e.printStackTrace();                  //打印堆栈信息
48                    }
49              }
50              private void getDataSource (String strPath) throws Exception{ //声明方法
51                    ......//该处省略了实现该方法的具体代码，将在后面给出
52              }
53              private void openFile(File f){                   //声明 openFile 方法
54                    Intent intent = new Intent();              //创建 Intent 对象
55                    intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);  //添加数据
56                    intent.setAction(android.content.Intent.ACTION_VIEW);
57                    String type = getMIMEType(f);   //调用 getMIMEType()来取得 Type
58                    intent.setDataAndType(Uri.fromFile(f),type);
                                                     //设定 intent 的 file 与 Type
59                    startActivity(intent);           //调用 startActivity 方法
60              }
61              private String getEType(File f){       //声明 getMIMEType 方法
62                    String type="";                  //空字符串
63                    String fName=f.getName();        //得到名称
64                    String end=fName.substring(fName.lastIndexOf(".")+1,fName.
length()).toLowerCase();
65                    if(end.equals("apk")){
66                      type = "application/vnd.android.package-archive"; //为 type 赋值
67                    }else{
68                      type += "/*";                  //为 type 赋值
69                    }
70                    return type;                     //返回 type 的值
71              }
72              private void deletedFile(String strFileName){      //删除软件包的方法
73                    File myFile = new File(strFileName);         //获取文件名称
74                    if(myFile.exists()){
75                      myFile.delete();                           //如果存在，删除文件
76                    }
77              }
78              @Override
79              protected void onPause(){                          //重写的 onPause 方法
80                    mTextView = (TextView)findViewById(R.id.TextView01); //创建对象
81                    mTextView.setText("下载成功");                 //下载成功
82                    super.onPause();                           //实现父类的方法
83              }
84              @Override
85              protected void onResume() {                        //重写的 onResume 方法
86                    deletedFile(currentTempFilePath);          //删除临时文件
87                    super.onResume();                          //实现父类的方法
88        } }
```

其中：

- 第 14～23 行表示为“开始安装”按钮添加监听器，首先改变 TextView 的显示为“下载中……”，获得 APK 文件路径，同时调用自己开发的 getFile 方法。

- 第 24～31 行表示单击 EditText 文本框，清空文本框的内容，同时提醒用户输入 URL 地址。
- 第 32～49 行表示创建自定义的 getFile 方法，并判断得到的路径是否是文件的路径，同时开启一个新的线程。
- 第 53～60 行表示创建自定义的 openFile 方法，并通过 Intent 发送消息。
- 第 72～77 行表示创建自定义的 deletedFile 方法，其主要作用是在软件安装完成后调用该方法，删除该软件的安装包。
- 第 79～89 行表示继承 Activity 需要重写的 onPause 方法，其主要作用是程序由运行状态进入暂停状态需要调用的方法。
- 第 86～88 行表示继承 Activity 需要重写的 onResume 方法，其主要作用是程序进入运行状态需要调用的方法。

上面已经介绍了在网络上下载并安装程序的主控制类，接下来介绍的是下载网络上的 APK 安装包到本地的 SD 卡中的代码，将下列代码插入到上面框架的第 52 行。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_14/src/com/bn/ex8n 目录下的 Sample8_14_Activity。

```
1   if (URLUtil.isNetworkUrl(strPath)){
2       URL myURL = new URL(strPath);                          //取得 URL
3       URLConnection conn = myURL.openConnection();           //建立联机
4       conn.connect();                                        //获得链接
5       InputStream is = conn.getInputStream();              //InputStream 下载文件
6       if (is == null){                                       //判断 is 是否为空
7           throw new RuntimeException("stream is null");   //如果为 null,抛出异常
8       }
9       File myTempFile = File.createTempFile(apkName, "."+apkPath);
                                                               //建立临时文件
10      currentTempFilePath = myTempFile.getAbsolutePath(); //取得临时存盘文件路径
11      FileOutputStream fos = new FileOutputStream(myTempFile);//将文件写入临时盘
12      byte buf[] = new byte[128];                            //创建 byte 数组
13      do{                                                    //do-While 循环
14          int numread = is.read(buf);
15          if (numread <= 0){                              //判断 numread 是否小于零
16              break;                                         //退出
17          }
18          fos.write(buf, 0, numread);                        //将文件写入
19      }while (true);
20      openFile(myTempFile);                                  //打开文件进行安装
21      try{
22          is.close();                                        //关闭输入流
23      }catch (Exception ex){                                 //捕获异常
24          ex.printStackTrace();                              //打印堆栈信息
25  } } else{
26      mtView.setText("URL 错误");                            //显示输入错误
27  }
```

提示：上面的代码主要是获得 URL 地址，同时通过 URL 地址建立联机，获得链接，并通过文件 I/O 操作将得到的数据存入 SD 卡中。在上面的操作中需要得到对 SD 卡进行读/写的操作，需要在 AndroidManifest.xml 中获得该权限。

实例 15　手机下载看 3gp 影片

下面向读者介绍的是通过网络连接下载 3gp 影片并播放，运行本程序进入主界面，要求用户输入 3gp 网址，然后单击“播放”按钮，播放下载的 3gp 软件。

【实例描述】

本节中主要介绍如何通过网络下载 3gp 影片，用户输入需要下载的 3gp 影片网址，单击“播放”按钮，调用按钮监听方法，通过网络下载 3gp 影片然后开始播放此影片。

本实例的运行效果如图 8-34、图 8-35、图 8-36 所示。

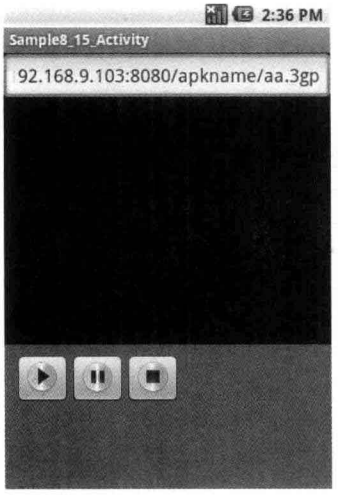


图 8-34　程序运行开始的界面

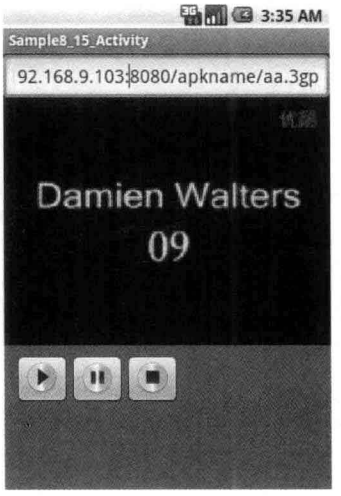


图 8-35　开始播放时的界面

图 8-36　影片播放中的界面

提示：在本程序的运行效果图 8-34、图 8-35、图 8-36 中，图 8-34 表示程序运行开始时的界面，图 8-35 表示开始播放影片时的界面，图 8-37 表示影片播放中的界面。

【实现过程】

在本实例中 EditText 输入网址，SurfaceView 中播放下载的 3gp 影片，单击“播放”按钮下载影片资源到手机 Sdcard 中，然后获取该资源播放，可以通过单击“暂停”按钮，暂停播放此影片，单击“停止”按钮，停止播放该影片。

搭建服务器：由于本程序需要连接网络下载 3GP 电影，因此读者需要自己搭建 Tomcat 服务器。本节为读者提供 Tomcat 压缩文件，其位于随书光盘中第 8 章的 Sample8_15_Tomcat 文件夹。解压该 Tomcat 压缩文件，配置 Tomcat 的环境。本程序用到的 3gp 资源位于压缩文件的 webapps 目录下的 apkname 文件夹中。

在解压完 Tomcat 压缩文件后，如果读者不清楚 Tomcat 服务器的配置或者使用方法，可以查阅有关 Tomcat 服务器的详细资料。

【代码解析】

下面向读者详细介绍本软件的开发过程。

首先必须在 AndroidManifest.xml 中添加 Internet 网络权限。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_15/AndroidManifest.xml

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.INTERNET"> <!--设置权限-->
```

接下来向读者介绍本软件主界面的开发。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_15/res/layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>          <!--版本号 编码方式-->
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
>
  <EditText
    android:id="@+id/editText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text=""
    android:singleLine="True" />                <!--EditText 的属性-->
  <SurfaceView
    android:id="@+id/surfaceView"
    android:visibility="visible"
    android:layout_width="320px"
    android:layout_height="240px">
  </SurfaceView>
  <LinearLayout
    android:orientation="horizontal"
    android:background="@drawable/lightblue"
    android:layout_height="fill_parent"
    android:layout_width="fill_parent"
    android:padding="10dip"
  >                                             <!--LinearLayout 的属性-->
  <ImageButton android:id="@+id/play"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:src="@drawable/play"
  />                                            <!--ImageButton 的属性-->
  <ImageButton android:id="@+id/pause"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:src="@drawable/pause"
  />                                            <!-- ImageButton 的属性-->
  <ImageButton android:id="@+id/stop"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:src="@drawable/stop"
  />                                            <!-- ImageButton 的属性-->
  </LinearLayout>
</LinearLayout>
```

提示：此布局文件为本软件的主界面 Xml，Xml 中主要有 ImageButton 图片按钮。SurfaceView 为播放 3gp 影片的控件，EditText 要求用户输入影片的网址。

然后向读者讲解该软件 Sample8_15_Activity 类的开发过程，代码如下所示。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_15/src/com.bn.ex8o /Sample8_15_Activity.java

```
package com.bn.ex8o;                                    //包名的声明
```

```
import android.app.Activity;                                    //相关类的引入
......//此处省略了部分类的引入代码，请读者自行查看随书光盘的源代码
import android.os.Bundle;                                       //相关类的引入
public class Sample8_15_Activity extends Activity
implements SurfaceHolder.Callback{                              //创建类
    public void onCreate(Bundle savedInstanceState){            //重写 onCreate 方法
        super.onCreate(savedInstanceState);                     //调用父类
        setContentView(R.layout.main);                          //跳转界面
        videoURL ="http://192.168.9.103:8080/apkname/aa.3gp";    //URL
        editText = (EditText)findViewById(R.id.editText);    //EditText 引入
        editText.setText(videoURL);
        aryFileDownloaded = new ArrayList<String>(); //初始化临时盘的路径数组
        surfaceView = (SurfaceView) findViewById(R.id.surfaceView);
                                                                //绑定 SurfaceView
        getWindow().setFormat(PixelFormat.TRANSPARENT); //设定 PixnelFormat
        surfaceHolder = surfaceView.getHolder();                //设定 SurfaceHolder
        surfaceHolder.addCallback(this);
        surfaceHolder.setFixedSize(160, 128);       //由于原有的影片为固定比例
        surfaceHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
        play = (ImageButton) findViewById(R.id.play);    //得到播放图片的引用
        pause = (ImageButton) findViewById(R.id.pause); //得到暂停图片的引用
        stop = (ImageButton) findViewById(R.id.stop);    //得到停止图片的引用
        play.setOnClickListener(                                //为播放按钮添加监听器
            new ImageButton.OnClickListener(){                  //匿名内部类
                public void onClick(View view){                 //重写的方法
                    if(checkSDCard()){
                        videoURL = editText.getText().toString(); //得到 URL
                        playVideo(videoURL);                    //调用方法
        } } } );
        pause.setOnClickListener(                               //为暂停按钮添加监听器
            new ImageButton.OnClickListener(){                  //匿名内部类
                public void onClick(View view){                 //重写的 onClick 方法
                    if(checkSDCard()) {
                        if (mediaPlayer != null) {              //判断播放器是否为空
                            if(bIsReleased == false){
                                if(bIsPaused==false){
                                                    //判断 blsPaused 是否为 false
                                  mediaPlayer.pause();    //影片暂停
                                  bIsPaused = true; //改变 blsPaused 为 true
                                }
                                mediaPlayer.start();        //影片播放
                                bIsPaused=false;  //改变blsPaused为false
        } } } } } );
        stop.setOnClickListener(                                //为停止按钮添加监听器
            new ImageButton.OnClickListener(){                  //结束按钮的实现
                public void onClick(View view){                 //重写的 onClick 方法
                    if(checkSDCard()){
                    try{                                        //捕获异常
                        if (mediaPlayer != null){               //判断播放器是否为空
                            if(bIsReleased==false){
                                                    //判断 blsPaused 是否为 false
                                mediaPlayer.seekTo(0);
                                mediaPlayer.pause();  //调用 pause 方法
                        } }
                    }catch(Exception e){                        //捕获异常
                        e.printStackTrace();                    //打印堆栈信息
                    }
```

```
56        } } } ); }
57        /*此处省略了下载影片并播放的方法，将在下面给出*/
58       /*此处省略了通过网络获得影片资源的方法，将在下面给出*/
59        /*此处省略了部分方法，将在下面给出*/
60    }
```

提示：以上代码为主控制类代码，用户单击图片按钮触动监听事件，执行相应的操作。

接下来介绍的是下载 3gp 影片并播放影片的方法，将此代码插入 Sample8_15_Activity 第 57 行。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_15/src/com.bn.ex8o /Sample8_15_Activity.java

```
1    private void playVideo(final String path){              //下载影片并播放
2        try{
3            if (path.equals(currentFilePath)&&
4              mediaPlayer != null){                         //如果 path 为现有播放的链接
5                mediaPlayer.start();                        //调用 start 方法
6                return;                                     //返回
7            }else if(mediaPlayer != null){                  //判断播放器是否为空
8                mediaPlayer.stop();                         //停止播放
9            }
10           currentFilePath = path;
11           mediaPlayer = new MediaPlayer();          //重新建构 MediaPlayer 对象
12           mediaPlayer.setAudioStreamType(2);              //设定播放音量 */
13           mediaPlayer.setDisplay(surfaceHolder);  //设定显示于 SurfaceHolder */
14           mediaPlayer.setOnErrorListener(                 //为播放器添加监听器
15               new MediaPlayer.OnErrorListener(){          //设置监听
16                   public boolean onError(MediaPlayer mp, int what, int extra){
17                       return false;                       //返回 false
18           } } );
19           Runnable r = new Runnable(){                    //开启线程
20               public void run() {                         //重写的 run()方法
21                   try{
22                       setDataSource(path);  //在线程运行中，调用自定义函数抓下文件
23                       mediaPlayer.prepare();              //下载完后才会调用 prepare
24                       mediaPlayer.start();                //调用 start 方法
25                       bIsReleased = false;                //blaReleased 为 false
26                   }catch (Exception e){                   //捕获异常
27           } } };
28           new Thread(r).start();                          //线程开始
29       }catch(Exception e) {
30           if (mediaPlayer != null) {
31               mediaPlayer.stop();                         //影片停止
32               mediaPlayer.release();                      //调用 release 方法
33   } } }
```

提示：此方法获取影片的网络地址和下载 3gp 影片，并调用 Sdcrad 中的资源文件进行影片播放。

然后介绍通过网络地址下载 3gp 资源的方法，将此代码放入 Sample8_15_Activity 第 58 行。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_15/src/com.bn.ex8o /Sample8_15_Activity.java

```
1    private void setDataSource(String strPath) throws Exception{
                                                    //自定义 setDataSource，由线程启动
```

```
2        if (!URLUtil.isNetworkUrl(strPath)){
3            mediaPlayer.setDataSource(strPath);     //获取资源
4        }else{
5            if(bIsReleased == false){               //判断 blaReleased 是否为空
6                URL myURL = new URL(strPath);       //创建 Url
7                URLConnection conn = myURL.openConnection();    //获取连接
8                conn.connect();                                 //打开连接
9                InputStream is = conn.getInputStream();
10               if (is == null) {                              //判断 is 是否为空
11                   throw new RuntimeException("stream is null");   //抛出异常
12               }
13               File    myFileTemp    =    File.createTempFile("hippoplayertmp",
"."+getFileExtension(strPath));
14               currentTempFilePath = myFileTemp.getAbsolutePath();
                                                    //取出影片资源路径
15               if(currentTempFilePath!=""){   //currentTempFilePath 长不为零
16                   aryFileDownloaded.add(currentTempFilePath);
                                                //aryFileDownloaded 中添加数据
17               }
18               FileOutputStream fos = new FileOutputStream(myFileTemp);
19               byte buf[] = new byte[128];             //声明数组
20               do{
21                   int numread = is.read(buf);         //读取数据
22                   if (numread <= 0) {                 //numread 小于等于零
23                       break;                          //退出
24                   }
25                   fos.write(buf, 0, numread);         //写入数据
26               }while (true);                          //do-while 循环
27               mediaPlayer.setDataSource(currentTempFilePath); //获取资源
28               try{
29                   is.close();                         //关闭输入流
30               }
31               catch (Exception ex){                   //捕获异常
32                   ex.printStackTrace();               //打印信息
33   } } } }
```

提示：此方法连接网络获取资源，然后下载资源到手机的 Sdcard 中。

接下来介绍的是创建影片的格式为 3gp 格式，且删除 Scard 的临时文件将此代码放入 Sample8_15_Activity 第 59 行。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_15/src/com.bn.ex8o /Sample8_15_Activity.java

```
1    private String getFileExtension(String strFileName)  {
2        File myFile = new File(strFileName);        //得到 File 的对象
3        String strFileExtension=myFile.getName();   //得到文件名称
4        strFileExtension=(strFileExtension.substring(strFileExtension.
lastIndexOf(".")+1)).toLowerCase();
5        if(strFileExtension==""){                   //strFileException 长度为零
6            strFileExtension = "dat";               //的取得扩展名，预设为.dat
7        }
8        return strFileExtension;
9    }
10   private void delFile(String strFileName) {
11       File myFile = new File(strFileName);
12       if(myFile.exists()){
13       myFile.delete();
14   } }
```

```
15  private boolean checkSDCard(){                              //判断记忆卡是否存在
16      if(android.os.Environment.getExternalStorageState().equals
17      (android.os.Environment.MEDIA_MOUNTED)){
18      return true;
19      }
20      else{
21      return false;
22  } }
23  protected void onPause(){
24      for(int i=0;i<aryFileDownloaded.size();i++){     //删除所有下载的临时文件
25          delFile(aryFileDownloaded.get(i).toString());
26      }
27      super.onPause();
28  }
```

提示：以上两个方法依次为转化 3gp 格式和删除 Sdcard 临时文件的方法。

实例 16　常用网站登录界面的制作

下面向读者介绍访问网站 LoginApI 的软件，运行本软件进入登录界面，要求用户输入登录的用户名，登录的密码，然后单击“登录”按钮，跳转登录成功界面。

【实例描述】

本节主要介绍如何访问网站的 LoginAPI，需要用户输入用户名，密码，登录网站，用户名和用户密码正确则进入登录成功界面，用户名或用户密码错误则进入登录失败界面。

本实例的运行效果如图 8-37、图 8-38、图 8-39 所示。

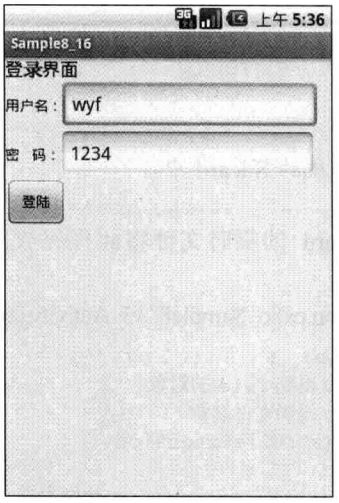


图 8-37　登录界面

图 8-38　登录成功界面

图 8-39　登录失败界面

提示：本软件首先进入如图 8-37 所示的登录界面，提示用户输入用户名和用户密码，单击“登录”进入网站，如果用户名和用户密码正确，则进入登录成功界面，如图 8-38 所示，如果用户名或用户密码错误进入登录失败界面，如图 8-39 所示。

【实现过程】

本节中主要介绍如何通过用户名及密码连接网站，通过创建 URL 加载网站地址和用户名，用户密码，登录网站，登录成功进入登录成功界面，并弹出 Toast 提示登录成功，登录失败进入登录失败界面，同时弹出 Toast 提示用户登录失败。

搭建服务器：本程序需要读者自己搭建 Tomcat 服务器。本节为读者提供 Tomcat 压缩文件，其位于随书光盘中第 8 章的 Sample8_16_Tomcat 文件夹中。解压该 Tomcat 压缩文件，配置 Tomcat 的环境。本程序使用判断是否登录成功的 JSP 为压缩文件 webapps 目录下的 test 文件夹中的 login.jsp。

在解压完 Tomcat 压缩文件后，如果读者不清楚 Tomcat 服务器的配置或者使用方法，可以查阅有关 Tomcat 服务器的详细资料。

【代码解析】

下面向读者详细介绍本软件的开发过程。首先是对本程序的服务器端的开发，代码如下。

代码位置：见随书光盘中源代码/第 8 章/Smaple8_16_Tomcat/webapps/test/login.jsp。

```
<%@ page contentType="text/html;charset=gbk"%>        <!--编码的类型和语言-->
<%
   String uid=request.getParameter("uid");
   String pwd=request.getParameter("pwd");
   if(uid.equals("wyf")&&pwd.equals("1234")){
       out.print("ok");
   }else{
       out.print("fail");
   }
%>
```

提示：该服务器端的主要作用是判断名称与密码是否符合。

上述代码介绍了本程序的服务器端的开发，接下来介绍的是在开发前需要在 AndroidManifest.xml 中添加 Internet 网络权限，代码如下。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_16/AndroidManifest.xml

```
<uses-permission android:name="android.permission.INTERNET"/>
                                                    <!--Internet 权限-->
```

接下来向读者介绍本软件主界面的开发。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_16/res/layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>          <!--版本号 编码方式-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"          android:layout_width="fill_parent"
android:layout_height="fill_parent"        android:background="#EEEEEE"
>                                                   <!-- LinearLayout 的属性-->
<TextView
android:text="登录界面"   android:id="@+id/TextView03"
android:textSize="18sp"   android:textColor="#000000"
android:layout_width="wrap_content"
android:layout_height="wrap_content">           <!--TextView 控件设置-->
</TextView>
android:id="@+id/LinearLayout01"    android:orientation="horizontal"
android:layout_width="wrap_content"  android:layout_height="wrap_content">
<TextView
android:text="用户名: "          android:id="@+id/TextView01"
```

```
android:textColor="#000000"    android:layout_width="wrap_content"
android:layout_height="wrap_content">                <!--TextView 控件设置-->
</TextView>
<EditText
android:text="wyf"          android:id="@+id/EditText01"
android:layout_width="250sp"  android:layout_height="wrap_content">
</LinearLayout>                                      <!--LinearLayout 设置-->
<LinearLayout
android:id="@+id/LinearLayout02"
android:orientation="horizontal"
android:layout_width="wrap_content"
<TextView
android:text="密  码: "        android:textColor="#000000"
android:id="@+id/TextView02"    android:layout_width="wrap_content"
android:layout_height="wrap_content">                <!--TextView 控件设置-->
<EditText
android:text="1234"          android:id="@+id/EditText02"
android:layout_width="250sp"  android:layout_height="wrap_content">
</EditText>                                          <!--EditText 控件设置-->
</LinearLayout>                                      <!--LinearLayout 设置-->
<Button
android:text="登录"            android:id="@+id/Button01"
android:textColor="#000000"     android:layout_width="60sp"
android:layout_height="wrap_content">
</Button>                                            <!-- Button 的属性-->
</LinearLayout>
```

提示：以上代码为登录界面的 Xml 文件，其中 TextView 显示本界面为登录界面，然后有 EditText 控件，要求用户输入用户名，用户密码，Button 为登录网站的按钮。

然后介绍的是登录成功界面的 Xml 布局文件。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_16/res/layout/ok.xml

```
<?xml version="1.0" encoding="utf-8"?>          <!--版本号 编码方式-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:background="#EEEEEE"
>                                               <!--LinearLayout 的属性-->
<TextView
android:text="登录成功!!! "
android:id="@+id/TextView01"
android:textSize="20sp"
android:textColor="#000000"
android:layout_width="wrap_content"
android:layout_height="wrap_content">           <!--TextView 控件设置-->
</TextView>
<ImageView
android:id="@+id/ImageView01"
android:src="@drawable/ok"
android:layout_width="wrap_content"
android:layout_height="wrap_content">           <!--控件大小-->
</ImageView>
</LinearLayout>
```

提示：此 Xml 布局文件为登录成功界面，TextView 提示用户登录成功，ImageView 显示图片。

最后介绍的是网站登录失败的 Xml 布局文件。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_16/res/layout/fail.xml

```
<?xml version="1.0" encoding="utf-8"?>          <!--版本号，编码方式-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:background="#EEEEEE"
>                                               <!--LinearLayout 设置-->
<TextView
android:text="登陆失败!!! "
android:id="@+id/TextView01"
android:textSize="20sp"
android:textColor="#000000"
android:layout_width="wrap_content"
android:layout_height="wrap_content">           <!--控件大小-->
</TextView>                                     <!--TextView 控件设置-->
<ImageView
android:id="@+id/ImageView01"
android:src="@drawable/fail"
android:layout_width="wrap_content"
android:layout_height="wrap_content">           <!--控件大小-->
</ImageView>                                    <!-- ImageView 的属性-->
</LinearLayout>
```

提示：此 Xml 布局文件为登录失败界面，其中 TextView 显示登录失败，ImageView 显示图片。

然后向读者讲解该软件 Sample8_16_Activity 类的开发过程，代码如下所示。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_16/src/com.bn.ex8p /Sample8_16_Activity.java

```
package com.bn.ex8p;                                    //包的声明
import java.net.URLConnection;                          //相关类的引入
......//此处省略了部分类的引入代码，请读者自行查看随书光盘源代码
import android.app.Activity;                            //相关类的引入
public class Sample8_16_Activity extends Activity {
    byte[] result = null;                               //定义变量
    String uid;                                         //定义用户名
    String pwd;                                         //定义用户密码
    String flag;
    public void onCreate(Bundle savedInstanceState) {   //重写的 onCreate 方法
      super.onCreate(savedInstanceState);               //调用父类
      setContentView(R.layout.main);                    //设置布局
      Button b=(Button)findViewById(R.id.Button01);     //Button 按钮
      b.setOnClickListener (                            //添加 button 监听器
        new OnClickListener(){                          //匿名内部类
          public void onClick(View v) {                 //重写的 onClic 方法
            try {
                EditText e1=(EditText)findViewById(R.id.EditText01);
                                                        //控件声明
                EditText e2=(EditText)findViewById(R.id.EditText02);
                                                        //得到 EditText02 的引用
                uid=e1.getText().toString().trim();     //得到用户名
                pwd=e2.getText().toString().trim();     //得到密码
                URL url=new URL(Constant.PRE_URL+"uid="+uid+"&pwd="+pwd);
                URLConnection uc=url.openConnection(); //url 的连接
```

```
                    InputStream in=uc.getInputStream();          //得到输入流
                    int ch=0;                                    //局部变量 ch 为零
                    ByteArrayOutputStream baos = new ByteArrayOutputStream();
                    while((ch=in.read())!=-1)  {
                        baos.write(ch);                          //写入数据
                    }
                    result=baos.toByteArray();
                    baos.close();                                //关闭输出流
                    in.close();                                  //关闭输入流
                }catch(Exception e) {                            //捕获异常
                    e.printStackTrace();                         //打印堆栈信息
                }
                flag=new String(result).trim();          //判断是否登录成功标志位
                if(flag.equals("ok")) {                  //判断是否登录成功
                    Toast.makeText(Sample8_16_Activity.this, "恭喜您, "+
                    "登录成功! ", 1000).show();                   //弹出 Toast 提示
                    gotoOk();                            //切换到登录成功界面
                }else if(flag.equals("fail")) {
                    Toast.makeText(Sample8_16_Activity.this, "对不起, "+
                    "登录失败! ",1000).show();                    //弹出 Toast 提示
                    gotoFail();                          //切换到登录失败界面
                }else {
                    Toast.makeText(Sample8_16_Activity.this, "对不起 sdfdsfds, "+
                            +"登录失败! ", 1000).show();          //弹出 Toast 提示
                }
} } ); }
    public void gotoOk() {                               //切换到登录成功界面的方法
        setContentView(R.layout.ok);                             //跳转界面
    }
    public void gotoFail() {                             //切换到登录失败界面的方法
        setContentView(R.layout.fail);
} }
```

提示：以上代码为主控制类的实现，首先 EditText 获取用户名和用户密码，单击“登录”按钮，调用按钮监听事件，创建 url 连接网站同时验证用户名、密码是否正确，如果用户登录成功进入登录成功界面，如果登录失败进入登录失败界面，弹出 Toast 提示信息。

最后介绍的是本软件用到的常量，常量类 Constant 代码如下所示。

代码位置：见本书随书光盘中源代码/第 8 章/Sample8_2/src/com.bn.ex8p /Constant.java

```
1   package com.bn.ex8p;
2   public class Constant {
3       public static final String PRE_URL="http://192.168.9.109:8080/test/login.
jsp?";  //IP 地址
4   }
```

提示：此类为本软件的常量类，提供本软件连接网络的网络地址。

小结

本章主要介绍了手机网络的应用，首先以网络连接测试软件为例进行介绍如何进行网络的

连接，然后以浏览器的开发为例，一种是 Uri 方式，另一种是 WebView 显示方式，详细讲述了如何在 Android 手机中嵌入浏览器。

再然后详细介绍了以 Tomcat 为服务器，开发安装 Android 程序的实例，读者通过本章的学习可以熟练地编写功能各异的软件小程序。

第 9 章　手机的 Google 服务功能

打开电脑搜索一些需要的资料，经常使用的工具便是 Google 搜索。Google 公司不仅仅提供了搜索功能，还提供了诸如 Google 地图、图表制作等服务功能，在本章将对 Android 与 Google 的结合使用进行详细介绍。

实例 1　手机客户端 Google 账号登录

Google 的很多服务功能都需要注册一个 Google 账号才能够使用，在本节将主要介绍如何获取 Token，并使用该 Token 浏览网页。

【实例描述】

本软件模拟 Google 账号登录，当运行该软件时，在主界面会提醒用户填写用户名和密码，若未填写用户名或未填写密码，单击登录按钮时，系统将提醒用户用户名和密码必须填写。若填写的用户名和密码不正确，系统提示用户“很遗憾，未能获取 Token，请检查账号和密码是否正确！”。若填写正确，则将获取的 Token 显示在主界面中，并提示用户获取 Token 成功。获取成功的同时将自动访问 http://www.google.com/ig?hl=zh-CN&refresh=1，并将该网址信息写入临时文件中。

本实例的运行效果如图 9-1 所示。

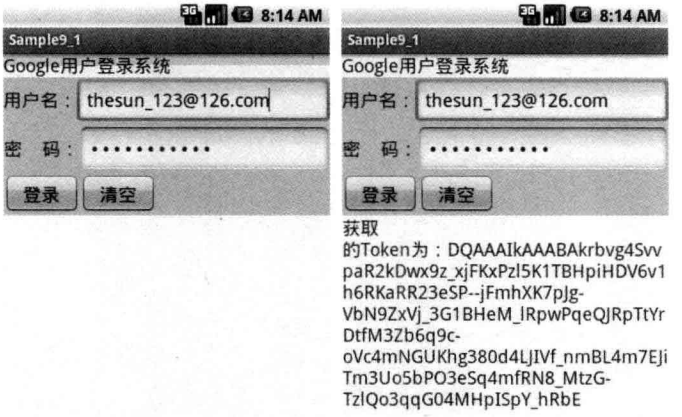


图 9-1　Google 用户登录系统效果图

提示：在图 9-1 中依次表示的是软件主界面效果，之后在主界面单击登录按钮后将获取的 Token 显示在主界面中，表示成功获取 Token。

【实现过程】

在本实例中主要学习 Google 所提供的 ClientLogin 机制，获取 Google Login 验证服务所发出的 Token，并使用该 Token 访问网站界面。在主界面中，设置 Button 按钮的监听器，当单击 Button 按钮时，获取所填入的用户名和密码。

使用 HttpHost 连接到 https://www.google.com/accounts/ClientLogin，同时将用户名和密码以 Name Value Pair 带入，通过自定义的方法获取 Google 认证的 Authentication Token。创建 Header 对象，模拟 Google 网络服务的 AuthSub 方法，首先自定义 Header 以及 HttpGet 方法带入 Token 访问网站，并将该网址临时写入 SD 卡中的 sdcard 文件夹下。

【代码解析】

经过上面的理论介绍，在本部分将对该其核心代码进行详细介绍，首先要介绍的是该实例中 AndroidManifest.xml 文件的实现，在该文件中主要完成的是权限的设置，该文件的主要代码如下所列。

代码位置：见随书光盘中源代码/第 9 章/Sample9_1/目录下的 AndroidManifest.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>                    <!--版本号和编码方式-->
2   <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3           package="com.bn.chap9.login"
4           android:versionCode="1"
5       android:versionName="1.0">                            <!--版本名称-->
6     <application android:icon="@drawable/icon" android:label="@string/app_name">
7       <activity android:name=".Sample9_1_Activity"
8               android:label="@string/app_name">     <!--显示的程序名称-->
9           <intent-filter>
10              <action android:name="android.intent.action.MAIN" />
                                                              <!--标识程序开始-->
11              <category android:name="android.intent.category.LAUNCHER" />
12          </intent-filter>
13      </activity>
14    </application>
15     <uses-permission android:name="android.permission.INTERNET"></uses-
permission>
16  </manifest>
```

提示：上述代码中第 15 行为上网权限的添加，若没有该权限，则该软件将无法正常运行。

权限设置完成后，下一步要做的就是主类 Sample9_1_Activity 的设计与实现，该类主要完成的是界面的设置以及功能的实现，在详细介绍该类之前，首先介绍一下该类的设计框架，代码如下所列。

代码位置：见随书光盘中源代码/第 9 章/Sample9_1/src/com/bn/chap9/login 目录下的 Sample9_1_Activity.java。

```
1   package com.bn.chap9.login;
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3   public class Sample9_1_Activity extends Activity {
4       EditText user;                                        //用户名
5       EditText password;                                    //密码
6       Button bOK;                                           //确定按钮
7       Button bClear;                                        //清空按钮
8       TextView tv;                                          //TextView 引用
```

```
9       DefaultHttpClient client;                          // DefaultHttpClient 引用
10      HttpPost httpPost;                                 //声明 HttpPost
11      HttpResponse response;
12    public void onCreate(Bundle savedInstanceState) {    //重写 onCreate 方法
13       super.onCreate(savedInstanceState);
14       setContentView(R.layout.main);                    //设置当前界面
15       client=new DefaultHttpClient();                   //创建对象
16       httpPost=new HttpPost("https://www.google.com/accounts/ClientLogin");
                                                           //创建连接
17       user=(EditText)this.findViewById(R.id.EditText01);    //用户名
18       password=(EditText)this.findViewById(R.id.EditText02);//密码
19       bOK=(Button)this.findViewById(R.id.Button01);         //确定按钮
20       bClear=(Button)this.findViewById(R.id.Button02);      //清空按钮
21       tv=(TextView)this.findViewById(R.id.TextView03);
22       bClear.setOnClickListener(                        //为清空按钮添加监听器
23          new OnClickListener(){
24                  public void onClick(View v) {
25                      user.setText("");                  //设置用户名显示为空
26                      password.setText("");              //设置密码显示为空
27     }});
28       bOK.setOnClickListener(                           //为确定按钮添加监听器
29      ……//该处省略了部分代码，将在下面详细介绍。
30      );
31    }
32    public String getToken(String name,String pwd){  //自定义方法
33      ……//该处省略了部分代码，将在下面详细介绍。
34    }
35    public String getAuth(InputStream is){            //自定义分离 Token 的方法
36      ……//该处省略了部分代码，将在下面详细介绍。
37    }}
```

其中：

- 第 4～11 行主要完成成员变量 EditText、Button、DefaultHttpClient、HttpPost 和 HttpResponse 引用的声明。
- 第 13～31 行为创建对象，并为清空按钮和确定按钮添加监听器，在清空按钮监听器中设置用户名和密码为空，确定按钮监听器会在下面进行详细介绍。
- 第 32～37 行为自定义获取 Token 和分离 Token 的方法。这两个方法将在下面进行详细介绍。

（1）下面要介绍的是 Sample9_1_Activity 类中确定按钮监听器的实现，将下列代码插入到 Sample9_1_Activity 框架的第 29 行。

代码位置：见随书光盘中源代码/第 9 章/Sample9_1/src/com/bn/chap9/login 目录下的 Sample9_1_Activity.java。

```
1   new OnClickListener(){                                    //创建对象
2       public void onClick(View v) {
3           String userName=user.getText().toString().trim();    //获取用户名
4           String pwd=password.getText().toString().trim();     //获取密码
5           if(userName.length()==0){                         //若没有填写用户名
6               Toast.makeText(                               //创建 Toast
7                   Sample9_1_Activity.this,
8                   "请输入用户名！",                          //Toast 提示信息
9                   Toast.LENGTH_SHORT).show();               //显示 Toast
10          }else if(pwd.length()==0){                        //若没有填写密码
11              Toast.makeText(                               //创建 Toast
```

```
                    Sample9_1_Activity.this,
                    "请输入密码！",                          //Toast 提示信息
                    Toast.LENGTH_SHORT).show();              //显示该 Toast
            }else{
                String result=getToken(userName,pwd);        //调用 getToken 方法
                if(result.length()!=0){                    //若获取的信息长度不为 0
                    Toast.makeText(                          //创建 Toast
                        Sample9_1_Activity.this,
                        "恭喜您，成功获取 Token！",
                        Toast.LENGTH_SHORT).show();          //显示该 Toast
                    tv.setText("获取的 Token 为："+result); //将获取的 Token 显示在界面
                    Header[] header=new BasicHeader[5];      //创建 Header 对象
                    header[0]=new BasicHeader("Content-type","application/x-
www-form-urlencoded");
                    header[1]=new BasicHeader("Authorzation","GoogleLogin auth=
\""+result+"\"");
                    header[2]=new BasicHeader("User-Agent","Java/1.5.0_06");
                    header[3]=new BasicHeader("Accept","text/html,image/
gif,image/jpeg,*;q=2,*/*;q=.2");
                    header[4]=new BasicHeader("Connection","keep-alive");
                    String url="http://www.google.com/ig?hl=zh-CN&refresh=1";
                    HttpGet get=new HttpGet(url);                //创建 HttpGet
                    for(int i=0;i<header.length;i++){
                        get.addHeader(header[i]);          //将信息放入 HttpGet 中
                    }try {
                        response=client.execute(get);            //取得服务应答
                        InputStream tempIs=response.getEntity().getContent();
                        BufferedReader read=new BufferedReader(new InputStream
Reader(tempIs));
                        StringBuffer sb=new StringBuffer();
                                                    //创建 StringBuffer 对象
                        String tempStr=null;
                        try{
                            while((tempStr=read.readLine())!=null){
                            sb.append(tempStr);              //组装字符串
                        }
                        }catch(Exception e){e.printStackTrace();    //打印异常
                        }finally{
                            try{
                                tempIs.close();              //关闭输入流
                            }catch(Exception e){e.printStackTrace();}//打印异常
                            }
                            tempStr=sb.toString().trim(); //转换为字符串
                        BufferedWriter bw=new BufferedWriter(new FileWriter
("/sdcard/user.txt"));
                        bw.write(tempStr,0,tempStr.length());  //写入文件
                        bw.flush();                              //刷新
                        } catch (ClientProtocolException e) {e.printStackTrace();
                        } catch (IOException e) {e.printStackTrace();}
                                                                 //打印异常
            }else{                                             //若获取字符串长度为 0
                Toast.makeText(                                //创建 Toast
                    Sample9_1_Activity.this,
                    "很遗憾，未能获取 Token，请检查账号和密码是否正确！",
                    Toast.LENGTH_SHORT).show();              //显示该 Toast
                    tv.setText("未能获取 Token！");          //设置提醒
    }}}}
```

其中：

- 第 1～15 行为简单的本地验证，获取 EditText 中的信息后，判断所获取的信息长度是否为 0，若不为 0 则进入下一步，若为 0，则提示用户填写用户名或密码。
- 第 16～33 行为通过本地验证后首先获取 Token，获取成功后将获取的 Token 信息设置在主界面中，然后创建 Header 对象，模拟 Google 网络服务的 AuthSub 方法，并将自定义的 Header 信息 HttpGet 带入 Token 取得网站信息。
- 第 34～55 行为获取服务应答，声明输入/输出流，将获取的信息重新组装为字符串写入临时文件中，读/写结束时关闭输入流。
- 第 56～61 行未成功获取 Token 数据，使用 Toast 提醒用户，并将未能获取 Token 数据显示在主界面中。

（2）下面介绍自定义的两个方法，其中一个为根据收到的用户名和密码获取 Token，另一个为将 Token 信息分离出来，将下面的 getToken 方法与 getAuth 方法分别插入到 Sample9_1_Activity 框架的第 33 行与第 36 行。

代码位置：见随书光盘中源代码/第 9 章/Sample9_1/src/com/bn/chap9/login 目录下的 Sample9_1_Activity.java。

```
1   public String getToken(String name,String pwd){
2       String result=null;
3       List<NameValuePair> list=new ArrayList<NameValuePair>();
                                                    //创建 NameValurPair 字符串
4       list.add(new BasicNameValuePair("Email",name));          //添加账号
5       list.add(new BasicNameValuePair("Passwd",pwd));          //添加密码
6       list.add(new BasicNameValuePair("source","clientstr"));
7       list.add(new BasicNameValuePair("service","reader"));
8       try{
9           httpPost.setEntity(                                  //创建连接
10                  new UrlEncodedFormEntity(
11                          list,
12                          HTTP.DEFAULT_CONTENT_CHARSET));
13          response=client.execute(httpPost);
14          if(response.getStatusLine().getStatusCode()!=200){  //若不等于 200
15              return "";                                       //返回空字符串
16          }
17          InputStream is=response.getEntity().getContent();
18          result=getAuth(is);                                  //获取信息
19      }catch(Exception e){e.printStackTrace();}                //打印异常
20      return result;                                           //返回结果
21    }
22    public String getAuth(InputStream is){                  //分离 Token 的方法
23      String result=null;
24      String line=null;                                        //创建对象
25      BufferedReader read=new BufferedReader(
26              new InputStreamReader(is));              //创建 BufferedReader 对象
27      try{
28          while((line=read.readLine())!=null){                 //读取信息
29              if(line.startsWith("Auth=")){
30                  result=line.substring(5);                    //截取字符串
31              }}
32      }catch(Exception e){e.printStackTrace();                 //打印异常
33      }finally{
34          try{
35              is.close();                                      //关闭输入流
36          }catch(Exception e){e.printStackTrace();}            //打印异常
```

```
37        }
38        return result;                                    //返回结果
39      }
```

其中：

- 第 1～21 行为自定义获取 Token 方法，在该方法中首先创建 NameValurPair 类型的 list，并将用户名，密码放入该 list 中，然后创建连接，并判断 response.getStatusLine().getStatusCode()获取的值是否为 200，若为 200 则表示连接成功，创建输入流，并调用分离 Token 的方法。
- 第 22～39 行为自定义分离 Token 的方法，在该方法中通过接收输入流，创建 BufferedReader 对象，读取获得的信息，将以“Auth=”开头的一行获取，随后从该字符串的第 5 位截取剩余字符串即为 Token 信息，最后返回截取的 Token 信息。

实例 2　使用手机进行 Google 搜索

Google 搜索是谷歌公司十分强大的服务功能，方便了用户搜索。在 Android 手机中同样可以进行 Google 搜索，在本节将介绍如何实现该项功能。

【实例描述】

本软件结构简单，在主界面包含有一个 EditText 和一个 Button 按钮，在 EditText 中输入要查询信息的关键字，单击查询后，在按钮的下方会将查询得到的结果显示出来，当单击其中的一条信息时，使用 Toast 提示多单击的信息。

本实例的运行效果如图 9-2 所示。

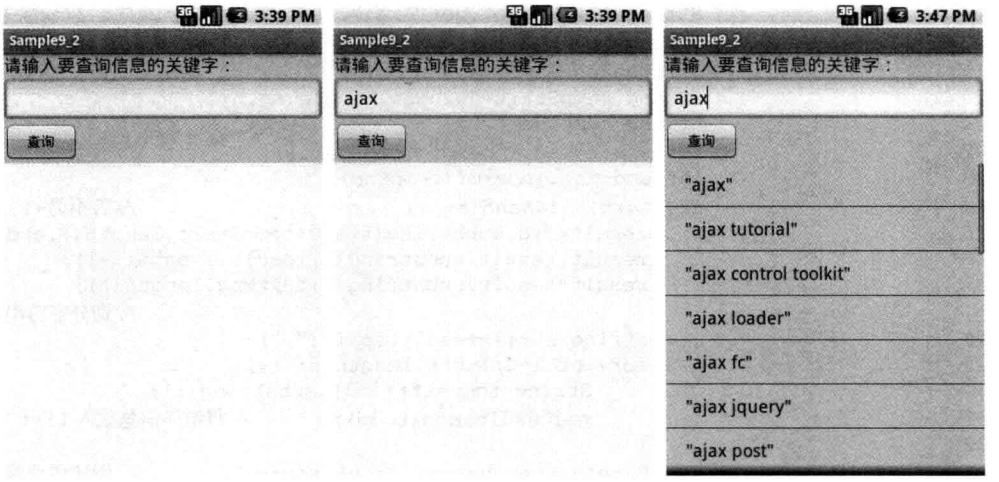


图 9-2　查询效果图

提示：在图 9-2 中依次为开始运行时的界面，填写要查询的信息和查询结果在 ListView 中的显示。

【实现过程】

在本实例中主要运用的是 Google Search Api，当单击查询按钮时，首先获取 EditText 中的

关键字，然后从获取的信息中取得所需的部分。将其放入 List 列表中，然后再将其分别放入 ListView 中。

在本实例中需要进行权限设置，其权限代码为：<uses-permission android:name="android.permission.INTERNET"></uses-permission>。

【代码解析】

经过上面的理论介绍，下面要介绍的是本软件的核心代码部分，首先介绍的是 Sample9_2_Activity 类的设计与实现，该类的代码如下所列。

代码位置：见随书光盘中源代码/第 9 章/Sample9_2/src/com/bn/chap9/search 目录下的 Sample9_2_Activity.java。

```
1   package com.bn.chap9.search;
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3   public class Sample9_2_Activity extends Activity {
4       Button search;                                          //查询按钮
5       EditText et;                                            //关键字
6       ListView lv;                                            //显示结果集
7       List<String> msgResult=new ArrayList<String>();         //存放结果信息
8      public void onCreate(Bundle savedInstanceState) {
9         super.onCreate(savedInstanceState);
10        setContentView(R.layout.main);                        //设置当前界面
11        search=(Button)this.findViewById(R.id.Button01);      //获取对象
12        et=(EditText)this.findViewById(R.id.EditText01);
13        lv=(ListView)this.findViewById(R.id.ListView01);
14        search.setOnClickListener(                            //添加监听器
15          new OnClickListener(){
16              public void onClick(View v) {
17                  String textMsg=et.getText().toString().trim(); //获取关键字
18                  String msg=search(textMsg);                 //调用 search 方法
19                  String result="";                           //声明字符串
20                  String tempStart="window.google.ac.h(";
21                  String tempEnd=")";
22                  int start=msg.indexOf(tempStart);           //获取位置索引
23                  int end=msg.indexOf(tempEnd);
24                  if(start!=-1&&end!=-1){                     //若不为-1
25                      result=msg.substring(start+tempStart.length(),end);
26                      result=result.substring(1,result.length()-1);
27                      result=result.substring(4+textMsg.length());
                                                                //切分字符串
28                      String str[]=result.split(",");
29                      for(int i=0;i<str.length/3;i++){
30                          String temp=str[i*3].substring(1);
31                          msgResult.add(temp);       //将结果信息放入 List 中
32                      }
33                      BaseAdapter ba=new BaseAdapter(){       //创建适配器
34                          public int getCount() {
35                              return msgResult.size();        //ListView 容量
36                          }
37                          public Object getItem(int position) {
                                                    //重写的 getItem 方法
38                              return null;        //返回空
39                          }
40                          public long getItemId(int position) {
                                                    //重写的 getItemId 方法
41                              return 0;           //返回 0
```

```
    42                                  }
    43                                  public View getView(int arg0, View arg1, ViewGroup
arg2) {
    44                                      LinearLayout ll=new LinearLayout(Sample9_2_
Activity.this);
    45                                      ll.setOrientation(LinearLayout.HORIZONTAL);
    //设置摆放顺序
    46                                      ll.setPadding(15, 5, 5, 5);
    47                                      TextView tv=new TextView(Sample9_2_Activity.
this);//初始化 TextView
    48                                      tv.setTextColor(Color.BLACK); //设置字体颜色
    49                                      tv.setPadding(5,5,5,5);
    50                                      tv.setText(msgResult.get(arg0));   //添加信息
    51                                      tv.setGravity(Gravity.LEFT);       //左对齐
    52                                      tv.setTextSize(18);                //字体大小
    53                                      ll.addView(tv);                 //添加 TextView
    54                                      return ll;
    55                                      }
    56                              };
    57                              lv.setAdapter(ba);                          //设置适配器
    58                              lv.setOnItemClickListener(          //设置选中菜单的监听器
    59                                      new OnItemClickListener(){
    60                                              public void onItemClick(AdapterView
<?> arg0, View arg1,
    61                                                      int arg2, long arg3) {
    62                                                  Toast.makeText(   //创建 Toast
    63                                                          Sample9_2_Activity.
this,
    64                                                          msgResult.get(arg2)+"",
    65                                                          Toast.LENGTH_SHORT).
show();
    66       }});}}});}
    67      public String search(String msg){
    68        ……//该处省略了部分代码，将在后面进行详细介绍。
    69      }}
```

其中：

- 第 1～13 行为创建该类的成员变量，并创建对象。
- 第 14～32 行主要完成的是为查询按钮添加监听器，获取填写的关键字，并调用自定义的 search 方法，将获取的查询信息重新组装，从中取得需要的信息，并将该结果放入 msgResult 列表中。
- 第 33～57 行为创建适配器，并为适配器添加监听器。在创建的适配器中首先获取列表的大小，决定 ListView 的容量，在重写的 getView 方法中，首先声明 LinearLayout，设置 LinearLayout 的属性，随后创建 TextView 对象，并将获取的结果信息添加到该 TextView 中，最后将 TextView 添加到 LinearLayout 中。
- 第 58～66 行为在设置的监听器中设置为单击其中一条信息时，使用 Toast 显示单击的信息内容。
- 第 67～69 行为自定义的 search 方法，在下面将会详细介绍。

接下来介绍自定义的 search 方法的设计与实现，在该方法中主要完成的是获取搜索信息的功能，将下列代码插入到上述代码的第 68 行。

代码位置：见随书光盘中源代码/第 9 章/Sample9_2/src/com/bn/chap9/search 目录下的 Sample9_2_Activity.java。

```
1   public String search(String msg){
```

```
2       String result="";                                    //创建新字符串
3       String uri="";
4       try{
5           uri="http://www.google.com/complete/search?hl=en&js=true&qu="+
6           URLEncoder.encode(msg,"utf-8");                  //对输入的关键字进行编码
7       }catch(Exception e){e.printStackTrace();}            //打印异常
8       URL url=null;
9       HttpURLConnection connection=null;                  //声明 HttpURLConnection 引用
10      InputStream is=null;
11      BufferedReader br=null;
12      try{
13          url=new URL(uri);                                //创建 URL 对象
14          connection=(HttpURLConnection)url.openConnection(); //打开连接
15          int code=connection.getResponseCode();
16          if(code==HttpURLConnection.HTTP_OK){             //如果连接成功
17              is=connection.getInputStream();              //获取输入流
18              br=new BufferedReader(new InputStreamReader(is));   //创建对象
19              StringBuffer sb=new StringBuffer();          //创建 StringBuffer 对象
20              String tempStr=null;
21              while((tempStr=br.readLine())!=null)         //读取信息
22              {
23                  sb.append(tempStr);                      //组装字符串
24              }
25              result=sb.toString().trim();                 //转换为字符串
26          }
27      }catch(Exception e){e.printStackTrace();             //打印异常
28      }finally{
29          try{
30              if(is!=null){
31                  is.close();                              //关闭输入流
32              }
33              if(connection!=null){
34                  connection.disconnect();                 //关闭连接
35              }
36          }catch(Exception e){e.printStackTrace();}        //打印异常
37      }
38      return result;                                       //返回结果
39    }
```

其中：

- 第 1～7 行为创建空字符串，设置 uri，并对其进行编码设置。
- 第 8～39 行为声明 URL 、HttpURLConnection、InputStream 和 BufferedReader 引用后，再一次创建对象，首先判断连接是否成功，若成功则将获取的信息重新组装，组装完毕后关闭输入流，关闭连接，并将最后的信息返回。

实例 3　制作成绩柱状图

Google 公司不仅仅提供了搜索功能，在其他方面提供的服务功能也非常强大，Google 公布了制图服务（Google Chart）的接口，可以用来为统计数据自动生成图片，在本节将对其进行详细介绍。

【实例描述】

本软件设计为一个成绩管理器。该软件在主界面有一个连接按钮，四个分值区域，以及生

成柱状图按钮。

用户首先需要确定手机与互联网是否能够成功连接，若连接成功，则只需要在四个分值区域填写人数，单击生成柱状图按钮后，系统根据所填写的数据自动生成一幅柱状图。

本实例的运行效果如图 9-3 所示。

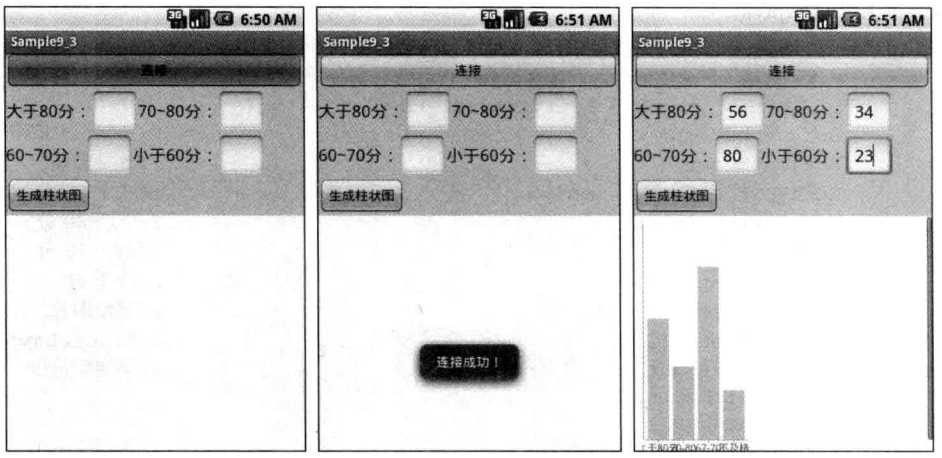


图 9-3　在手机中生成的柱状图

提示：在图 9-3 中，左图为软件开始运行时的效果，中图为单击连接按钮后，系统提示连接成功，右图为输入四个分值段数据后，单击“生成柱状图”按钮，在按钮的下方自动生成的柱状图。

【实现过程】

在本实例中使用到的 API 地址为 http://chart.apis.google.com/chart，在该 API 地址的后面需要添加参数，其格式为“参数名=参数值”，不同的参数之间使用“&”分隔，不分先后次序。该 API 后面所带的参数可能会很多，但每个参数名都有其特定含义，其具体说明如表 9-1 所示。

表 9-1　参数名及其含义

参 数 名	代表的含义	举　例
chs	表示图片的大小	chs=200x200
chd=t:	表示图片所用的数据	chd=t:56,34,80,23
cht	表示图片的类型	cht=bvs
chl	表示图片所用文字的内容	chl=大于 80 分|70～80

Google Chart 提供的服务功能包括折线图（line charts）、条状图（bar charts）、饼图（pie charts）、Venn 图（venn diagrams）和散点图（scatter plots）等。

在该实例运行成功时，需要判断手机网络是否成功连接，该判断是在连接按钮的监听器中完成的。自定义方法 getBVS，在该方法中组成要显示的远程图像的网址，最后以<img src=””>的方式组成 HTML TAG，将图像在设定的 WebView 中显示。

【代码解析】

经过上面的理论介绍，接下来将进入核心代码的讲解部分，Sample9_3_Activity 类中主要

的代码为自定义的 getConnect 方法和 getBVS 方法，其中 getConnect 方法为判断设定的 uri 是否连接成功，连接成功后则使用 getBVS 方法获取图片。

首先介绍的是 Sample9_3_Activity 类框架的设计与开发，代码如下。

代码位置：见随书光盘中源代码/第 9 章/Sample9_3/src/com/bn/chap9/tb 目录下的 Sample9_3_Activity.java。

```
1   package com.bn.chap9.tb;
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码
3   public class Sample9_3_Activity extends Activity {
4       Button button;                                          //Button 按钮
5       Button bConn;                                           //连接按钮
6       EditText etEight;                                       //大于 80 分
7       EditText etSeven;                                       //70～80 分
8       EditText etSix;                                         //60～70 分
9       EditText etFive;                                        //小于 60
10      WebView view;                                           //显示图片
11      LinearLayout ll;                                        //LinearLayout 布局
12      String strEight;                                        //声明字符串
13      String strSeven;
14      String strSix;
15      String strFive;                                         //声明字符串
16      boolean flag=false;
17      public void onCreate(Bundle savedInstanceState) {
18        super.onCreate(savedInstanceState);
19        setContentView(R.layout.main);                        //设置当前界面
20        button=(Button)this.findViewById(R.id.Button01);      //生成图片按钮
21        bConn=(Button)this.findViewById(R.id.Button02);       //连接按钮
22        etEight=(EditText)this.findViewById(R.id.EditText80); //大于 80 分
23        etSeven=(EditText)this.findViewById(R.id.EditText70); //70～80 分
24        etSix=(EditText)this.findViewById(R.id.EditText60);   //60～70 分
25        etFive=(EditText)this.findViewById(R.id.EditText50);  //小于 60 分
26        ll=(LinearLayout)this.findViewById(R.id.LinearLayout03);
27        view=new WebView(Sample9_3_Activity.this);            //创建 WebView 对象
28        ll.addView(view);                                     //添加到 LinearLayout 中
29        final String uri="http://code.google.com/intl/zh-CN/apis/chart/";
                                                                //设定 uri
30        bConn.setOnClickListener(                             //连接按钮监听器
31          new OnClickListener(){
32                    public void onClick(View v) {
33                        flag=getConnect(uri);                 //判断连接是否成功
34                        if(flag){                             //若连接成功
35                            Toast.makeText(                   //创建 Toast 对象
36                                    Sample9_3_Activity.this,
37                                    "连接成功！",
38                                    Toast.LENGTH_SHORT).show();    //提示
39                            button.setClickable(true);        //设置按钮可单击
40                        }else{
41                            Toast.makeText(                   //创建 Toast
42                                    Sample9_3_Activity.this,
43                                    "连接失败！",
44                                    Toast.LENGTH_SHORT).show();    //提示
45                        }}});
46        button.setOnClickListener(                                  //添加监听器
47          new OnClickListener(){
48                    public void onClick(View v) {
49                        strEight=etEight.getText().toString().trim();
```

```
                                                    //获取大于 80 分数据
50                      strSeven=etSeven.getText().toString().trim();
                                                    //获取 70～80 分数据
51                      strSix=etSix.getText().toString().trim();
                                                    //获取 60～70 分数据
52                      strFive=etFive.getText().toString().trim();
                                                    //获取小于 60 分数据
53                      String parent="[0-9]*";     //全为数字
54                      if(strEight.length()==0){   //若长度为 0
55                          Toast.makeText(         //创建 Toast
56                                  Sample9_3_Activity.this,
57                                  "请输入大于 80 分的数据！",
58                                  Toast.LENGTH_SHORT).show();    //提示
59                      }else if(!strEight.matches(parent)){   //若不全为数字
60                          Toast.makeText(                    //创建 Toast
61                                  Sample9_3_Activity.this,
62                                  "数据不包含字母！",
63                                  Toast.LENGTH_SHORT).show();    //提示
64                      }else if(strSeven.length()==0){            //若长度为 0
65                          Toast.makeText(                    //创建 Toast
66                                  Sample9_3_Activity.this,
67                                  "请输入 70～80 分的数据！",
68                                  Toast.LENGTH_SHORT).show();    //提示
69                      }
70                      ……//该处省略了部分类似代码，读者可自行查看随书光盘中源代码。
71                      }else if(!strFive.matches(parent)){    //若不全为数字
72                          Toast.makeText(                    //创建 Toast
73                                  Sample9_3_Activity.this,
74                                  "数据不包含字母！",
75                                  Toast.LENGTH_SHORT).show();    //提示
76                      }else{
77                          String msg=getBVS();           //调用 getBVS 方法
78                          view.loadData(msg, "text/html", "utf-8");
                                                    //在 WebView 中显示图片
79                      }}});}
80      public String getBVS(){
81        ……//该处省略了部分代码，在后面会详细介绍
82      }
83      public boolean getConnect(String uri){
84        ……//该处省略了部分代码，在后面会详细介绍
85      }}
```

其中：

- 第 1～16 行为相关包的导入和该类成员变量的声明。
- 第 17～29 行主要完成对象的创建，其中将创建的 WebView 对象添加到 LinearLayout 中，为后面显示图片做好准备。
- 第 30～45 行主要完成连接按钮的监听器设置，在该监听器内，首先判断给出的 uri 是否能够连接成功，若连接成功，则能单击获取图片按钮；若连接不成功，则弹出 Toast 提示连接失败。
- 第 46～79 行主要完成生成柱状图按钮的监听器设置，在监听器内首先获取所填写的数据，然后判断数据是否为空字符串或者获取的数据中不全为数字，若符合要求则弹出 Toast 提示用户填写正确的数据，若不符合要求则在 WebView 中显示图片。
- 第 80～85 行为自定义检验手机网络是否连接和获取 BVS 图片的方法，该处两个方法在后面会详细介绍。

（1）上述代码主要是对本类的框架的介绍，接下来介绍的是该类的 getBVS()方法，该方法为组装一个 HTML TAG 的字符串，将下列代码插入到上述 Sample9_3_Activity 类框架的第 81 行。

代码位置：见随书光盘中源代码/第 9 章/Sample9_3/src/com/bn/chap9/tb 目录下的 Sample9_3_Activity.java。

```
String result=null;                                          //声明字符串为 null
try{
    result= "<html><body><img src=http://chart.apis.google.com/chart?" +
                                                             //组装字符串
            "cht=bvs&chd=t:"+strEight+","+strSeven+","+strSix+","+strFive+
            "&chs=250x250&chl=大于 80 分|70-80|67-70|不及格>" +"</body></html>";
  }catch(Exception e){e.printStackTrace();}                  //打印异常
    return result;                                           //返回组装的字符串
}
```

提示：本方法主要是自定义 getBVS 方法，在该方法中将获取成绩段的数据并对数据重新组装，将其封装为 xml 文件，使其能够在 WebView 显示。

（2）在介绍完 getBVS 方法后，需要介绍的是该类的 getConnect()方法的开发，该方法为判断手机网络是否连接成功，将下列代码插入到 Sample9_3_Activity 类框架的第 84 行。

```
try{
   HttpURLConnection connection=null;                  //声明 HttpURLConnection 引用
    URL url=new URL(uri);                              //创建 URL 对象
   connection=(HttpURLConnection)url.openConnection();  //打开连接
   connection.setRequestMethod("GET");                 //设定模式
   connection.setDoOutput(true);                       //设置为 true
   connection.setDoInput(true);
    connection.setRequestProperty("User-Agent",
        "Mozilla/4.0(compatible;MSIE 6.0; Windows 2000)");
    connection.setRequestProperty("Content-type", "text/html;charset=utf-8");
   connection.setConnectTimeout(10000);                //等待 10 秒钟
   connection.connect();                               //连接
   int code=connection.getResponseCode();
   if(code==200){                                      //若为 200，表示连接成功
    return true;                                       //返回 true
   }else{
    return false;                                      //若连接失败，返回 false
   }
}catch(Exception e){
    e.printStackTrace();                               //打印异常
    return false;                                      //返回 false
}
```

提示：在该方法中首先声明 HttpURLConnection 引用，创建 URL 对象。打开连接并将 setDoOutput 和 setDoInput 方法设置为 true，设定编码为 utf-8，然后调用 connect 方法，判断是否连接成功，若成功则返回 true，否则返回 false。

实例 4　实现 Google 地图

当下很多中高端手机都带有 GPS 定位功能，这给用户的出行带来了极大的方便。而 Android 由于 Google 的大力推动，加上著名的 Google Map 服务有着其得天独厚的优势。

【实例描述】

本软件主界面包含经度和纬度的编辑框以及显示地图上的位置，当给定 GPS 经纬度坐标后，便会在 Google Map 中绘制气球，提醒用户所查询地点的位置。该软件的实现在用户出行查询指定 GPS 经纬度时，提供了非常大的帮助。

本实例的运行效果如图 9-4-1 所示。

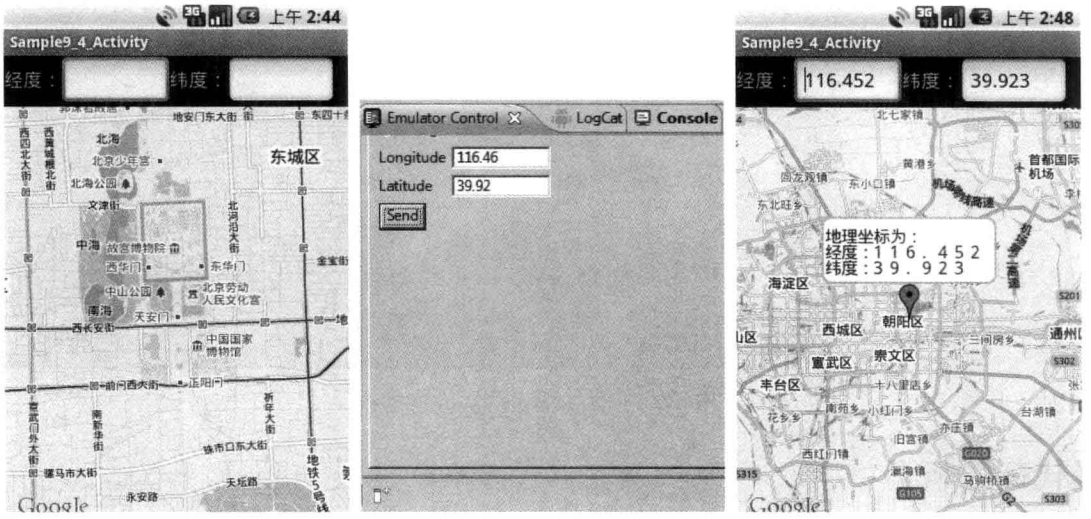


图 9-4-1　指定经纬度查询

提示：在图 9-4-1 中依次为软件运行时的主界面，在 Emulator Control 中输入经纬度坐标，以及单击气球获取输入的经纬度坐标，并在地图上显示其位置。

【实现过程】

GPS 定位功能是 Android 手机的标准配置，并且也给开发人员提供了方便的编程接口。想要使用 Android 中的 GPS 定位功能，首先需要在 AndroidManifest.xml 中声明权限，代码为：<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />。

显示地图所使用的是 MapView 对象，com.google.android.maps.MapView 为其 main.xml 文件的声明方式。Google Map 为开发人员提供了强大的二次开发功能，支持各种客户端。在 Android 平台下进行 Google Map 开发前需要取得 Map 的键值，其步骤如下所列。

（1）首先在 Eclipse 下的 DDMS 中获取 debug.keystore 的路径，如图 9-4-2 所示。

（2）进入 DOS 命令模式，将路径设置到 JAVA_HOME 的 bin 目录下，如笔者的路径为：C:\Program Files\Java\jdk1.6.0_20\bin。

（3）执行如下命令获取 MD5 的拇指纹。path=C:\Program Files\ Java\jdk1.6.0_20\bin>keytool –list –alias androiddebugkey –keystore “C:\Documents and Settings\Administrator\.android\debug.keystore” –storepass android –keypass android

（4）计算出 MD5 的拇指纹后从如下网址申请 Map 键值：http://code.google.com/intl/zh-CN/android/maps-api-signup.html。

（5）获取键值后就可以进行 Google Map 应用程序的开发了。

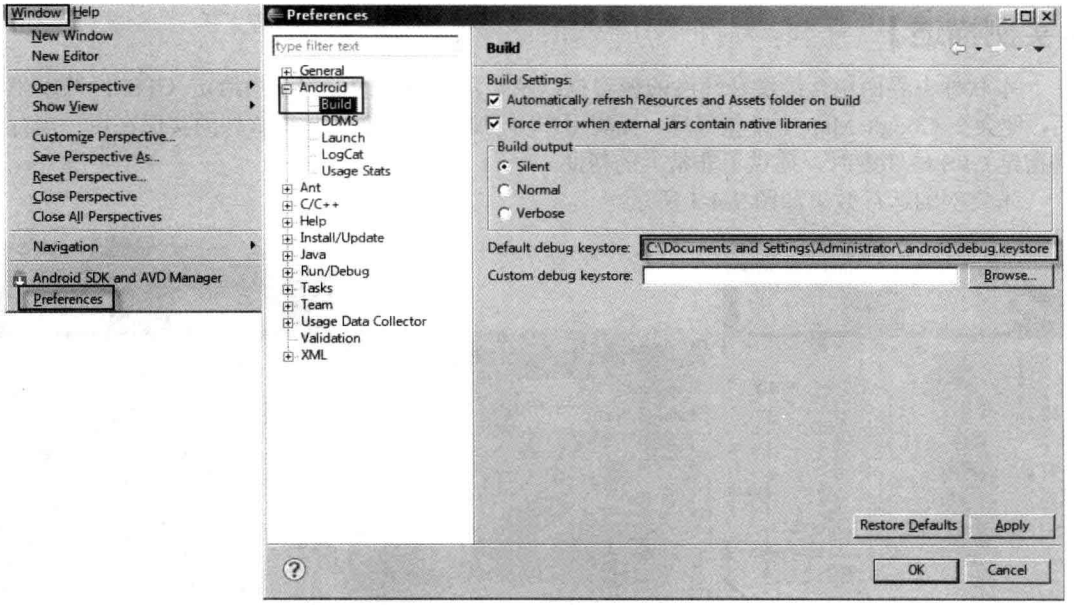


图 9-4-2　获取 debug.keystore 路径

【代码解析】

经过上面的理论介绍，在本部分便要介绍该软件的核心代码部分，首先介绍的是 AndroidManifest.xml 文件，在该文件中完成权限的设置，代码如下所列。

代码位置：见随书光盘中源代码/第 9 章/Sample9_4/目录下的 AndroidManifest.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>                    <!--版本号和编码方式-->
2   <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3        package="com.bn.chap9.map"
4        android:versionCode="1"
5        android:versionName="1.0">                           <!--版本名称-->
6      <application android:icon="@drawable/icon" android:label="@string/app_name">
7        <activity android:name=".Sample9_4_Activity"
8                android:label="@string/app_name">            <!--显示的程序名称-->
9           <intent-filter>
10             <action android:name="android.intent.action.MAIN" />
                                                              <!--标识程序开始-->
11             <category android:name="android.intent.category.LAUNCHER" />
12          </intent-filter>
13       </activity>
14       <uses-library android:name="com.google.android.maps"/>
15     </application>
16     <uses-sdk android:minSdkVersion="7" />
17     <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
18     <uses-permission android:name="android.permission.INTERNET" />
19  </manifest>
```

提示：上述代码中第 17、18 行为使用 GPS 功能和上网功能的权限声明。

接下来介绍 Sample9_4_Activity 类的设计与实现，该类的代码如下所列。

代码位置：见随书光盘中源代码/第 9 章/Sample9_4/src/com/bn/chap9/map 目录下的 Sample9_4_Activity.java。

```
package com.bn.chap9.map;
……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
public class Sample9_4_Activity extends MapActivity {
    static Bitmap bitmap;                                    //气球图片
    MapController mc;                                        //地图控制器
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);                         //设置当前界面
      bitmap= BitmapFactory.decodeResource(
            this.getResources(), R.drawable.ballon);         //初始化气球图片
      String serviceName=Context.LOCATION_SERVICE;           //获取位置管理器实例
      LocationManager lm=(LocationManager)this.getSystemService(serviceName);
      LocationListener ll=new LocationListener(){            //位置变化监听器
            public void onLocationChanged(Location location) {
                                                             //当位置变化时触发
                updateWhenNewLocation(location);
            }
            public void onProviderDisabled(String provider) {
                                                  //Location Provide 被禁用时更新
            }
            public void onProviderEnabled(String provider) {
                                                //Location Provider 被启用时更新
            }
            public void onStatusChanged(String provider, int status,Bundle
extras) {
            } };
      lm.requestLocationUpdates(                             //添加位置变化监听器
            LocationManager.GPS_PROVIDER,                    //使用 GPS 定位
            2000,                                            //时间分辨率 ms
            5,                                               //距离分辨率 m
            ll                                               //位置变化监听器
      );
      MapView mv=(MapView)findViewById(R.id.myMapView); //对地图进行初始化
      mv.setBuiltInZoomControls(true);                       //设置地图上要缩放的控制条
      mc=mv.getController();                                 //获取地图控制器
      mc.setZoom(14);                                        //设置地图缩放比例
      GeoPoint gp = new GeoPoint(
            (int)(39.9083*1E6),                              //纬度
            (int)(116.3975*1E6)                              //经度
      );
      mc.animateTo(gp);}                                     //设置经纬度
    //根据位置的变化更新地图与文本信息的方法
    public void updateWhenNewLocation(Location location){
        String result=null;
        if(location!=null){
        String latStr=Math.round(location.getLatitude()*1000)/1000.0+"";
                                                             //纬度
        String  longStr=Math.round(location.getLongitude()*1000)/1000.0+"";
                                                             //经度
        result="经度: "+longStr+" 纬度: "+latStr;            //组装字符串
        EditText la=(EditText)Sample9_4_Activity.this.findViewById(R.id.
EditText01); //经度
        EditText lo=(EditText)Sample9_4_Activity.this.findViewById(R.id.
EditText02); //纬度
        la.setText(longStr);                                         //设置文字
        lo.setText(latStr);
        MapView mv=(MapView)findViewById(R.id.myMapView); //创建 MapView 对象
        List<Overlay> overlays = mv.getOverlays();
```

```
51          GeoPoint gp = new GeoPoint(
52                  (int)(location.getLatitude()*1E6), //纬度
53                  (int)(location.getLongitude()*1E6) //经度
54           );
55           mc.animateTo(gp);                                    //设置地图中心点的经纬度
56           MyBallonOverlay mbo=new MyBallonOverlay(
57                  gp,                                           //气球的坐标
58                  "地理坐标为: \n 经度: "+longStr+"\n 纬度: "+latStr //气球的信息
59          );
60           mbo.showWindow=true;                                 //设置新气球的信息窗口打开
61           overlays.add(mbo);
62      }
63      else{
64          result="对不起，当前 GPS 不可用! ";
65          TextView tv=(TextView)findViewById(R.id.TextView01);
66          tv.setText(result);                                   //设置提醒
67      }}
68      protected boolean isRouteDisplayed() {
69          return false;                                         //返回 false
70  }}
```

其中：

- 第 6～37 行主要完成设置 main.xml 为当前界面，初始化气球图片，获取位置管理器。为位置变化设置监听器，并重写该监听器的方法。初始化地图，设置地图上要缩放的控制条，最后设置初始的经纬度。
- 第 39～67 行主要完成根据位置的变化更新地图与文本信息的方法，首先将获取的经纬度填写到 EditText 中，创建 Overlay 类型的 List 对象和 GeoPoint 对象，将该地图的中心点设置为获取的经纬度，并在该地点绘制气球，气球的上方绘制信息窗口。

下面介绍绘制气球和信息窗口的实现类，首先介绍绘制气球的方法，代码如下所列。

代码位置：见随书光盘中源代码/第 9 章/Sample9_4/src/com/bn/chap9/map 目录下的 MyBallonOverlay.java。

```
1   package com.bn.chap9.map;
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3   class MyBallonOverlay extends Overlay{
4       final static int picWidth=20;                            //气球图的宽度
5       final static int picHeight=34;                           //气球图的高度
6       final static int arcR=8;                                 //信息窗口的圆角半径
7       static MyBallonOverlay currentBallon=null;               //表示当前选中的气球
8       String msg;                                              //此气球对应的文字信息
9       boolean showWindow=false;                        //是否显示文字信息窗口的标志位
10      GeoPoint gp;                                             //此气球对应的经纬度
11      public MyBallonOverlay(GeoPoint gp,String msg){          //构造器
12          this.gp=gp;
13          this.msg=msg;
14      }
15     public boolean onTouchEvent(MotionEvent event, MapView mv) {
16        if(currentBallon!=null&&currentBallon!=this){ //若当前气球不为空且不是自己
17          return false;                                        //不执行任何操作
18        }
19       if(event.getAction() == MotionEvent.ACTION_DOWN) {
                                                          //若在气球上按下，设置为当前气球
20          int x=(int)event.getX();
21         int y=(int)event.getY();
22         Point p= getPoint(mv);
23         int xb=p.x-picWidth/2;
```

```
24          int yb=p.y-picHeight;
25          if(x>=xb&&x<xb+picWidth&&y>=yb&&y<yb+picHeight){
                                                    //若将当前气球设置为自己
26              currentBallon=this;
27              return true;
28          }}
29      else if (event.getAction() == MotionEvent.ACTION_MOVE)
30      {//移动事件返回当前气球状态，若在当前气球上则返回 true，屏蔽其他移动事件
31          return currentBallon!=null;
32      }
33        else if (event.getAction() == MotionEvent.ACTION_UP) {
34          int x=(int)event.getX();                //获取触控笔位置
35          int y=(int)event.getY();
36          Point p= getPoint(mv);                  //获取气球在屏幕上的坐标范围
37          int xb=p.x-picWidth/2;
38          int yb=p.y-picHeight;
39
if(currentBallon==this&&x>=xb&&x<xb+picWidth&&y>=yb&&y<yb+picHeight)
40          {//若当前气球为自己且在当前气球上抬起触控，则显示当前气球内容
41              currentBallon=null;                 //显示完内容后清空当前气球
42              showWindow=!showWindow;
43              List<Overlay> overlays = mv.getOverlays();
44              overlays.remove(this);              //删除此气球再添加
45              overlays.add(this);                 //此气球就位于最上面了
46              for(Overlay ol:overlays){           //清除其他气球标志位
47                  if(ol instanceof MyBallonOverlay){
48                      if(ol!=this){
49                          ((MyBallonOverlay)ol).showWindow=false;
                                                    //将 showWindow 设为 false
50                      }}}
51              return true;
52          }
53          else if(currentBallon==this)
54          {//若当前气球为自己但抬起触控不在自己上，则清空气球状态并返回 true
55              currentBallon=null;
56              return true;
57          }}
58        return false;
59      }
60       public void draw(Canvas canvas, MapView mapView, boolean shadow) {
61       Point p= getPoint(mapView);                //将经纬度转为 X、Y 坐标
62          canvas.drawBitmap(                      //在坐标指定位置绘制气球
63              Sample9_4_Activity.bitmap, p.x-picWidth/2, p.y-picHeight, null);
64          if(showWindow){                         //若标志位为 true 则显示信息窗口
65              drawWindow(canvas,p,160);           //绘制信息窗口
66          }
67          super.draw(canvas, mapView, shadow);    //调用父类绘制
68       }
69      public Point getPoint(MapView mapView){     //将经纬度转为屏幕上的 X、Y 坐标
70          Projection projettion = mapView.getProjection();
71          Point p = new Point();                  //创建对象
72          projettion.toPixels(gp, p);
73          return p;                               //返回坐标
74      }
75   /*该处省略了绘制信息窗口的方法，将在下面给出*/
76}
```

其中：

- 第 4～14 行为设置该类的成员变量和构造器，在构造器内获取 GeoPoint 对象引用以及

msg 字符串。

- 第 15～59 行为重写 onTouchEvent 方法，在该方法中首先判断是否为自己，若不是自己则返回 false，判断 event 事件是否为键按下事件，若为该事件，判断若在气球上按下则设置当前气球为自己，且当前状态为在气球上；然后判断 event 事件是否为 MotionEvent.ACTION_MOVE，若是该事件，返回当前气球状态；若当前在气球上，则返回 true 屏蔽其他移动事件；最后判断该事件是否为 MotionEvent.ACTION_UP，若为该事件则绘制信息窗口。
- 第 60～74 行为绘制气球和将经纬度转为屏幕上 X、Y 坐标的两个方法，绘制信息窗口时首先调用将经纬度转为屏幕上 X、Y 坐标的方法，然后在指定位置绘制气球，并根据标志位判断是否绘制信息窗口。getPoint 方法通过接收 MapView 参数，将经纬度转换为手机屏幕上的 X、Y 坐标。

气球的绘制开发完成后，下一步要做的就是绘制信息窗口，实现绘制信息窗口的方法名称为 drawWindow。将下列代码插入到上述代码的第 75 行。

代码位置：见随书光盘中源代码/第 9 章/Sample9_4/src/com/bn/chap9/map 目录下的 MyBallonOverlay.java。

```
1       public void drawWindow(Canvas canvas,Point p,int winWidth) {
                                                          //绘制信息窗口的方法
2           int charSize=15;
3           int textSize=16;
4           int leftRightPadding=2;
5           int lineWidth=winWidth-leftRightPadding*2;        //每行的宽度
6           int lineCharCount=lineWidth/(charSize);           //每行字符数
7           ArrayList<String> alRows=new ArrayList<String>();
                                                    //记录所有行的 ArrayList
8           String currentRow="";                             //当前行的字符串
9           for(int i=0;i<msg.length();i++){
10              char c=msg.charAt(i);
11              if(c!='\n'){                                  //若当前字符不为换行符
12                  currentRow=currentRow+c;
13              }
14              else{
15                  if(currentRow.length()>0){                //若长度大于 0
16                      alRows.add(currentRow);               //加入列表中
17                  }
18                  currentRow="";                            //清空当前行
19              }
20              if(currentRow.length()==lineCharCount){
21                  alRows.add(currentRow);
22                  currentRow="";                            //清空当前行
23              }}
24          if(currentRow.length()>0){                        //若当前行长度大于零
25              alRows.add(currentRow);         //将当前行加入记录所有行的 ArrayList
26          }
27          int lineCount=alRows.size();                      //获得总行数
28          int winHeight=lineCount*(charSize)+2*arcR;        //自动计算信息窗体高度
29          Paint paint=new Paint();                          //创建 paint 对象
30          paint.setAntiAlias(true);                         //打开抗锯齿
31          paint.setTextSize(textSize);                      //设置文字大小
32          paint.setARGB(255, 255,251,239);                  //绘制信息窗体圆角矩形
33          int x1=p.x-winWidth/2;
34          int y1=p.y-picHeight-winHeight-1;
35          int x2=x1+winWidth;                               //设定起始位置
```

```
36            int y2=y1+winHeight;
37            RectF r=new RectF(x1,y1,x2,y2);                    //绘制矩形
38            canvas.drawRoundRect(r, arcR, arcR, paint);
39            paint.setARGB(255,198,195,198);                    //绘制信息窗体圆角矩形边框
40            paint.setStyle(Paint.Style.STROKE);                //设置字体
41            paint.setStrokeWidth(2);
42            canvas.drawRoundRect(r, arcR, arcR, paint);
43            paint.setStrokeWidth(0);                           //循环绘制每行文字
44            paint.setARGB(255, 10, 10, 10);
45            int lineY=y1+arcR+charSize;
46            for(String lineStr:alRows){                        //对每一行进行循环
47                for(int j=0;j<lineStr.length();j++){           //对一行中的每个字循环
48                    String colChar=lineStr.charAt(j)+"";
49                    canvas.drawText(colChar,   x1+leftRightPadding+j*charSize,
lineY, paint);
50                }
51                lineY=lineY+charSize;                          //y 坐标移向下一行
52  }}}
```

其中：

- 第 1～8 行为声明控制字体大小，留白，每行宽度和字符数的变量，并创建记录所有行的 ArrayList 对象。
- 第 9～23 行为使用 for 循环重新组装字符串，若当前字符为换行符则检查当前行长度，若当前行长度大于零，则将当前行加入记录所有行的 ArrayList；若当前行的长度达到一行规定的字符数，则将当前行加入记录所有行的 ArrayList。
- 第 24～52 行首先判断，若当前行长度大于零，将当前行加入记录所有行的 ArrayList，然后设置所要绘制矩形的属性，绘制矩形，设定要绘制的文字信息，并将文字绘制在矩形内。

实例 5　Google 地图地点查询功能

在上一节简单介绍了 Google Map 的实现，在本节将对其进一步进行讲解，将其设计成一个具有查询地点功能的软件，下面将对其的设计与实现进行详细介绍。

【实例描述】

该软件仅含有一个界面，运行该软件进入主界面后，在界面的上层可以填写要查询地点的名称，该名称可以分为中文和英文。单击界面中的放大镜时，系统自动按照所填写的地点名称在地图上查询，并在该地点绘制一个气球和信息窗口。

单击第二个图表时，会弹出地点历史记录对话框，单击历史记录对话框中的历史记录时，系统会自动按照地点名称再次查找该地点。单击星星图表时，弹出提示信息，可以选择将地图切换为普通模式或者卫星模式，增加用户体验。

本实例的运行效果如图 9-5 所示。

提示：在图 9-5 中，左图为填写地点名称后查询的结果图，中图为查看地点历史记录，右图为选择地图模式。

【实现过程】

在该软件中除了继续使用上一节中介绍的主要知识点外，使用 Geocoder 查找指定名称景点

的经纬度列表，以及自定义对话框 Dialog 的实现，在自定义的 Dialog 中加入 ListView，将填写的地点名称记录，然后显示在 ListView 中，当单击其中的一个记录时，返回系统地点名称。

图 9-5　Google Map 地点查询

【代码解析】

经过上面的理论介绍，相信读者对此软件有了一定的了解，下面就将对其核心代码进行详细介绍，首先介绍的是 AndroidManifest.xml 文件，该文件的代码如下所列。

代码位置：见随书光盘中源代码/第 9 章/Sample9_5/目录下的 AndroidManifest.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>                    <!--版本号和编码方式-->
2   <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3       package="com.bn.chap9.ddcx"
4       android:versionCode="1"
5       android:versionName="1.0">                            <!--版本名称-->
6     <application android:icon="@drawable/icon" android:label="@string/app_name">
7       <activity android:name=".Sample9_5_Activity"
8               android:label="@string/app_name">             <!--显示的程序名称-->
9         <intent-filter>
10          <action android:name="android.intent.action.MAIN" />
                                                              <!--标识程序开始-->
11          <category android:name="android.intent.category.LAUNCHER" />
12        </intent-filter>
13      </activity>
14      <uses-library android:name="com.google.android.maps"/>
15    </application>
16    <uses-sdk android:minSdkVersion="7" />
17    <uses-permission android:name="android.permission.INTERNET" />
18  </manifest>
```

提示：上述代码中第 14 行为声明 Google Map，第 17 行为添加访问网络的权限。

下面介绍 Sample9_5_Activity 类的设计与实现，首先要介绍的是该类的框架，代码如下。

代码位置：见随书光盘中源代码/第 9 章/Sample9_5/src/com/bn/chap9/ddcx 目录下的 Sample9_5_Activity.java。

```
1   package com.bn.chap9.ddcx;
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码。
3   public class Sample9_5_Activity extends MapActivity {
4       final int HISTORY_DIALOG=0;                          //历史对话框的 ID
5       final int MODE_DIALOG=1;                             //模式选择对话框的 ID
6       static Bitmap bitmap;                                //气球图片
7       MapController mc;                                    //地图控制器
8       String jdmcArray[];                                  //景点名称数组
9       Dialog historyDialog;                                //历史对话框的引用
10      public void onCreate(Bundle savedInstanceState) {
11        super.onCreate(savedInstanceState);
12        setContentView(R.layout.main);                     //设置当前界面
13        bitmap= BitmapFactory.decodeResource(this.getResources(), R.drawable.
ballon); //初始化图片
14        MapView mv=(MapView)findViewById(R.id.myMapView); //对地图进行初始化
15        mv.setBuiltInZoomControls(true);                   //设置地图上要缩放的控制条
16        mc=mv.getController()   ;                          //获取地图控制器
17        mc.setZoom(14);                                    //设置地图缩放比例
18        GeoPoint gp = new GeoPoint(
19              (int)(39.9083*1E6),                          //纬度
20              (int)(116.3975*1E6)                          //经度
21        );
22        mc.animateTo(gp);                                  //设置经纬度
23        ImageButton ib=(ImageButton)this.findViewById(R.id.ImageButton01);
                                                             //给查询按钮添加监听器
24        ib.setOnClickListener(
25          new OnClickListener(){
26              public void onClick(View v) {
27              EditText et=(EditText)findViewById(R.id.EditText01);
                                                             //获取输入景点的名称
28              String jdmcStr=et.getText().toString();
29              //通过 Geocoder 查找指定名称景点的经纬度
30              Geocoder gc=new Geocoder(Sample9_5_Activity.this,Locale.
getDefault());
31              try {
32                  //通过 Geocoder 查找指定名称景点的经纬度列表
33                List<Address> addressList=gc.getFromLocationName(jdmcStr, 1);
34                if(addressList.size()>0){                  //取列表中的第一条
35                  Address tempa=addressList.get(0);
36                  int latE6=(int)(tempa.getLatitude()*1000000);
37                  int longE6=(int)(tempa.getLongitude()*1000000);
38                  GeoPoint gp=new GeoPoint(latE6, longE6);
39                  String latStr=Math.round(gp.getLatitudeE6()/1000.00)/1000.
0+"";        //纬度
40                  String longStr=Math.round(gp.getLongitudeE6()/1000.00)/
1000.0+"";   //经度
41                  MapView mv=(MapView)findViewById(R.id.myMapView);
                                                             //在地图的对应位置显示气球
42                List<Overlay> overlays = mv.getOverlays();    //获取 Overlay
43                boolean isExisted=false;
44                MyBallonOverlay curr=null;
45                for(Overlay ol:overlays){                  //循环扫描列表
46                  if(ol instanceof MyBallonOverlay){
47                    MyBallonOverlay tempMbo=(MyBallonOverlay)ol;
48                    tempMbo.showWindow=false;      //关闭已经存在景点的信息提示窗口
49                    if(gp.getLatitudeE6()==tempMbo.gp.getLatitudeE6()&&
50                    gp.getLongitudeE6()==tempMbo.gp.getLongitudeE6()){
                                                             //景点已经存在
51                    isExisted=true;
```

```
52                      tempMbo.showWindow=true;        //打开要查询地点的信息提示窗口
53                      curr=tempMbo;
54                      } }}
55                      if(!isExisted){                 //如果景点不存在添加新气球
56                          MyBallonOverlay mbo=new MyBallonOverlay(
                                                        //在单击位置添加新气球
57                          gp,                         //气球的坐标
58                          jdmcStr+"的地理坐标为:\n 经度:"+longStr+"\n 纬度:"+latStr,
                                                        //气球的信息
59                          jdmcStr                     //景点名称
60                         );
61                        mbo.showWindow=true;          //设置新气球的信息窗口打开
62                        overlays.add(mbo);            //将新气球添加到地图上
63                      }
64                      else {                          //将此气球移动到最上层
65                            overlays.remove(curr);
66                            overlays.add(curr);
67                       }
68                          mc.animateTo(gp);           //设置地图中央经纬度
69                       }
70                       else{
71                        Toast.makeText(               //创建 Toast 对象
72                                  Sample9_5_Activity.this,
73                                  "对不起，你要找的景点没有找到！",
74                                  Toast.LENGTH_SHORT).show();    //显示提示信息
72                      } }
73                        catch (IOException e) { e.printStackTrace();} //打印异常
74      }});
75  ……//该处省略了部分代码，将在后面给出
76  }
77  /*该处省略了创建对话框的代码，将在后面给出*/
78  /*该处省略了部分方法，将在后面给出*/
79  }
```

其中：

- 第 4～9 行为声明该类成员变量。
- 第 10～23 行为创建该类的对象，并设置地图缩放比例和开始地图的中心点位置。
- 第 24～74 行为给查询按钮添加监听器，在该监听器内首先获取输入的地点名称，随后通过 Geocoder 查找指定名称景点的经纬度，创建 Overlay 类型列表，循环扫描该列表，关闭已经存在地点的信息提示窗口，然后将新地点的信息窗口标志位设为 true，并在该地点绘制气球。

（1）上述代码主要是对本类框架的介绍，接下来介绍历史按钮和模式按钮的监听器设置，将下列代码插入到 Sample9_5_Activity 类框架的第 75 行。

代码位置：见随书光盘中源代码/第 9 章/Sample9_5/src/com/bn/chap9/ddcx 目录下的 Sample9_5_Activity.java。

```
1  ib=(ImageButton)this.findViewById(R.id.ImageButton02);
                                                        //获得 ImageButton02 的引用
2  ib.setOnClickListener(                               //给历史按钮添加监听器
3       new OnClickListener(){
4           public void onClick(View v) {
5               MapView mv=(MapView)findViewById(R.id.myMapView);
6               List<Overlay> overlays = mv.getOverlays(); //获取列表信息
7               if(overlays.size()!=0){                 //若当前有历史记录
8                   showDialog(HISTORY_DIALOG);         //显示对话框
```

```
9                }
10               else{                                          //若当前没有历史记录则提示
11                   Toast.makeText(                            //创建 Toast 对象
12                       Sample9_5_Activity.this,
13                       "对不起，目前没有历史记录！",
14                       Toast.LENGTH_SHORT).show();            //显示 Toast
15                   }}});
16   ib=(ImageButton)this.findViewById(R.id.ImageButton03);
17   ib.setOnClickListener(                                     //给模式按钮添加监听器
18       new OnClickListener(){
19           public void onClick(View v) {                      //显示地图模式选择对话框
20               showDialog(MODE_DIALOG);                       //显示对话框
21           }});}
```

其中：

- 第 1～15 行为历史按钮的监听器设置，在该监听器中获取 MapView 对象，获取列表信息，若历史记录不为空，则显示历史记录对话框。
- 第 16～21 行为模式按钮监听器的设置，当单击模式按钮时显示模式选择对话框。

（2）上述代码介绍历史按钮和模式按钮的监听器设置，接下来要介绍的是 onCreateDialog 方法，当创建对话框时，首先会调用该方法，其通过接收的 id 来判断是哪一个对话框，将下列代码插入到 Sample9_5_Activity 类框架的第 77 行。

代码位置：见随书光盘中源代码/第 9 章/Sample9_5/src/com/bn/chap9/ddcx 目录下的 Sample9_5_Activity.java。

```
1    Dialog result=null;                                        //声明 Dialog 引用
2    switch(id){
3         case HISTORY_DIALOG:                                  //历史记录对话框的初始化
4           AlertDialog.Builder b=new AlertDialog.Builder(this); //创建对象
5            b.setItems(
6                null,
7                null
8            );
9            historyDialog=b.create();                          //创建 Dialog
10               result=historyDialog;                          //为 result 赋值
11        break;
12        case MODE_DIALOG:                                     //地图模式选择对话框的初始化
13         AlertDialog.Builder bl=new AlertDialog.Builder(this);
14           bl.setIcon(R.drawable.mode);                       //设置图标
15           bl.setTitle(R.string.mode);                        //设置标题
16           bl.setItems(
17               new String[]{"普通模式","卫星模式"},           //添加名称
18                   new DialogInterface.OnClickListener(){ //匿名内部类
19                       public void onClick(DialogInterface dialog, int which) {
20                           MapView mv=(MapView)findViewById(R.id.myMapView);
21                               switch(which){
22                                  case 0:
23                                     mv.setStreetView(true);
                                                   //设置普通模式为 true
24                                     mv.setSatellite(false);
                                                   //设置卫星模式为 false
25                                   break;
26                                  case 1:
27                                     mv.setSatellite(true); //设置卫星模式为 true
28                                     mv.setStreetView(false);
                                                       //设置普通模式为 false
29                                   break;
30                               }}});
```

```
31            result=bl.create();                                    //创建对话框
32          break;                                                   //退出
33    }
34    return result;                                                 //返回 result
```

其中：

- 第 3～11 行为创建历史记录对话框。
- 第 12～34 行为创建地图模式对话框，在该对话框内首先设置图标和标题，然后创建监听器，当单击普通模式时，将地图的模式设置为普通模式，属性为 true，卫星模式设为 false，若单击的是卫星模式，则将卫星模式设置为 true，普通模式设置为 false。

（3）在介绍完创建对话框的方法后，需要介绍的是每次弹出对话框时被回调以动态更新对话框内容的方法，将下列代码插入到 Sample9_5_Activity 类框架的第 78 行。

代码位置：见随书光盘中源代码/第 9 章/Sample9_5/src/com/bn/chap9/ddcx 目录下的 Sample9_5_Activity.java。

```
1    if(id!=this.HISTORY_DIALOG)return;                             //若不是历史对话框则返回
2       MapView mv=(MapView)findViewById(R.id.myMapView);
3        List<Overlay> overlays = mv.getOverlays();                 //获取当前景点名称的列表
4        jdmcArray=new String[overlays.size()];                     //创建字符串数组
5          int i=0;
6        for(Overlay ol:overlays){                                  //循环扫描 overlays
7            if(ol instanceof MyBallonOverlay){
8                MyBallonOverlay tempMbo=(MyBallonOverlay)ol;
9                jdmcArray[i++]=tempMbo.jdmc;                       //添加到数组中
10           }}
11       LinearLayout ll=new LinearLayout(Sample9_5_Activity.this);
12       ll.setOrientation(LinearLayout.VERTICAL);                  //设置朝向
13       ll.setGravity(Gravity.CENTER_HORIZONTAL);                  //设置为水平居中
14       ll.setBackgroundResource(R.drawable.dialog);
15       LinearLayout lln=new LinearLayout(Sample9_5_Activity.this);
                                                                    //标题行的水平 LinearLayout
16       lln.setOrientation(LinearLayout.HORIZONTAL);               //设置朝向
17       lln.setGravity(Gravity.LEFT);                              //设置为居左
18       lln.setLayoutParams(new ViewGroup.LayoutParams(
19           200, LayoutParams.WRAP_CONTENT));
20       ImageView iv=new ImageView(Sample9_5_Activity.this);       //标题行的图标
21       iv.setImageResource(R.drawable.history);                   //设置图片
22       iv.setLayoutParams(new ViewGroup.LayoutParams(24, 24));
23       lln.addView(iv);                                           //添加 ImageView 对象
24       TextView tvTitle=new TextView(Sample9_5_Activity.this); //标题行的文字
25       tvTitle.setText(R.string.list);
26       tvTitle.setTextSize(20);                                   //设置字体大小
27       tvTitle.setTextColor(Sample9_5_Activity.this.getResources().getColor(R.
color.white));    //设置颜色
28       lln.addView(tvTitle);
29       ll.addView(lln);                                           //将标题行添加到总 LinearLayout
30           ListView lv=new ListView(this);
31       lv.setChoiceMode(ListView.CHOICE_MODE_SINGLE);
32       BaseAdapter ba=new BaseAdapter(){                          //为 ListView 准备内容适配器
33           public int getCount() {
34               return jdmcArray.length;                           //设定选项个数
35           }
36           public Object getItem(int arg0) { return null; }
37           public long getItemId(int arg0) { return 0; }
38           public View getView(int arg0, View arg1, ViewGroup arg2) {
39               TextView tv=new TextView(Sample9_5_Activity.this);
```

```
                                                        //动态生成 TextView
    40              tv.setGravity(Gravity.CENTER_HORIZONTAL);
    41              tv.setText(jdmcArray[arg0]);          //设置内容
    42              tv.setTextSize(20);                   //设置字体大小
    43              tv.setTextColor(Sample9_5_Activity.this.getResources().
getColor(R.color.white));
    44              tv.setPadding(0,0,0,0);               //设置四周留白
    45              return tv;
    46          } };
    47      lv.setAdapter(ba);                            //为 ListView 设置内容适配器
    48      lv.setOnItemClickListener(                    //设置选项被单击的监听器
    49        new OnItemClickListener(){
    50          public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,long
arg3) {
    51              TextView tv=(TextView)arg1;    //获取历史记录中当前选中的 TextView
    52              EditText et=(EditText)Sample9_5_Activity.this.findViewById(
    53              R.id.EditText01);                     //获取地名输入框
    54              et.setText(tv.getText());             //将内容设置到地名输入框
    55              Sample9_5_Activity.this.historyDialog.cancel(); //关闭对话框
    56          }});
    57      ll.addView(lv);                               //将 ListView 加入总 LinearLayout
    58       dialog.setContentView(ll);                   //设置界面
    59    }
    60    protected boolean isRouteDisplayed() {        //重写 isRouteDisplayed 方法
    61        return false;
    62    }
```

其中：

- 第 1～10 行为首先判断所收到的 ID 是否为历史对话框的 id，若不是则直接返回，若是则创建 MapView 对象，获取当前景点名称的列表，并循环扫描该列表，将该列表的信息添加到创建的字符串数组中。
- 第 11～31 行为创建 LinearLayout 对象，并设置 LinearLayout 属性，创建 ImageView 对象，为其添加图片，声明 TextView 对象，设置其显示的字体。最后将创建的对象添加到 LinearLayout 中。
- 第 32～59 行主要完成的是为 ListView 添加适配器和设定监听器，在 BaseAdapter 中首先设定选项的个数，然后在 getView 方法中动态生成 TextView，并设定 TextView 的内容、字体以及四周留白。在监听器内当单击一项内容时，将选中的内容设置到地名输入框中，最后将 ListView 添加到 LinearLayout 中。
- 第 60～62 行为重写 isRouteDisplayed 方法，该方法返回值为 false。

实例 6　随身小词典

在上面讲解了 Google 地图功能的实现，在本节将对 Google 提供的网络翻译服务进行介绍。

【实例描述】

本软件设计简单，在主界面中的文本编辑框为填写英文单词，填写完成后单击下方的翻译按钮即可得到正确翻译，但若填写的单词错误，则显示在编辑框中填写的信息。

本实例的运行效果如图 9-6 所示。

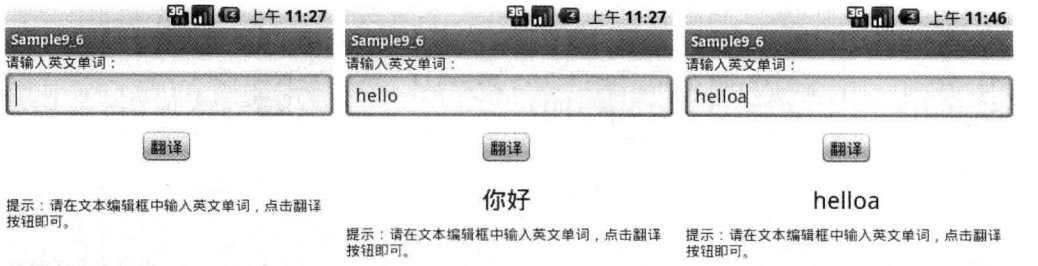


图 9-6　随身词典运行效果图

提示：在图 9-6 中依次表示的是软件运行时的主界面，在文本框中填写正确的英文单词，单击翻译按钮后的效果图，以及填写错误单词时的效果图。

【实现过程】

在本软件中重点使用了 Google 提供的翻译 API，在界面中设置 WebView 用于显示 HTML 文件，使用 HTML 文件里的 JavaScript 语言访问 Google Translate API。首先取得 WebView 的 WebSetting 控制，然后使用 setJavaScriptEnabled()方法设定允许运行 JavaScript。使用 addJavascriptInterface()方法设定给 html 调用的对象及名称，最后载入 google_translate.html。

在 HTML 文件中以 window.irdc.runJavaScript()方法来表示时，HTML 内的程序可以返回调用 Activity 内的 runJavaScript()方法。

【代码解析】

经过上面的理论介绍，相信读者对此有了一定的了解，在本部分将对其核心代码进行详细介绍，首先介绍 AndroidManifest.xml 文件，在该文件中主要完成权限的设置。

代码位置：见随书光盘中源代码/第 9 章/Sample9_6/目录下的 AndroidManifest.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>                    <!--版本号和编码方式-->
2   <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3       package="com.bn.chap9.translate"
4       android:versionCode="1"
5       android:versionName="1.0">                            <!--版本名称-->
6     <application android:icon="@drawable/icon" android:label="@string/app_name">
7       <activity android:name=".Sample9_6_Activity"
8               android:label="@string/app_name">             <!--显示的程序名称-->
9           <intent-filter>
10              <action android:name="android.intent.action.MAIN" />
                                                              <!--标识程序开始-->
11              <category android:name="android.intent.category.LAUNCHER" />
12          </intent-filter>
13      </activity>
14    </application>
```

```
    15      <uses-permission android:name="android.permission.INTERNET"></uses-
permission>
    16      <uses-sdk android:minSdkVersion="7"></uses-sdk>
    17  </manifest>
```

提示：上述代码中第 15 行为权限的声明，第 16 行为 sdkVersion 的设置。

上述代码主要介绍 AndroidManifest.xml 的开发，接下来要介绍 main.xml 的设计与实现，在该文件中主要完成的是界面布局的摆放，代码如下。

代码位置：见随书光盘中源代码/第 9 章/Sample9_6/ res/layout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>      <!--版本号和编码方式-->
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3                 android:orientation="vertical"
4                 android:layout_width="fill_parent"
5                 android:layout_height="fill_parent"
6                 android:id="@+id/LinearLayout01"
7      android:background="#ffffcc">            <!-- LinearLayout 的属性设置-->
8       <TextView
9                 android:id="@+id/myTextView1"
10                android:layout_width="wrap_content"
11                android:layout_height="wrap_content"
12                android:text="请输入英文单词："
13       android:textColor="#222222"
14       >                                       <!--TextView 的属性设置-->
15       </TextView>
16       <EditText
17       android:id="@+id/myEditText1"
18                android:layout_width="fill_parent"
19                android:layout_height="wrap_content"
20                android:textSize="18dip"
21                android:textColor="#222222"
22       >                                       <!--EditText 的属性设置-->
23       </EditText>
24       <WebView
25                android:id="@+id/myWebView1"
26                android:layout_height="wrap_content"
27                android:layout_width="fill_parent"
28                android:focusable="false"
29       android:background="#ffcc66"/>          <!--WebView 的属性设置-->
30       <TextView
31                android:id="@+id/myTextView2"
32                android:layout_width="wrap_content"
33                android:layout_height="wrap_content"
34                android:text="提示：请在文本编辑框中输入英文单词，单击翻译按钮即可。"
35                android:textColor="#222222"
36       >                                       <!--TextView 的属性设置-->
37       </TextView>
38  </LinearLayout>
```

其中：

- 第 2～7 行为 LinearLayout 布局的属性设置，将其中的控件摆放顺序设置为竖直摆放，设定背景色为“#ffffcc”。
- 第 8～15 行为 TextView 的属性设置，在其中设置其显示文本和 ID，将控件的宽设置为充满屏幕，高设定为包裹内容，将字体颜色设置为黑色。
- 第 16～23 行为 EditText 的属性设置，在其中设置控件的 ID，将控件的宽设置为充满屏幕，高设定为包裹内容，将字体颜色设置为黑色。

- 第 24～29 行为设置 WebView 的属性，设定控件的 ID 和大小，并设定焦点为 false。
- 第 30～36 行为 TextView 的属性设置，在其中设置其显示文本和 ID，将控件的宽设置为充满屏幕，高设定为包裹内容，将字体颜色设置为黑色。

上述代码主要介绍的是本程序主界面的开发，接下来要介绍的是 Sample9_6_Activity 类的设计与实现，在该类中完成对软件的系统设置，代码如下。

代码位置：见随书光盘中源代码/第 9 章/Sample9_6/src/com/bn/chap9/translate 目录下的 Sample9_6_Activity.java。

```
1   package com.bn.chap9.translate;                          //声明包
2   import android.app.Activity;                             //导入相关包
3   import android.os.Bundle;                                //导入相关包
4   import android.os.Handler;                               //导入相关包
5   import android.webkit.WebSettings;
6   import android.webkit.WebView;                           //导入相关包
7   import android.widget.EditText;                          //导入相关包
8   public class Sample9_6_Activity extends Activity {
9       EditText et;                                         //声明 EditText
10      WebView wv;                                          //声明 WebView
11      Handler hd = new Handler();
12      public void onCreate(Bundle savedInstanceState){     //重写的 onCreate 方法
13         super.onCreate(savedInstanceState);
14         setContentView(R.layout.main);                    //设置当前界面
15         et = (EditText) findViewById(R.id.myEditText1);
16         wv = (WebView) findViewById(R.id.myWebView1);     //创建 WebView 对象
17         WebSettings webSettings = wv.getSettings();       //取得 WebSettings
18         webSettings.setJavaScriptEnabled(true);       //设定可执行 JavaScript
19         wv.addJavascriptInterface(this, "irdc");//设定给 html 调用的对象及名称
20         String url = "file:///android_asset/google_translate.html";
                                                    //载入 google_translate.html
21         wv.loadUrl(url);
22      }
23      public void runJavaScript(){                         //自定义方法
24          hd.post(new Runnable(){
25            public void run(){                             //重写的 run()方法
26               String str=et.getText().toString().trim(); //得到文本框内容
27              if(str.length()!=0){                         //判断长度是否为零
28                wv.loadUrl("javascript:translate('"        //调用 javascript
29                   + str+ "')");
30              }}});}}
```

其中：

- 第 2～11 行为该类相关包的导入和声明 EditText 和 WebView 引用并创建 Handler 对象。
- 第 12～22 行为获取 WebView 和 EditText 引用，取得 WebSettings，并设定可以执行 JavaScript，使用 addJavascriptInterface() 方法指定对象及其名称，最后载入 google_translate.html。
- 第 23～30 行为自定义方法，在该方法中使用 Handler 对象通过实现 Runnable()接口调用 JavaScript。

在最后要介绍的是 google_translate.html 的实现，该文件的源代码如下所列。

代码位置：见随书光盘中源代码/第 9 章/Sample9_5/assets 目录下的 google_translate.html。

```
1   <html>
2   <head>                                                <!--头文件-->
3     <meta
4     http-equiv="Content-Type"
```

```
5     content="text/html; charset=GB2312" />                    <!--编码格式-->
6   </head>
7   <script type="text/javascript" src="http://www.google.com/jsapi"><!--Google
翻译 API-->
8   </script>
9   <script type="text/javascript">
10     google.load("language", "1");
11     function mytranslate(strInput,target){                   <!--翻译方法-->
12       try{
13         google.language.translate(strInput, "", "zh-TW", function(result){
14           if (!result.error){                                <!--结果正确时-->
15             document.getElementById(target).innerHTML
16             =result.translation;
17           }
18           else{                                              <!--结果错误时-->
19             google.language.translate (strInput, "en", "zh-TW", function(result){
20               if (!result.error){                            <!--结果正确时-->
21                 document.getElementById(target).innerHTML
22                 =result.translation;
23               }
24               else{
25                 document.getElementById(target).innerHTML
26                 =result.translation;
27               }});}});}
28       catch(e){alert("Error is:"+e);}                        <!--打印异常-->
29     }
30  </script>
31  <script language="javascript">
32  function translate(strTranslate){                           <!--自定义方法-->
33     alert(strTranslate);
34     mytranslate(strTranslate, "showDiv");                    <!--调用翻译方法-->
35     alert("Done!");
36  }
37  </script>
38  <body>
39    <p align="center">                                        <!--居中-->
40    <input type="button" value="翻译" onClick="window.irdc.runJavaScript()">
41    </p>
42    <div id="showDiv"                                         <!--div 标记-->
43    style="text-align:center;font-size:18pt;
44    padding-left:5px; padding-right:5px;
45    padding-top:2px; padding-bottom:2px">
46    </div>
47  </body>
48  </html>
```

其中：

- 第 3～5 行为设置文本格式和设置编码格式。
- 第 7 行为引入 Google 翻译 API。
- 第 10～29 行为自定义一个翻译方法，在该方法中调用 Google 提供的 google.language.translate()方法，该方法中的第一个参数表示所输入的字符串，第二个参数表示输入语言，第三个参数表示输出语言，第四个参数表示一个方法，在此方法中可以对查询的结果进行处理，若查询正确则显示翻译，若不正确则将单词显示在按钮下方。
- 第 32～35 行为自定义方法，该方法主要实现的功能为调用 mytranslate()方法。
- 第 40 行为一个 Button 按钮，单击按钮时调用 window.irdc.runJavaScript()方法。
- 第 42～46 行为翻译结果显示的位置。

小结

在本章中详细介绍了 Google 为 Android 平台下的手机与服务功能。通过本章的学习，读者应该可以自行开发出简单的基于 Google 服务的小软件。在本章的学习中，笔者建议读者应该首先运行各个程序，在基本会使用后再观看相应程序的代码。

第 10 章　手机多媒体服务功能

在上一章主要介绍在 Android 手机上应用 Google 提供的服务，可以看出 Android 手机的强大功能，在本章将对手机自身的多媒体服务功能进行详细介绍。

实例 1　获取图片的宽高

手机中存储着自己喜欢的图片，每一张图片都有自己的高度和宽度，如何获取手机中图片的大小？在本节将对其进行详细介绍。

【实例描述】

本软件仅包含一个界面，加载图片资源同时在界面中显示所加载图片的宽度和高度。本实例的运行效果如图 10-1 所示。

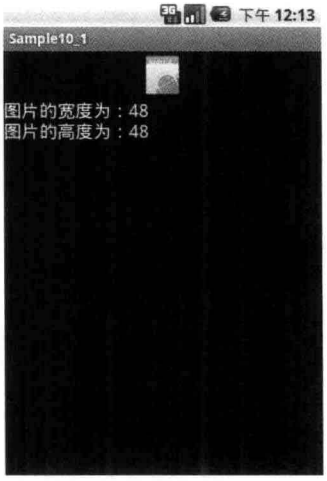


图 10-1　获取图片的宽度和高度

提示：图 10-1 显示了本软件加载图片的宽度和高度。

【实现过程】

在该实例中主要使用 Bitmap，首先创建 Bitmap 对象，然后使用 BitmapFactory.decodeResource()方法加载位图，通过 getWidth()以及 getHeight()方法获取所加载图片的高度和宽度。

【代码解析】

在本部分要介绍的是界面布局的实现文件——main.xml，详见光盘源代码。

代码位置：见光盘源代码/第 10 章/Sample10_1/ res/layout/main.xml。

提示：代码在主 LinearLayout 中设置 ImageView 和 TextView，其中 LinearLayout 包含 ImageView，并将图片设置为居中显示。

接下来要介绍的是 Sample10_1_Activity 类的设计与实现，在该类中完成了对图片的加载和图片高度宽度的获取，该类的代码如下所列。

代码位置：见光盘源代码/第 10 章/Sample10_1/src/com/bn/ex10a/Sample10_1_Activity.java。

```
1   package com.bn.ex10a;
2   ……相关包的导入参考代码部分
7   public class Sample10_1_Activity extends Activity {
8       Bitmap bm;                                                    //声明位图对象
9       TextView tv;                                                  //文本显示
10      public void onCreate(Bundle savedInstanceState) {
11          super.onCreate(savedInstanceState);
12          setContentView(R.layout.main);                            //设置当前界面
13          bm=BitmapFactory.decodeResource(this.getResources(),R.drawable.icon);
                                                                      //声明位图对象
14          tv=(TextView)this.findViewById(R.id.tv);                  //文本显示
15          String bmWidth=bm.getWidth()+"";                          //获取宽度
16          String bmHeight=bm.getHeight()+"";                        //获取高度
17          tv.setText("图片的宽度为："+bmWidth+"\n"+"图片的高度为："+bmHeight);
                                                                      //设置文本显示
18      }
19  }
```

其中：第 7～19 行主要完成位图对象的声明和 TextView 的对象声明以及图片宽度和高度的获取，其中第 8～14 行为设置 main.xml 为当前界面，初始化对象；第 15～17 行为获取图片宽度和高度，并将获取的值设置在 TextView 的文本中且显示在主界面上。

实例 2　绘制简单图形

在开发 2D 或 2.5D 游戏时往往需要自己进行图元的绘制，在 Android 手机中便可以实现上述要求，在本节将对手机中简单图形的绘制进行介绍。

【实例描述】

在该软件中主要完成的是红色矩形、半透明蓝色椭圆以及未填充椭圆的绘制。本实例的运行效果如图 10-2 所示。

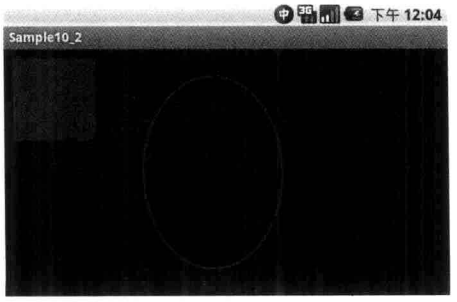


图 10-2　简单图形的绘制

提示：图 10-2 为本程序的运行效果图，其主要是绘制椭圆与正方形。

【实现过程】

图形的绘制可以通过继承并扩展 View 类实现，但从游戏的性能及开发的方便性来说，继承并扩展 SurfaceView 类效果更好。在绘制图形时需要声明 Paint 对象，指定画笔 Paint 的颜色以及绘制的形状，即可在指定位置绘制出集合图形。

绘制图形的过程中需要使用 synchronized()方法锁定画布，当此轮绘制结束后，使用 unlockCanvasAndPost()方法解锁。

【代码解析】

经过上面的理论知识的介绍后，接下来要介绍的是该软件的核心代码部分，首先介绍的是 Sample10_2_Activity 类的设计与实现，该类主要完成的是界面的设置，代码如下。

代码位置：见光盘源代码/第 10 章/Sample10_2/src/com/bn/ex10b/Sample10_2_Activity.java。

```
1   package com.bn.ex10b;                                          //包名的声明
2   import android.app.Activity;                                   //相关包的引入
3   import android.os.Bundle;                                      //相关类的引入
4   public class Sample10_2_Activity extends Activity {
5      public void onCreate(Bundle savedInstanceState) {
6           super.onCreate(savedInstanceState);                   //调用父类方法
7           MySurfaceView mySurfaceView = new MySurfaceView(this);  //创建对象
8           this.setContentView(mySurfaceView);                   //设置当前界面
9      }  }
```

提示：上述代码中主要完成创建 MySurfaceView 对象，并将其设置为主界面。

接下来要介绍的是 MySurfaceView 类，该类继承并扩展 SurfaceView 类，实现 SurfaceHolder.Callback 接口。在该类中完成图形的绘制，代码如下所列。

代码位置：见光盘源代码/第 10 章/Sample10_2src/com/bn/ex10b/MySurfaceView.java。

```
1   package com.bn.ex10b;
2       ......//该处省略了部分类的导入，读者可自行查看随书光盘中源代码
10  implements SurfaceHolder.Callback{                            //实现接口
11      Sample10_2_Activity activity;
12      Paint paint;                                               //画笔
13      public MySurfaceView(Sample10_2_Activity activity) {//重写的onCreate方法
14          super(activity);
15          this.activity = activity;                              //获取引用
16          this.getHolder().addCallback(this);                //设置生命周期回调接口
17          paint = new Paint();                                   //创建画笔
18          paint.setAntiAlias(true);                              //打开抗锯齿
19      }
20      public void onDraw(Canvas canvas){
21          //绘制红色填充矩形
22          paint.setColor(Color.RED);                             //设置画笔颜色
23          canvas.drawRect(10, 10, 100, 100, paint);              //绘制矩形
24          //绘制蓝色半透明填充椭圆形
25          paint.setARGB(128, 8, 77, 253);                        //设置画笔颜色
26          RectF r=new RectF(50,10,160,180);
```

```
27            canvas.drawOval(r, paint);                          //绘制椭圆
28            //绘制红色椭圆
29            paint.setColor(Color.RED);                          //设置画笔颜色
30            r=new RectF(150,30,300,240);
31            paint.setStyle(Style.STROKE);                       //设置模式
32            paint.setStrokeWidth(3);                            //设置线条宽度
33            canvas.drawOval(r, paint);                          //绘制空心椭圆
34        }
35        public void surfaceChanged(SurfaceHolder holder, int format, int width,int
height) {
36        }
37        public void surfaceCreated(SurfaceHolder holder) {
                                                    //重写的 SurfaceHolder 方法
38            Canvas canvas = holder.lockCanvas();                //获取画布
39            try{
40                synchronized(holder){
41                    onDraw(canvas);                             //绘制
42                }    }
43            catch(Exception e){e.printStackTrace();}            //打印异常
44            finally{
45                if(canvas != null){                             //canvas 不为空
46                    holder.unlockCanvasAndPost(canvas);         //解锁
47                }   }   }
48        public void surfaceDestroyed(SurfaceHolder holder) {
49        }   }
```

其中：

- 第 8～19 行为声明 Sample10_2_Activity 引用以及画笔，在该类的构造器内获取 Sample10_2_Activity 引用，设置生命周期回调接口；创建 Paint 对象，并打开抗锯齿。
- 第 20～34 行为 onDraw()方法的实现，在该方法中设置画笔颜色为红色，绘制矩形，随后依次完成蓝色半透明填充椭圆形的绘制和红色未填充椭圆的绘制。
- 第 35～49 行为重写 surfaceChanged()方法、surfaceCreated()方法和 surfaceDestroyed()方法，在 surfaceCreated()方法中获取画布，并将获取的画布锁定，然后绘制图形，绘制结束后解除锁定。

提示：surfaceChanged()方法在横屏和竖屏之间变化时被调用，surfaceCreated()方法在创建时被调用，surfaceDestroyed()方法在销毁时被调用。

实例 3　实现平面贴图

开发 2D 或 2.5D 游戏有时候不仅仅需要自己进行图元的绘制，还需要进行平面贴图，在 Android 手机中也可以实现平面贴图，在本节将对手机中的平面贴图进行介绍。

【实例描述】

本程序实现了手机中的平面贴图效果，在界面中包含未处理的贴图、旋转一定角度的贴图和半透明效果的贴图。

本实例的运行效果如图 10-3 所示。

提示：图 10-3 表示本程序的运行效果图，其中主要是对 2D 贴图技术的应用。

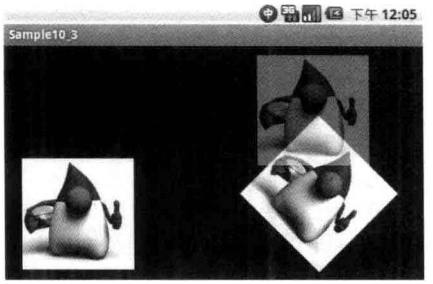


图 10-3　平面贴图效果

【实现过程】

平面贴图同样可以通过继承并扩展 SurfaceView 类来实现。在进行平面贴图时需要声明 Paint 对象，使用 Bitmap 创建位图，绘制时需要指定画笔和位图即可。图片旋转效果的实现首先要创建 Matrix 对象，然后调用 Matrix 的 setRotate()方法，半透明效果的绘制是通过设置图片的 Alpha 值确定的。

在绘制图形的过程中需要使用 synchronized()方法锁定画布，当此轮绘制结束后，使用 unlockCanvasAndPost()方法解锁。

【代码解析】

在本部分将对核心代码进行详细介绍，首先介绍 Sample10_3_Activity 类的设计与实现，在该类中主要完成的是界面的设置，代码如下。

代码位置：见光盘源代码/第 10 章/Sample10_3/src/com/bn/ex10c/Sample10_3_Activity.java。

```
1   package com.bn.ex10c;
2   import android.app.Activity;                                    //引入相关包
3   import android.os.Bundle;                                       //导入相关包
4   public class Sample10_3_Activity extends Activity { //创建继承 Activity 的类
5      public void onCreate(Bundle savedInstanceState) {      //重写的 onCreate 方法
6         super.onCreate(savedInstanceState);
7         MySurfaceView mySurfaceView = new MySurfaceView(this);      //创建对象
8         this.setContentView(mySurfaceView);                          //设置当前界面
9      }   }
```

提示：上述代码主要完成创建 MySurfaceView 对象，并将其设置为主界面。

接下来要介绍 MySurfaceView 类的设计与实现，该类继承并扩展 SurfaceView 类，实现声明周期回调接口，该类代码如下所列。

代码位置：见光盘源代码/第 10 章/Sample10_3/src/com/bn/ex10c/MySurfaceView.java。

```
1   package com.bn.ex10c;
2       ......//该处省略了部分类的导入，读者可自行查看随书光盘中源代码
9   public class MySurfaceView extends SurfaceView                  //继承 SurfaceView
10  implements SurfaceHolder.Callback{                              //实现生命周期回调接口
11      Sample10_3_Activity activity;                                //声明引用
12      Paint paint;                                                 //画笔
13      public MySurfaceView(Sample10_3_Activity activity) {     //构造器
14          super(activity);
15          this.activity = activity;                                //获取引用
16          this.getHolder().addCallback(this);                      //设置生命周期回调接口
```

```
17          paint = new Paint();                                    //创建画笔
18          paint.setAntiAlias(true);                               //打开抗锯齿
19      }
20      public void onDraw(Canvas canvas){
21          Bitmap bitmapTmp=BitmapFactory.decodeResource(
22              activity.getResources(), R.drawable.duke);          //加载图片
23          //在屏幕的 20，180 位置贴图
24          canvas.drawBitmap(bitmapTmp, 20, 130, paint);
25          //将图片旋转 45° 并移动到 200，100 位置贴图
26          Matrix m1=new Matrix();                                 //创建对象
27          m1.setTranslate(360,80);                                //移动矩阵
28          Matrix m2=new Matrix();                                 //创建对象
29          m2.setRotate(45);                                       //旋转 45°
30          Matrix mz=new Matrix();                                 //得到 Matrix 对象
31          mz.setConcat(m1, m2);
32          canvas.drawBitmap(bitmapTmp, mz, paint);                //绘制贴图
33          //改变图片透明度并在 250，10 位置贴图
34          paint.setAlpha(128);                                    //设置透明度
35          canvas.drawBitmap(bitmapTmp, 290, 10, paint);           //绘制透明效果贴图
36      }
37      public void surfaceChanged(SurfaceHolder arg0, int arg1, int arg2, int arg3) {
38      }
39      public void surfaceCreated(SurfaceHolder holder) {          //创建时被调用
40          Canvas canvas = holder.lockCanvas();                    //获取画布
41          try{
42              synchronized(holder){
43                  onDraw(canvas);                                 //绘制
44              }}
45          catch(Exception e){e.printStackTrace();}                //打印异常
46          finally{
47              if(canvas != null){
48                  holder.unlockCanvasAndPost(canvas);             //解锁
49              } } }
50      public void surfaceDestroyed(SurfaceHolder arg0) {//销毁时被调用
51      } }
```

其中：

- 第 9～19 行主要完成 Sample10_3_Activity 引用的声明和 Paint 的声明，在该类的构造器内完成 Sample10_3_Activity 引用的获取以及画笔的创建。
- 第 20～36 行为绘制三种效果的平面贴图，在 onDraw()方法中首先做的是位图的加载，然后在指定位置绘制第一幅贴图；创建矩阵对象，通过移动和旋转后绘制第二幅贴图；设置图片的 Alpha 值，绘制第三幅贴图。
- 第 37～51 行主要完成在创建 MySurfaceView 对象时获取画布并锁定画布后绘制贴图，绘制完成后解除锁定。

实例 4　简单淡入淡出效果

游戏的开始界面经常会出现淡入淡出效果的界面，这样的效果大大增加了游戏的吸引力，在 Android 手机中实现淡入淡出的效果是很简单的，在本节将对其进行详细介绍。

【实例描述】

运行软件时，首先进入的是程序主界面，该界面由亮变暗，随后出现 LOGO 界面，该界面

同样是慢慢变暗，最后消失。本实例的运行效果如图 10-4 所示。

图 10-4　淡入淡出效果图

提示：在图 10-4 中依次表示的是开始时的主界面，以及在上一界面完全淡化后进入的界面。

【实现过程】

在该软件中使用 setRequestedOrientation()方法将屏幕设置为横屏，其参数设置为 ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE。使用 requestWindowFeature()方法设置为全屏，通过 DisplayMetrics 获取手机的分辨率。

绘制图片时首先通过 BitmapFactory.decodeResource()方法加载位图，在 surfaceCreated()方法中设定线程动态改变图片的 Alpha 值，改变图片的透明度，从而实现图片淡淡消失的效果。

【代码解析】

在本部分将对其核心代码进行介绍，首先介绍 Sample10_4_Activity 类的设计与实现，在该类中主要完成横屏与全屏的设置，并获取手机屏幕的分辨率，代码如下所列。

代码位置：见光盘源代码/第 10 章/Sample10_4/src/com/bn/ex10d/Sample10_4_Activity.java。

```
1   package com.bn.ex10d;
2       ......//该处省略了部分类的导入，读者可自行查看随书光盘中源代码
8   public class Sample10_4_Activity extends Activity {
9       static float screenHeight;                              //屏幕高度
10      static float screenWidth;                               //屏幕宽度
11      StartView startView;
12      public void onCreate(Bundle savedInstanceState) {
13          super.onCreate(savedInstanceState);
14          this.setRequestedOrientation(                       //设置为横屏
15              ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
16          requestWindowFeature(Window.FEATURE_NO_TITLE);  //设置全屏
17          getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
18                               WindowManager.LayoutParams.FLAG_FULLSCREEN);
19          DisplayMetrics dm=new DisplayMetrics();             //创建对象
20          getWindowManager().getDefaultDisplay().getMetrics(dm);
21          screenHeight=dm.heightPixels;                       //屏幕高度
22          screenWidth=dm.widthPixels;                         //屏幕宽度
23          startView=new StartView(this);                      //开始界面对象
24          this.setContentView(startView);                     //设置主界面
25      }   }
```

其中：

- 第 9～25 行为声明成员变量和 StartView 引用，在 onCreate()方法中设置为横屏和全屏显示，创建 DisplayMetrics 对象，获取该手机屏幕的高度和宽度，最后创建 StartView 对象并将主界面设置为 StartView 对象。

上述代码介绍本程序的 Activity 类的实现，接下来介绍 StartView 类的设计与实现，在该类中主要完成的是淡入淡出的效果的实现，代码如下所列。

代码位置：见光盘源代码/第 10 章/Sample10_4/src/com/bn/ex10d/StartView.java。

```
1   package com.bn.ex10d;
2       ......//该处省略了部分类的导入，读者可自行查看随书光盘中源代码
9   public class StartView extends SurfaceView          //继承 SurfaceView
10  implements SurfaceHolder.Callback{                  //实现生命周期回调接口
11      Sample10_4_Activity activity;                    //声明引用
12      Paint paint;                                     //声明画笔
13      int currentAloha=0;                              //当前的透明度
14      int screenWidth=(int)Sample10_4_Activity.screenWidth;    //屏幕宽度
15      int screenHeight=(int)Sample10_4_Activity.screenHeight; //屏幕高度
16      int sleepSpan=50;                                        //线程睡眠时间
17      Bitmap[] logos=new Bitmap[2];                            //位图数组
18      int pic1;
19      int pic2;
20      Bitmap currentLogo;                              //当前 logo 图片引用
21      int currentX;                                    //图片位置
22      int currentY;
23      public StartView(Sample10_4_Activity activity) {
24          super(activity);
25          this.activity=activity;                      //获取引用
26          this.getHolder().addCallback(this);          //设置生命周期回调接口
27          paint=new Paint();                           //创建画笔
28          paint.setAntiAlias(true);                    //打开抗锯齿
29          pic1=R.drawable.bg0;                         //图片编号
30          pic2=R.drawable.bg1;
31          logos[0]=BitmapFactory.decodeResource(activity.getResources(),
pic1);//加载位图
32          logos[1]=BitmapFactory.decodeResource(activity.getResources(),
pic2);//加载位图
33      }
34      public void onDraw(Canvas canvas){
35          try{
36              paint.setColor(Color.BLACK);             //设置画笔颜色
37              paint.setAlpha(255);                     //设置透明度
38              canvas.drawRect(0, 0,screenWidth, screenHeight ,paint);
                                                         //绘制黑色填充矩形全背景
39              if(currentLogo==null)return;             //进行平面贴图
40              paint.setAlpha(currentAloha);            //设置 Alpha
41              canvas.drawBitmap(currentLogo, 0, 0,paint);     //绘制图片
42          }
43          catch(Exception e){e.printStackTrace();}     //打印异常
44      }
45      public void surfaceChanged(SurfaceHolder arg0, int arg1, int arg2, int arg3) {
46      }
47      public void surfaceCreated(SurfaceHolder arg0) {     //创建时被调用
48          new Thread(){
49              public void run(){
50                  for(Bitmap bm:logos){                //扫描位图
51                      currentLogo=bm;
```

```
52                          //图片的位置
53                          currentX=screenWidth/2-bm.getWidth()/2;      //X 坐标位置
54                          currentY=screenHeight/2-bm.getHeight()/2;    //Y 坐标位置
55                          for(int i=255;i>-10;i=i-10) { //动态更改图片的透明度值
56                              currentAloha=i;
57                              if(currentAloha<0){
58                                  currentAloha=0;                      //归零
59                              }
60                              SurfaceHolder myholder=StartView.this.getHolder();
61                              Canvas canvas = myholder.lockCanvas(); //获取画布
62                              try{
63                                  synchronized(myholder){
64                                      onDraw(canvas);                  //绘制
65                                  }
66                              }catch(Exception e){e.printStackTrace();}
                                                                         //捕获异常
67                              finally{
68                                  if(canvas !=null){                   //若不为空
69                                      myholder.unlockCanvasAndPost(canvas);
70                                  }}
71                              try{
72                                  if(i==255){
73                                      Thread.sleep(1000);   //使线程睡眠 1 秒钟
74                                  }
75                                  Thread.sleep(sleepSpan);  //捕获异常
76                              }catch(Exception e){e.printStackTrace();} //打印异常
77                          }}}}.start();
78      }
79      public void surfaceDestroyed(SurfaceHolder arg0) {   //销毁时被调用
80      }}
```

其中：

- 第 9～33 行主要为成员变量的声明和构造器的创建，在构造器内获取 Sample10_4_Activity 引用，设置生命周期回调接口，创建 Paint 对象并设置画笔打开抗锯齿，最后通过 BitmapFactory.decodeResource 方法加载位图。
- 第 34～44 行主要完成贴图的绘制，首先绘制屏幕大小的黑色背景，然后根据不同 Alpha 值绘制不同透明的图片，实现图片渐渐消失的效果。
- 第 47～78 行主要完成在创建该类对象时启动线程改变图片的 Alpha 值。在线程内扫描 Bitmap 数组，使用 for 循环动态更改图片的透明度值。绘制图片时首先获取画布，对画布进行锁定，绘制结束后解除锁定再进行下一次绘制。

实例 5　虚拟键的设计与实现

对于没有键盘的手机在进行一些操作时会十分不方便，虚拟键的设置可以解决以上问题，在本节将对虚拟键的设计和实现进行介绍。

【实例描述】

该软件实现的功能简单，在主界面中包含向上和向下两个按钮，一张图片。当单击向上按钮时图片向上移动，单击向下按钮时图片向下移动。

本实例的运行效果如图 10-5 所示。

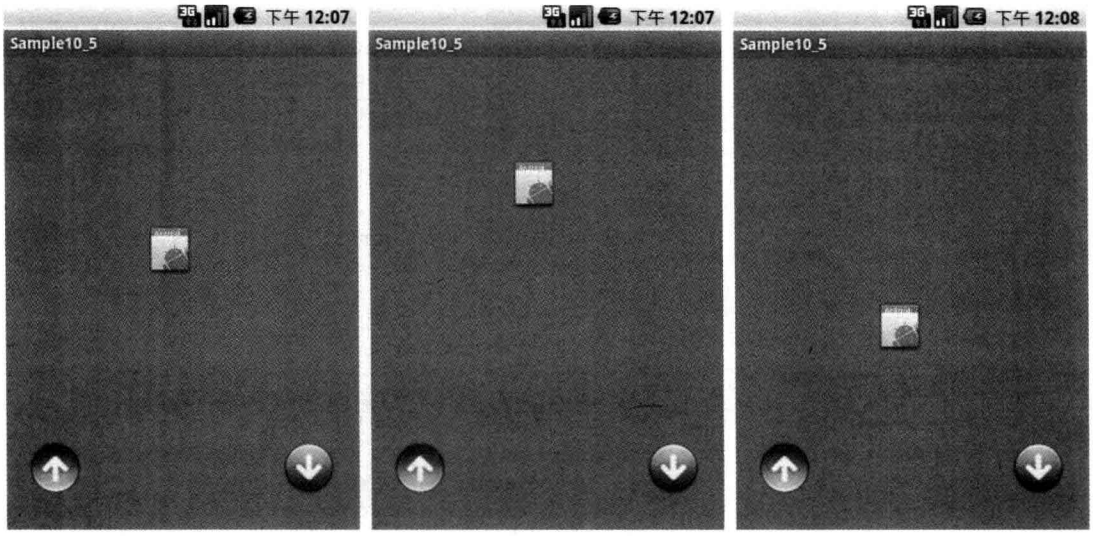


图 10-5　虚拟键的使用

提示：在图 10-5 中依次表示的是图片的初始位置，以及在界面单击向上的按钮图标图片向上移动，单击向下的按钮图标图片向下移动的效果。

【实现过程】

在本软件中主要运用 onTouchEvent 方法的实现，其中 MotionEvent.ACTION_DOWN 表示键按下时的事件，MotionEvent.ACTION_UP 表示键弹起时的事件，MotionEvent.ACTION_MOVE 表示键按下后移动时的事件。

【代码解析】

经过上面的理论介绍，在此要介绍的是核心代码，首先介绍 Sample10_5_Activity 类的设计与实现，在该类中主要完成创建 MySurfaceview 对象，并将其设置为当前界面。

代码位置：见光盘源代码/第 10 章/Sample10_5/src/com/bn/ex10e/Sample10_5_Activity.java。

```
1   package com.bn.ex10e;                                          //声明包
2   import android.app.Activity;                                   //导入相关包
3   import android.os.Bundle;                                      //导入相关包
4   public class Sample10_5_Activity extends Activity { //创建继承 Activity 的类
5       public void onCreate(Bundle savedInstanceState) {     //重写 onCreate 方法
6           super.onCreate(savedInstanceState);
7           MySurfaceview mySurfaceView = new MySurfaceview(this);   //创建对象
8           this.setContentView(mySurfaceView);                    //设置当前界面
9       }}
```

提示：上述代码中主要完成相关包的导入以及界面对象的创建，并将其设置为当前界面。

上述代码介绍的是 Activity 类的开发，接下来介绍 MySurfaceview 类的设计与实现，在该类中主要完成虚拟键和图片的绘制，在该类中重写 onTouchEvent 方法，实现虚拟键的功能，代

码如下。

代码位置：见光盘源代码/第 10 章/Sample10_5/src/com/bn/ex10e/MySurfaceview.java。

```
1   package com.bn.ex10e;
2       ......//该处省略了部分类的导入，读者可自行查看随书光盘中源代码
9   import android.view.SurfaceView;
10  public class MySurfaceview extends SurfaceView          //继承 Surfaceview
11  implements SurfaceHolder.Callback{                      //实现生命周期回调接口
12      Sample10_5_Activity myActivity;
13      Paint paint;                                         //声明画笔
14      Bitmap down;                                         //向上按钮
15      Bitmap up;                                           //向下按钮
16      Bitmap duke;                                         //duke 图片
17      boolean flag;                                        //线程标志位
18      float tempX;                                         //tempX
19      float tempY;                                         //tempY
20      public MySurfaceview(Sample10_5_Activity myActivity) {
21          super(myActivity);
22          this.myActivity=myActivity;                      //获取引用
23          this.getHolder().addCallback(this);              //设置生命周期回调接口
24          paint=new Paint();
25          paint.setAntiAlias(true);                        //打开抗锯齿
26          down=BitmapFactory.decodeResource(
27              myActivity.getResources(), R.drawable.down);     //向下按钮图片
28          up=BitmapFactory.decodeResource(
29              myActivity.getResources(), R.drawable.up);       //向上按钮图片
30          duke=BitmapFactory.decodeResource(
31              myActivity.getResources(), R.drawable.icon);     //duke 图片
32          flag=true;                                       //标志位设为 true
33      }
34      public boolean onTouchEvent(MotionEvent event){
35          switch(event.getAction()){
36              case MotionEvent.ACTION_DOWN:                //键按下事件
37                  float x=event.getX();                    //获取 X 轴坐标
38                  float y=event.getY();                    //获取 Y 轴坐标
39                  if(x>20&&x<65&&y>350&&y<395){            //向上按钮范围
40                      tempY-=2f;
41                      repaint();                           //重绘画面
42                  }
43                  if(x>250&&x<295&&y>350&&y<395){          //向下按钮范围
44                      tempY+=2f;
45                      repaint();                           //重绘画面
46                  }
47              break;
48          }
49          return true;                                     //返回 true
50      }
51      public void onDraw(Canvas canvas){
52          canvas.drawColor(Color.GRAY);                    //设置背景色
53          canvas.drawBitmap(up, 20,350, paint);            //绘制向上图片
54          canvas.drawBitmap(down, 250,350, paint);         //绘制向下图片
55          canvas.drawBitmap(duke, 125, 150+tempY, paint);  //duke
56      }
57      public void surfaceChanged(SurfaceHolder holder, int format, int width,int
height) {
58      }
59      public void surfaceCreated(SurfaceHolder holder) {
60          Canvas canvas = holder.lockCanvas();             //获取画布
61          try{
```

```
62                synchronized(holder){
63                    onDraw(canvas);                          //绘制
64                }}
65            catch(Exception e){
66                e.printStackTrace();                         //打印异常
67            }
68            finally{
69                if(canvas != null){                          //判断 canvas 是否为空
70                    holder.unlockCanvasAndPost(canvas);      //解锁
71                }}}
72        public void surfaceDestroyed(SurfaceHolder holder) {     //重写方法
73        }
74        public void repaint(){                               //声明方法
75            SurfaceHolder holder=this.getHolder();           //获取 SurfaceHolder
76            Canvas canvas = holder.lockCanvas();             //获取画布
77            try{
78                synchronized(holder){
79                    onDraw(canvas);                          //绘制
80                }}
81            catch(Exception e){                              //捕获异常
82                e.printStackTrace();                         //打印异常
83            }
84            finally{
85                if(canvas != null){
86                    holder.unlockCanvasAndPost(canvas);      //解除锁定
87                }}}}
```

其中：

- 第 10～33 行为声明成员变量和创建构造器。在构造器中获取 Activity 引用，设置生命周期回调接口，创建 Paint 对象并打开抗锯齿；加载向上、向下和 duke 图片。
- 第 34～50 行为重写 onTouchEvent()，在该方法中判断事件是否为 MotionEvent.ACTION_DOWN，若是该事件则判断触摸的位置是否在规定的范围内，若在范围内则执行改变图片 Y 坐标的代码，重绘画面实现图片的向上或向下的移动。
- 第 51～56 行为重写 onDraw()方法，在该方法中按照指定坐标依次绘制向上、向下和 duke 图片。
- 第 57～73 行为重写 surfaceChanged()、surfaceCreated()和 surfaceDestroyed()方法，其中在 surfaceCreated()方法中获取画布并对其进行锁定，然后绘制画面，绘制结束后解除对画布的锁定。
- 第 74～87 行为自定义 repaint()方法，在该方法中首先获取 Holder 对象，然后获取画布，对其进行锁定后绘制画面，绘制结束后解除对画布的锁定。

实例 6　获取手机内置媒体图片

在手机菜单中选择多媒体图标可以进入多媒体库查看图片等资源，在 Android 平台下同样可以自定义进行多媒体库的访问，在本节将对其进行详细介绍。

【实例描述】

在本软件中的主界面仅含有一个 Button 按钮，当单击按钮时进入手机的媒体图片库中，在图片库中选取一幅图片时，系统将自动返回到主界面，并在主界面显示所选择的图片。

本实例的运行效果如图 10-6 所示。

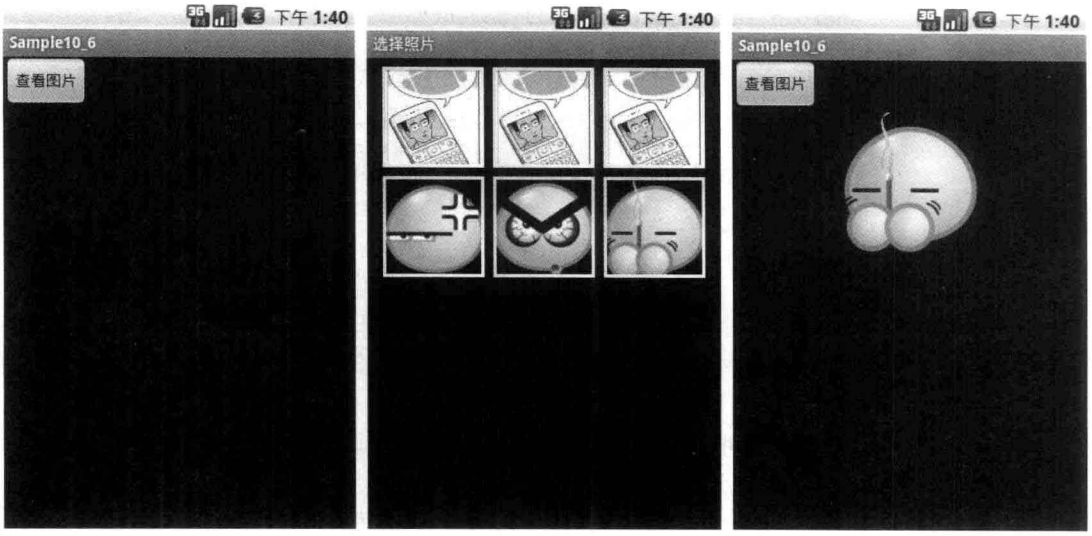


图 10-6　运行效果图

提示：在图 10-6 中左图为软件运行时的主界面，中图表示访问手机的图片媒体库，右图为选择其中一幅图片后将其显示在主界面上。

【实现过程】

打开手机中图片库的关键在于使用 Intent.setType()方法，在此方法中必须将类型设置为 image/*才可以打开手机多媒体库中的相片集。使用 startActivityForResult()方法发送广播，当在图片集中选中一幅图片时，系统自动返回至主界面。

为了使系统能够显示图片，需要在 Sample10_6_Activity 类中重写 onActivityResult()方法，判断 resultCode 标识码，通过 Intent.getData()获取 Uri 对象，将其加载为 Bitmap 类型后将其设置在 ImageView 中，显示在屏幕上。

【代码解析】

经过上面的理论介绍后，下面要介绍的是核心代码部分，在此部分首先介绍 main.xml 文件的设计与实现，该文件代码如下所列。

代码位置：见光盘源代码/第 10 章/Sample10_6/ res/layout/main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>      <!--版本号 ，编码方式-->
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3                 android:orientation="vertical"
4                 android:layout_width="fill_parent"
5      android:layout_height="fill_parent"
6      >                                          <!-- LinearLayout 的属性设置-->
7       <Button
8                     android:text="查看图片"
9                     android:id="@+id/Button01"
10                    android:layout_width="wrap_content"
11          android:layout_height="wrap_content">  <!--控件大小-->
12      </Button>                                  <!--Button 按钮的属性设置-->
13      <LinearLayout
14          android:orientation="vertical"
```

```
15          android:layout_width="fill_parent"
16          android:layout_height="wrap_content"
17      android:gravity="center">                <!--LinearLayout 的属性设置-->
18          <ImageView
19                      android:id="@+id/ImageView01"
20                      android:layout_width="wrap_content"
21              android:layout_height="wrap_content"
22              >                                <!--ImageView 的属性设置-->
23          </ImageView>
24      </LinearLayout>                          <!-- LinearLayout 的属性设置-->
25  </LinearLayout>
```

其中：

- 第 7～12 行为 Button 按钮的属性设置，将其显示文本设置为“查看图片”，大小为包裹内容。
- 第 13～17 行为内层 LinearLayout 的属性设置，该 LinearLayout 在宽度上充满屏幕，高度上包裹内容，并将其内部控制设置为居中。
- 第 18～23 行为 ImageView 的属性设置，设置其 ID 和大小为包裹内容。

接下来要介绍的是 Sample10_6_Activity 类的设计与实现，在该类中通过 Intent 发送 Intent.ACTION_GET_CONTENT 消息打开媒体图片库，查看手机中的图片，该类代码如下所列。

代码位置：见光盘源代码/第 10 章/Sample10_6/src/com/bn/ex10f/Sample10_6_Activity.java。

```
1   package com.bn.ex10f;
2   import android.app.Activity;                                    //导入相关包
3   import android.content.ContentResolver;                         //相关包的引入
4   import android.content.Intent;                                  //相关包的引入
5   import android.graphics.Bitmap;                                 //相关包的引入
6   import android.graphics.BitmapFactory;                          //相关包的引入
7   import android.net.Uri;
8   import android.os.Bundle;                                       //相关包的引入
9   import android.view.View;                                       //相关包的引入
10  import android.view.View.OnClickListener;
11  import android.widget.Button;                                   //相关包的引入
12  import android.widget.ImageView;
13  public class Sample10_6_Activity extends Activity { //创建继承 Activity 的类
14     public void onCreate(Bundle savedInstanceState) {            //重写方法
15        super.onCreate(savedInstanceState);
16        setContentView(R.layout.main);                            //设置当前界面
17        Button button=(Button)this.findViewById(R.id.Button01);    //查看按钮
18        button.setOnClickListener(                                //添加监听器
19          new OnClickListener(){
20                  public void onClick(View v) {
21                      Intent intent=new Intent();          //创建 Intent 对象
22                      intent.setType("image/*");                //设置 type
23                      intent.setAction(Intent.ACTION_GET_CONTENT);//调用 Action
24                      startActivityForResult(intent,1);  //发送 Intent 消息
25                  } } ); }
26     public void onActivityResult(int requestCode,int resultCode,Intent data){
27      if(resultCode==RESULT_OK){                                  //若为 RESULT_OK
28          Uri uri=data.getData();                                 //获取 Uri
29          ContentResolver cr=this.getContentResolver();
30          try{
31              Bitmap bitmap=BitmapFactory.decodeStream(cr.openInputStream
(uri));  //加载图片
32              ImageView
imageView=(ImageView)this.findViewById(R.id.ImageView01);//创建对象
```

```
33                imageView.setImageBitmap(bitmap);                    //显示图片
34            }catch(Exception e){e.printStackTrace();}                //打印异常
35        }
36        super.onActivityResult(requestCode, resultCode, data);       //调父类方法
37    }}
```

其中：

- 第 14～25 行为重写 onCreate()方法，在该方法中首先将 main.xml 设置为当前界面，创建 Button 对象，并为按钮添加监听器，在监听器内创建 Intent 对象，设置 Intent 后使用 startActivityForResult()方法发送 Intent 消息。
- 第 26～37 行为重写 onActivityResult()方法，判断返回的数据是否为 RESULT_OK，若满足条件，则获取 Uri 后加载图片，并将图片显示在主界面上。

实例 7　手机音量大小的调节

在实际生活中有时需要根据实际情况设置手机铃声的大小或调整手机的声音模式。在 Android 平台上提供了能够改变手机声音的解决方案，在本节将对其进行详细介绍。

【实例描述】

在本软件的主界面中包含两个图片按钮（向上按钮表示增大声音，向下按钮表示减小声音）、用于显示当前音量大小的进度条和用于表示当前声音模式的图片提示。

当单击向上图片按钮时，手机的声音增大；当单击向下图片按钮时，手机的声音减少，当减少为 0 时，手机自动将声音模式设置为震动模式，当继续减少时则将声音模式设置为静音模式。

本实例的运行效果如图 10-8 所示。

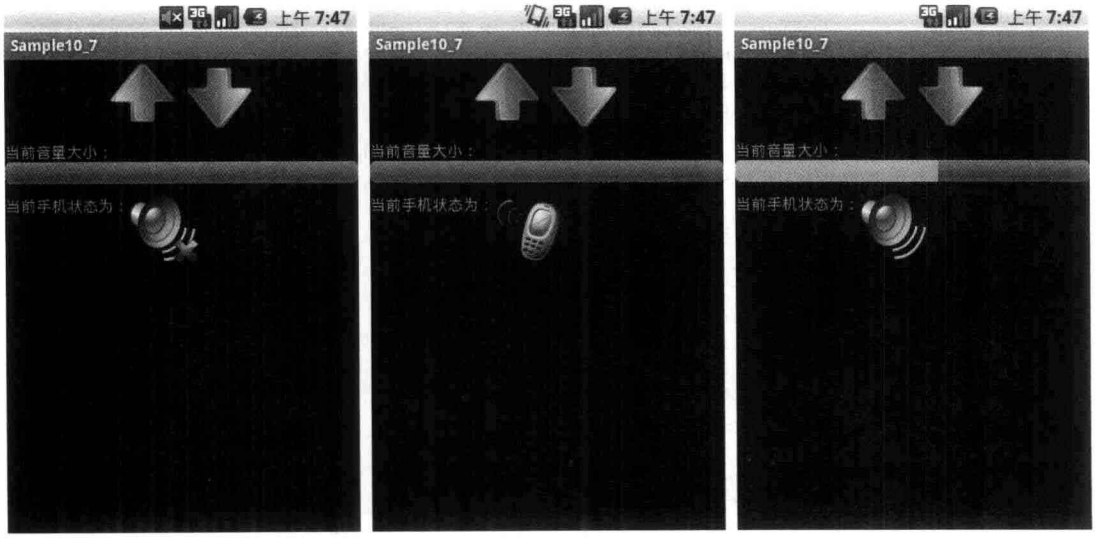


图 10-8　手机音量大小的调节

提示：在图 10-8 中，左图为将手机调成静音模式，中图为将手机调整为震动模式，右图为调节手机为正常模式，并且设定声音的大小。

【实现过程】

本软件是通过 getSystemService 取得 AudioManager 对象的，通过该对象的 adjustVolume() 方法来调整音量的大小，其中 AudioManager.ADJUST_RAISE 表示增大音量，AudioManager.ADJUST_LOWER 表示减小音量。

随着音量的增大和减少，手机的声音模式可能也会发生变化，这就需要 am.getRingerMode() 方法获取当前声音模式，并设置相应的图片显示在主界面上。

【代码解析】

经过上面的理论介绍，在本部分将对其核心代码进行详细介绍，首先介绍 main.xml 文件的设计与实现，在该文件中主要完成的是主界面布局的设置，代码如下。

代码位置：见光盘源代码/第 10 章/Sample10_7/ res/layout/main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>          <!--版本号和编码方式-->
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3                 android:orientation="vertical"
4                 android:layout_width="fill_parent"
5                 android:layout_height="fill_parent"
6       >                                           <!--LinearLayout 的属性设置-->
7        <LinearLayout
8                 android:id="@+id/LinearLayout01"
9                 android:layout_width="fill_parent"
10                android:layout_height="wrap_content"
11                android:orientation="horizontal"
12                android:gravity="center">         <!--LinearLayout 的属性设置-->
13          <ImageButton
14                android:id="@+id/ImageView01"
15                android:layout_width="wrap_content"
16                android:layout_height="wrap_content"
17                android:background="@drawable/up"> <!--ImageButton 的属性设置-->
18          </ImageButton>
19          <ImageButton
20                android:id="@+id/ImageView02"
21                android:layout_width="wrap_content"
22                android:layout_height="wrap_content"
23                android:background="@drawable/down">
                                                    <!--ImageButton 的属性设置-->
24          </ImageButton>
25      </LinearLayout>
26      <LinearLayout
27                android:id="@+id/LinearLayout02"
28                android:layout_width="fill_parent"
29                android:layout_height="wrap_content"
30                android:orientation="vertical"
31                android:paddingTop="10dip">       <!--LinearLayout 的属性设置-->
32          <TextView
33                    android:text="当前音量大小："
34                    android:id="@+id/TextView01"
35                    android:layout_width="wrap_content"
36                    android:layout_height="wrap_content"> <!--TextView 的大小-->
37          </TextView>                             <!--TextView 的属性设置-->
38          <ProgressBar
39                    android:id="@+id/ProgressBar01"
40                    android:layout_width="fill_parent"
41                    android:layout_height="wrap_content"
42                style="@android:style/Widget.ProgressBar.Horizontal">
43          </ProgressBar>                          <!--ProgressBar 的属性设置-->
```

```
44      </LinearLayout>                                  <!--LinearLayout 的属性设置-->
45      <LinearLayout                                    <!--LinearLayout 的属性设置-->
46              android:id="@+id/LinearLayout02"
47              android:layout_width="fill_parent"
48              android:layout_height="wrap_content"
49              android:orientation="horizontal"
50              android:paddingTop="10dip">              <!--LinearLayout 的属性设置-->
51          <TextView
52                  android:text="当前手机状态为："
53                  android:id="@+id/TextView02"
54                  android:layout_width="wrap_content"
55                  android:layout_height="wrap_content"> <!--TextView 的大小-->
56          </TextView>                                  <!--TextView 的属性设置-->
57          <ImageView
58                  android:id="@+id/ImageView03"
59                  android:layout_width="wrap_content"
60                  android:layout_height="wrap_content">
61          </ImageView>                                 <!--ImageButton 的属性设置-->
62      </LinearLayout>                                  <!--LinearLayout 的属性设置-->
63  </LinearLayout>
```

其中：

- 第 2～6 行为该界面最外层 LinearLayout 的属性设置，将该 LinearLayout 设置为充满整个手机屏幕，并将其中的控件设置为竖直摆放。
- 第 7～25 行为该界面内层 LinearLayout 的属性设置，将该 LinearLayout 在宽度上设置为充满这个屏幕，高度上设置为包裹内容。在该 LinearLayout 中添加两个 ImageButton，分别为其设定为向上图片和向下图片，大小均为包裹内容。
- 第 26～44 行为该界面内层 LinearLayout 的属性设置，将该 LinearLayout 在宽度上设置为充满这个屏幕，高度上设置为包裹内容。在该 LinearLayout 中添加 TextView 和 ProgressBar，分别设置其属性，其中将 ProgressBar 的类型设置为 Widget.ProgressBar.Horizontal。
- 第 45～62 行为该界面内层 LinearLayout 的属性设置，将该 LinearLayout 在宽度上设置为充满这个屏幕，高度上设置为包裹内容。在该 LinearLayout 中添加 TextView 和 ImageView，分别设置其属性，大小均为包裹内容。

上述代码介绍本程序的主界面的搭建，接下来介绍 Sample10_7_Activity 类的设计与实现，在该类中主要完成获取手机当前的模式，为 ImageButton 添加监听器，完成软件中的功能，代码如下。

代码位置：见光盘源代码/第 10 章/Sample10_7/src/com/bn/ex10g/Sample10_7_Activity.java。

```
1   package com.bn.chap10.sound;                          //声明包
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码
3   public class Sample10_7_Activity extends Activity {
4       AudioManager am;                                  //声明 AudioManger 引用
5       int volume;                                       //声明成员变量
6      public void onCreate(Bundle savedInstanceState) {
7         super.onCreate(savedInstanceState);            //调用父类
8         setContentView(R.layout.main);                 //设置当前界面
9         am=(AudioManager)getSystemService(Context.AUDIO_SERVICE);  //创建对象
10        ImageButton up=(ImageButton)this.findViewById(R.id.ImageView01);
                                                          //增加声音
11        ImageButton down=(ImageButton)this.findViewById(R.id.ImageView02);
                                                          //降低声音
12       final ProgressBar pb=(ProgressBar)this.findViewById(R.id.ProgressBar01);
```

```
                                                          //进度条
13        final ImageView iv=(ImageView)this.findViewById(R.id.ImageView03);
                                                          //ImageView 对象
14        volume=am.getStreamVolume(AudioManager.STREAM_RING);   //当前声音大小
15        pb.setMax(70);                                  //设置最大值
16        pb.setProgress(volume*10);                      //设置进度条显示
17        int mode=am.getRingerMode();
18        if(mode==AudioManager.RINGER_MODE_NORMAL){      //正常模式
19          iv.setImageDrawable(getResources().
20                  getDrawable(R.drawable.ring));        //设置图片
21        }else if(mode==AudioManager.RINGER_MODE_SILENT){ //静音模式
22          iv.setImageDrawable(getResources().
23                  getDrawable(R.drawable.jing));        //设置图片
24        }else if(mode==AudioManager.RINGER_MODE_VIBRATE){ //震动模式
25          iv.setImageDrawable(getResources().
26                  getDrawable(R.drawable.zhen));        //设置图片
27        }
28        up.setOnClickListener(                          //声音调大
29          new OnClickListener(){                        //匿名内部类
30                  public void onClick(View v) {         //重写的方法
31                      am.adjustVolume(AudioManager.ADJUST_RAISE, 0);
32                      volume=am.getStreamVolume(AudioManager.STREAM_RING);
33                      pb.setProgress(volume*10);        //进度条显示值
34                      int mode=am.getRingerMode();
35                      if(mode==AudioManager.RINGER_MODE_NORMAL){ //正常模式
36                      iv.setImageDrawable(getResources().
37                              getDrawable(R.drawable.ring));    //设置图片
38                      }else if(mode==AudioManager.RINGER_MODE_SILENT){
                                                          //静音模式
39                      iv.setImageDrawable(getResources().
40                              getDrawable(R.drawable.jing));    //设置图片
41                      }else if(mode==AudioManager.RINGER_MODE_VIBRATE){
                                                          //震动模式
42                      iv.setImageDrawable(getResources().
43                              getDrawable(R.drawable.zhen));    //设置图片
44                  }});
45        down.setOnClickListener(                        //声音调小
46          new OnClickListener(){                        //匿名内部类
47                  public void onClick(View v) {         //重写的方法
48                      am.adjustVolume(AudioManager.ADJUST_LOWER, 0);
49                      volume=am.getStreamVolume(AudioManager.STREAM_RING);
50                      pb.setProgress(volume*10);        //设置进度条
51                      int mode=am.getRingerMode();
52                    if(mode==AudioManager.RINGER_MODE_NORMAL){    //正常模式
53                      iv.setImageDrawable(getResources().
54                              getDrawable(R.drawable.ring));    //设置图片
55                    }else if(mode==AudioManager.RINGER_MODE_SILENT){
                                                          //静音模式
56                      iv.setImageDrawable(getResources().
57                              getDrawable(R.drawable.jing));    //设置图片
58                    }else if(mode==AudioManager.RINGER_MODE_VIBRATE){ //震动模式
59                      iv.setImageDrawable(getResources().
60                              getDrawable(R.drawable.zhen));    //设置图片
61    }});}}
```

其中：

- 第 7～27 行主要完成的是设置当前界面，创建 AudioManager 对象，以及创建界面中控

件的对象。通过 AudioManager 对象获取当前声音的大小，将获取的值赋予 ProgressBar。根据 AudioManager. getRingerMode()方法获取当前的手机模式，并根据模式设置显示在主界面上的图片。

- 第 28～44 行为向上 ImageButton 按钮的监听器，在该监听器中每次单击时调用 am.adjustVolume()方法，并将其中的第一个参数设置为 AudioManager.ADJUST_RAISE，为手机增加一格音量，并将增加后的音量赋值给 ProgressBar，并时刻判断当前手机的模式，根据模式的不同设置不同的提示图片。
- 第 45～61 行为向下 ImageButton 按钮的监听器，在该监听器中每次单击时调用 am.adjustVolume()方法，同样将其中的第一个参数设置为 AudioManager.ADJUST_LOWER，使手机减少一格音量，将减少后的音量赋值给 ProgressBar，并时刻判断当前手机的模式，根据模式的不同设置不同的提示图片。

实例 8　采集音频数据

多媒体数据采集包含音频数据的采集，而且音频数据的采集是每个手机都含有的一项功能，在本节将介绍如何在 Android 手机中自定义一个用于音频数据采集的软件。

【实例描述】

该软件的主界面设计十分简单，在主界面中包含开始按钮和停止按钮，当单击开始按钮时系统将自动检测手机中是否存在 SD 卡，若存在则开始正常录音，反之则弹出消息对用户进行提示。单击停止按钮时，系统停止录音，并将录制的音频存放到 SD 卡中。

本实例的运行效果如图 10-8 所示。

图 10-8　音频的采集

提示：在图 10-8 中从左至右依次表示的是软件开始运行时的初始界面，以及单击“录音”按钮以录制声音。

【实现过程】

在该软件的实现中主要运用了 MediaRecorder，使用 MediaRecorder 设置录制声音的设备、格式和编码器，在录音前使用 Environment.getExternalStorageState()判断手机中是否存在 SD 卡，若不存在则拒绝音频的录制。

设定 Bundle 绑定发送的消息，通过 Handler 发送以及接收消息改变用于显示录制时长的 TextView 的文本。

【代码解析】

经过前面的理论介绍，接下来要介绍的是 main.xml 文件和录音功能的实现类 Sample10_8_Activity。首先要介绍的是界面布局 main.xml 的实现，代码如下。

代码位置：见光盘源代码/第 10 章/Sample10_8/ res/layout/main.xml。

```
<?xml version="1.0" encoding="utf-8"?>          <!--版本号和编码方式-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:orientation="vertical"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
    >                                           <!--LinearLayout 的属性设置-->
    <LinearLayout
              android:orientation="horizontal"
              android:id="@+id/LinearLayout01"
              android:layout_width="fill_parent"
      android:layout_height="wrap_content">     <!--LinearLayout 的大小-->
     <TextView
              android:id="@+id/TextView01"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
          android:text="录制时长 "
              android:textColor="@color/white"
              android:textSize="20dip"
     />                                         <!--TextView 的属性设置-->
     <TextView
              android:id="@+id/TextView02"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="0m:0s"
              android:textColor="@color/white"
              android:textSize="20dip"
     />                                         <!--TextView 的属性设置-->
    </LinearLayout>
    <LinearLayout
      android:orientation="horizontal"
      android:id="@+id/LinearLayout02"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content">     <!--LinearLayout 的大小-->
      <ImageButton
           android:id="@+id/ImageButton01"
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:src="@drawable/record"
          >                                     <!--ImageButton 的属性设置-->
          </ImageButton>
          <ImageButton
            android:id="@+id/ImageButton02"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/stop"
          >
          </ImageButton>                        <!--ImageButton 的属性设置-->
    </LinearLayout>                             <!--LinearLayout 的属性设置-->
</LinearLayout>
```

其中：

- 第 7～28 行为界面中内层 LinearLayout 的属性设置，在此 LinearLayout 中添加两个 TextView，分别用于显示提示录制时长和显示时间。
- 第 29～48 行为界面中内层 LinearLayout 的属性设置，代码中将 LinearLayout 宽度设置为充满整个屏幕，高度设置为包裹内容，并在此 LinearLayout 中添加两个 ImageButton。

接下来要介绍的是 Sample10_8_Activity 类的设计与实现，在该类中主要完成音频的录制功能，在详细介绍该类之前，首先来了解一下该类的总体框架，代码如下。

代码位置：见光盘源代码/第 10 章/Sample10_8/src/com/bn/ex10h/Sample10_8_Activity.java。

```
1   package com.bn.chap10.recorder;                                    //声明包
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码
3   public class Sample10_8_Activity extends Activity implements OnClickListener{
4       public static final int UPDATE_TIME=0;                         //更新录音时间的消息编号
5       ImageButton ibRecord;                                          //录制按钮
6       ImageButton ibPause;                                           //暂停按钮
7       ImageButton ibStop;                                            //停止按钮
8       TextView tvTime;                                               //时间长度显示
9       Handler hd;                                                    //消息处理器
10      File myFile;                                                   //用于存放音轨的文件
11      MediaRecorder myMediaRecorder;                                 //媒体录音机
12      int countSecond=0;                                             //录制的秒数
13      boolean recordFlag=false;                                      //录制中标志
14      public void onCreate(Bundle savedInstanceState) {
15        super.onCreate(savedInstanceState);
16        setContentView(R.layout.main);                               //设置当前界面
17        ibRecord=(ImageButton)this.findViewById(R.id.ImageButton01);
                                                                       //初始化按钮引用
18        ibStop=(ImageButton)this.findViewById(R.id.ImageButton02);
19        tvTime=(TextView)findViewById(R.id.TextView02);              //初始化文本框
20        ibRecord.setOnClickListener(this);                           //为录制按钮添加监听器
21        ibStop.setOnClickListener(this);                             //为停止按钮添加监听器
22        hd=new Handler(){
23          public void handleMessage(Message msg){
24              super.handleMessage(msg);                              //调用父类处理
25              switch(msg.what){
26                case UPDATE_TIME:                                    //更新时间编号
27                    Bundle b=msg.getData();                          //获取消息中的数据
28                    String msgStr=b.getString("msg");                //获取内容字符串
29                    tvTime.setText(msgStr);                          //设置字符串到文本框中
30                break;
31              }}};}
32      public void onClick(View v) {
33      ……//该处省略部分代码，将在后面详细给出。
34          }
35      public void setTime(){                                         //设置显示时间的方法
36          int second=countSecond%60;                                 //计算分钟和秒
37          int minute=countSecond/60;
38          String msgStr=minute+"m:"+second+"s";                      //创建内容字符串
39          Bundle b=new Bundle();                                     //创建消息数据 Bundle
40          b.putString("msg", msgStr);                               //将内容放进数据 Bundle 中
41          Message msg=new Message();                                 //创建消息对象
42          msg.setData(b)    ;                                       //设置数据 Bundle 到消息中
43          msg.what=UPDATE_TIME;                                      //设置消息的 what 值
44          hd.sendMessage(msg);                                       //发送消息
45      }}
```

其中：

- 第 14～31 行为重写 onCreate()方法，在该方法中首先将 main.xml 设置为当前主界面，初始化按钮对象和 Handler 对象，在创建 Handler 对象时，检测接收消息编号为 UPDATE_TIME 的信息，并将获取的信息设置到文本框中。
- 第 35～45 行为自定义显示时间的方法，在该方法中计算分钟和秒后组装为新的字符串，利用所创建的 Bundle 对象绑定到 Message 中，设置消息的 what 值，使用 Handler 对象发送消息。

下面要介绍的是 Sample10_8_Activity 类中 onClick()方法的实现，该方法主要完成图片按钮的监听，将下列代码插入到上述代码的第 33 行。

代码位置：见光盘源代码/第 10 章/Sample10_8/src/com/bn/ex10h/Sample10_8_Activity.java。

```
1   if(v == ibRecord){                                          //按下录音按钮
2       if(!Environment.getExternalStorageState().equals(
3               android.os.Environment.MEDIA_MOUNTED)){         //若没有插闪存卡则报错
4               Toast.makeText(                                 //创建 Toast 对象
5                   this,
6                   "请检测内存卡",                              //提示信息
7                   Toast.LENGTH_SHORT).show();                 //显示 Toast
8           return;
9       }
10      try {
11          if(recordFlag==true){                   //若已经在录音中则提示并返回
12              Toast.makeText(                             //创建 Toast
13                  this,
14                  "录音中，请结束本次录音再开始新录音！",
15                  Toast.LENGTH_SHORT).show();             //显示 Toast
16              return;
17          }
18          myFile = File.createTempFile(                   //初始化临时文件对象
19              "myAudio",                                  //基本文件名
20              ".amr",                                     //后缀
21              Environment.getExternalStorageDirectory()   //目录路径
22          );
23          myMediaRecorder = new MediaRecorder();          //创建录音机对象
24          myMediaRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
                                                            //设置输入设备
25          myMediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.DEFAULT);
                                                            //设置输出格式
26          myMediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.DEFAULT);
                                                            //设置音频编码器
27          myMediaRecorder.setOutputFile(myFile.getAbsolutePath());
                                                            //设置输出文件的路径
28          myMediaRecorder.prepare();                      /准备录音
29          myMediaRecorder.start();                        //开始录音
30          recordFlag=true;                              //设置录音中标记为 true
31          new Thread(){                                   //启动一个线程进行计时
32              public void run(){
33                  while(recordFlag){
34                      countSecond++;                      //计时器加 1
35                      setTime();                          //调用方法设置新时长
36                      try {                               //休息 1000ms
37                          Thread.sleep(1000);
38                      } catch (InterruptedException e) {
39                          e.printStackTrace();            //打印异常
40                      }}}
```

```
41              }.start();
42          } catch (IOException e) {e.printStackTrace();              //打印异常
43          }}
44          else if(v == ibStop){                                      //按下停止按钮
45          if(myFile != null&&myMediaRecorder!=null){
46              myMediaRecorder.stop();                                //停止录音
47              myMediaRecorder.release();                             //释放录音机对象
48              myMediaRecorder = null;                    //将录音机对象引用设置为 null
49          }
50          recordFlag=false;                                  //设置录音中标记为 false
51          countSecond=0;                                     //计时器清 0
52          setTime();                                         //调用方法设置新时长
53      }
```

其中：

- 第 2～9 行表示按下录音按钮，首先判断手机中是否插闪存卡，若没有则使用 Toast 弹出提示信息。
- 第 11～30 行表示判断是否在录音中，若在录音中则使用 Toast 弹出提示信息，若没有则初始化临时文件对象，设置输入设备为麦克风，设置输出格式为 amr 格式，设置音频编码器为默认编码器。在设置好输出文件路径后准备录音并开始录音，此时将录音标志位设为 true。
- 第 31～41 行表示创建线程，每休息 1000ms 计时器加 1，并启动该线程。
- 第 44～53 表示按下停止按钮，首先判断 myFile 和 myMediaRecorder 是否为空，若不为空则表示处于录音过程中，停止录音释放录音机对象并将录音机对象设置为 null。最后将录音标志位设为 false，计数器清 0。

实例 9　采集图像数据

手机中另一个多媒体数据采集为图像的数据采集，在本节将对图像数据的采集开发进行介绍。

【实例描述】

该软件的主界面中包含 3 个按钮，分别为开始、拍照和保存。单击开始按钮时，系统打开手机的摄像头对景象进行扫描，在模拟器中只是模拟拍照的情形；单击拍照按钮时，进行拍照，将拍摄的照片显示在按钮右方，单击保存按钮时对拍摄的按钮进行保存处理。

本实例的运行效果如图 10-9 所示。

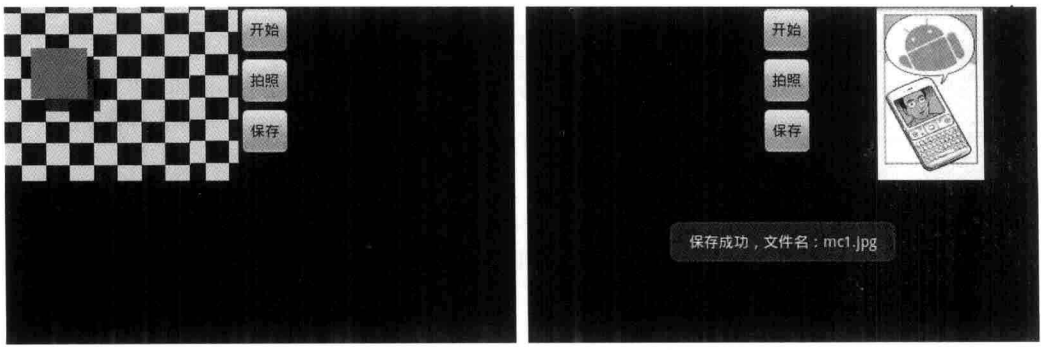


图 10-9　相片的拍摄

提示：在图 10-9 中从左至右依次表示的是单击开始按钮后的效果图，以及单击保存按钮后的效果图。

【实现过程】

在该软件中通过 Camera 对象进行相片的拍摄，在照片预览处使用的是 SurfaceView，通过 SurfaceViewHolder 控制 SurfaceView 要显示的内容，其中关键的技术在于初始化相机的方法，将相机参数赋值通过 Camera. startPreview()将 Preview 画面呈现在 SurfaceView 中。

为 3 个按钮设置初始化相机、拍照和保存事件的监听。使用流处理拍摄的照片，将其写入 SD 卡中，在进行该项操作时需要借助 Environment.getExternalStorageState()判断手机中是否存在 SD 卡，否则会出现异常。同时需要为该项目添加相应的权限，声明拍照的权限以及读/写存储卡的权限，修改权限见光盘源代码/第 10 章/Sample10_9/AndroidManifest.xml。

【代码解析】

经过权限的设置后，接下来要介绍的是 main.xml 文件和 Sample10_9_Activity 类，首先介绍主界面布局的实现文件，代码如下。

代码位置：见光盘源代码/第 10 章/Sample10_9/ res/layout/main.xml。

```
<?xml version="1.0" encoding="utf-8"?>          <!--版本号和编码方式-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:orientation="horizontal"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
    >                                           <!--LinearLayout 的属性设置-->
    <SurfaceView
              android:id="@+id/mySurfaceView"
              android:gravity="center_horizontal"
              android:layout_width="220px"
              android:layout_height="165px"
    />                                          <!--SurfaceView 的属性设置-->
    <LinearLayout
              android:id="@+id/LinearLayout01"
              android:layout_width="wrap_content"
         android:layout_height="wrap_content"
              android:orientation="vertical"
    >                                           <!--LinearLayout 的属性设置-->
      <Button
              android:text="开始"
              android:id="@+id/Button01"
              android:layout_width="wrap_content"
       android:layout_height="wrap_content">    <!--Button 的大小-->
      </Button>                                 <!--Button 的属性设置-->
      <Button
              android:text="拍照"
              android:id="@+id/Button03"
              android:layout_width="wrap_content"
        android:layout_height="wrap_content">   <!--Button 的大小-->
      </Button>                                 <!--Button 的属性设置-->
      <Button
              android:text="保存"
              android:id="@+id/Button02"
              android:layout_width="wrap_content"
```

```
35          android:layout_height="wrap_content">     <!--Button 的大小-->
36        </Button>                                   <!--Button 的属性设置-->
37    </LinearLayout>                               <!-- LinearLayout 的属性设置-->
38    <ImageView
39              android:id="@+id/myImageView"
40              android:layout_width="220px"
41              android:layout_height="165px"
42    />                                            <!--ImageView 的属性设置-->
43  </LinearLayout>                                 <!-- LinearLayout 的属性设置-->
```

其中：

- 第 7～12 行为 SurfaceView 的属性设置，设置 SurfaceView 的大小以及将内容设置为居中。
- 第 13～37 行为内部 LinearLayout 的属性设置和其内部 Button 的属性设置。
- 第 38～42 行为 ImageView 的属性设置，设置 ImageView 的大小以及 ID。

接下来介绍的是 Sample10_9_Activity 类的设计与实现，首先介绍的是该类框架的开发，代码如下。

代码位置：见光盘源代码/第 10 章/Sample10_9/src/com/bn/ex10i/Sample10_9_Activity.java。

```
1   package com.bn.chap10.camera;                                    //声明包
2   ……//该处省略了部分类的导入，读者可自行查阅随书光盘中源代码
3   public class Sample10_9_Activity extends Activity
4   implements SurfaceHolder.Callback, OnClickListener{ //继承 Activity 实现接口
5       SurfaceView mySurfaceView;                                   //SurfaceView 的引用
6       SurfaceHolder mySurfaceHolder;
7       Button bStart;                                               //打开按钮
8       Button bSave;                                                //关闭按钮
9       Button bTpg;                                                 //拍照按钮
10      Camera myCamera;                                             //Camera 的引用
11      boolean isView = false;                                      //是否正在浏览中
12      Bitmap bm;
13    public void onCreate(Bundle savedInstanceState){ //重写的方法
14        super.onCreate(savedInstanceState);
15        this.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_
LANDSCAPE);//设置为横屏
16        requestWindowFeature(Window.FEATURE_NO_TITLE);     //设置无标题
17        getWindow().setFlags(                                        //设置全屏
18              WindowManager.LayoutParams.FLAG_FULLSCREEN ,
19              WindowManager.LayoutParams.FLAG_FULLSCREEN
20        );
21        setContentView(R.layout.main);                               //设置显示主界面
22        mySurfaceView = (SurfaceView) findViewById(R.id.mySurfaceView);
                                                                       //得到拍照预览的引用
23        bStart = (Button) findViewById(R.id.Button01);     //初始化开始按钮引用
24        bSave = (Button) findViewById(R.id.Button02);      //初始化停止按钮引用
25        bTpg = (Button) findViewById(R.id.Button03);       //初始化拍照按钮引用
26        bStart.setOnClickListener(this);                             //为三个按钮添加监听器
27        bSave.setOnClickListener(this);                              //添加监听
28        bTpg.setOnClickListener(this);
29        mySurfaceHolder = mySurfaceView.getHolder();       //获得 SurfaceHolder
30        mySurfaceHolder.addCallback(this);                           //添加回调接口的实现
31        mySurfaceHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
32    }
33    protected void onPause() {                                       //重写的方法
34        super.onPause();                                             //调用父类
35        isView = false;
36        if(myCamera != null){                                  //若相机不为空则释放相机
```

```
37              myCamera.stopPreview();                          //停止预览
38              myCamera.release();                              //释放照相机对象
39              myCamera = null;                                 //清空照相机引用
40          }}
41      public void initCamera(){                                //初始化相机的方法
42       if(myCamera==null&&!isView)
43          myCamera = Camera.open();                            //打开照相机
44       }
45         if(myCamera != null && !isView){                      //判断是否为 true
46          try{
47            Camera.Parameters myParameters = myCamera.getParameters();
48            myParameters.setPictureFormat(PixelFormat.JPEG); //设置照片格式
49            //myParameters.setPreviewSize(200, 200);         //设置预览大小
50            myCamera.setParameters(myParameters);            //将参数设置入 Camera
51              myCamera.setPreviewDisplay(mySurfaceHolder); //设置照相机的预览者
52              myCamera.startPreview();                       //开始预览
53              isView = true;                               //将状态标志位设成 true
54          }
55          catch (IOException e){e.printStackTrace();       //打印异常
56      }}}
57      ShutterCallback myShutterCallback = new ShutterCallback(){ //快门回调接口
58          public void onShutter(){}                          //空实现
59      };
60      PictureCallback myRawCallback = new PictureCallback(){    //拍照回调接口
61      }
62          public void onPictureTaken(byte[] data, Camera camera) {} //空实现
63       };
64     PictureCallback myjpegCallback = new PictureCallback(){    //拍照回调接口
65       public void onPictureTaken(byte[] data, Camera camera) {
66          bm = BitmapFactory.decodeByteArray(data, 0, data.length); //创建对象
67          ImageView myImageView = (ImageView) findViewById(R.id.myImageView);
68          myImageView.setImageBitmap(bm);                  //图片显示到 ImageView 中
69          isView = false;
70          myCamera.stopPreview();                          //停止预览模式
71          myCamera.release();                              //释放
72          myCamera = null;                                 //将对象设为 null
73          isView = false;                                  //将标志位设为 false
74      }};
75      public void surfaceChanged(SurfaceHolder arg0, int arg1, int arg2, int arg3) {
76      }                                                    //重写 surfaceChanged
77      public void surfaceCreated(SurfaceHolder arg0) {
78      }                                                    //重写 surfaceCreated
79      public void surfaceDestroyed(SurfaceHolder arg0) {
80      }                                                    //重写 surfaceDestroyed
81      public void onClick(View v){
82      ……//该处省略部分代码，将在下面详细给出。
83      }}
```

其中：

- 第 13～32 行为重写 onCreate()方法，在该方法中首先设置为横屏和全屏拍摄，初始换 SurfaceView 和 Button 的引用，并为 3 个按钮添加监听器。获得 SurfaceHolde 并添加回调接口的实现，设定 mySurfaceHolder 类型为 SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS。
- 第 33～40 为重写 onPause()方法，主要完成的任务是判断相机是否为空，若不为空则释放相机。

- 第 41～56 行为自定义初始化相机的方法，在该方法中首先判断是否在非预览状态且未打开照相机，若满足条件则打开照相机，然后判断照相机是否成功打开且处在非预览状态，若是则进入预览状态。
- 第 57～74 行为实现快门和拍照会点接口，在 onPictureTaken()方法中创建对象，将标志位设为 false，并停止预览模式，释放相机对象并将其设置为 null。
- 第 75～83 行为实现 SurfaceHolder.Callback 接口中的 surfaceChanged()、surfaceCreated()以及 surfaceDestroyed()方法，onClick()方法将在后面详细介绍。

经过上述对 Sample10_9_Activity 的介绍，相信读者对此类有了一定的了解，接下来要介绍的是该类中 onClick()方法的实现，将下列代码插入到 Sample10_9_Activity 类框架的第 82 行。

代码位置：见光盘源代码/第 10 章/Sample10_9/src/com/bn/ex10i/Sample10_9_Activity.java。

```
1   if(v == bStart){                                          //按下的为开始按钮
2       initCamera();                                         //初始化相机
3   }
4   else if(v == bSave){                                      //按下保存按钮
5       if(bm==null){                                         //若没有拍摄照片则报错
6           Toast.makeText(                                   //创建 Toast
7                   this,
8                   "请拍摄成功后再保存!!! ",
9                   Toast.LENGTH_SHORT
10          ).show();                                         //显示 Toast
11          return;                                           //返回
12      }
13      if(!Environment.getExternalStorageState().equals(
14          android.os.Environment.MEDIA_MOUNTED)){           //若没有插闪存卡则报错
15          Toast.makeText(this, "请检测内存卡", Toast.LENGTH_SHORT).show();
16          return;                                           //返回
17      }
18      String path=Environment.getExternalStorageDirectory().getAbsolutePath();
19      int c=0;                                              //寻找可以使用的文件名
20      File fTest=new File(path+"/mc"+c+".jpg");             //创建 File 对象
21      while(fTest.exists()){
22          c++;                                              //计数器自加
23          fTest=new File(path+"/mc"+c+".jpg");              //创建新的对象
24      }
25      try{
26          FileOutputStream fout=new FileOutputStream(fTest);
                                                              //创建 FileOutputStream 对象
27          BufferedOutputStream bos = new BufferedOutputStream(fout);
28          bm.compress(
29                  Bitmap.CompressFormat.JPEG,               //图片格式
30                  80,                                       //品质 0～100
31                  bos                                       //使用的输出流
32          );
33          bos.flush();
34          bos.close();                                      //关闭 BufferedOutputStream
35          Toast.makeText(                                   //创建 Toast
36                  this,
37                  "保存成功，文件名：mc"+c+".jpg",
38                  Toast.LENGTH_SHORT
39          ).show();                                         //显示信息提示
40      }
41      catch(Exception e){                                   //捕获异常
42          Toast.makeText(                                   //创建 Toast
43                  this,
```

```
44                      "保存失败！",
45                      Toast.LENGTH_SHORT
46              ).show();                                          //显示信息提示
47          }}
48      else if(v == bTpg){                                        //按下拍照按钮
49          if(isView&&myCamera!=null){                            //判断是否符合条件
50              myCamera.takePicture(myShutterCallback, myRawCallback, myjpegCallback);
51          }
52          else{                                                  //不符合条件
53              Toast.makeText(                                    //创建 Toast
54                      this,
55                      "请按下开始按钮进入预览模式再拍照！",
56                      Toast.LENGTH_SHORT
57              ).show();                                          //显示 Toast
58      }}
```

其中：

- 第 1～3 行为单击开始按钮，调用初始化相机的方法。
- 第 4～47 行为单击保存按钮，首先判断拍摄的照片有没有报错，再判断手机中是否有 SD 卡。之后创建 File 对象并指定存储相片的路径和名称，然后创建流对象，对像片进行 SD 卡的储存，设置照片属性，保存完毕后关闭输出流，并使用 Toast 提示保存成功。
- 第 48～58 行为单击拍照按钮，首先判断标志位是否为 true 和 Camera 对象是否为 null，若满足则进行拍照，若不满足则使用 Toast 进行提示。

实例 10　采集视频数据

视频数据的采集也是手机多媒体数据采集中的一员，在本节将对其进行详细介绍。

【实例描述】

在本软件的主界面中包含两个图片按钮，一个为开始录制按钮，另一个为停止录制按钮，当单击开始按钮时，手机自动开始录制视屏，当到达最大录制时间或单击停止按钮时，手机停止录像。

本实例的运行效果如图 10-10 所示。

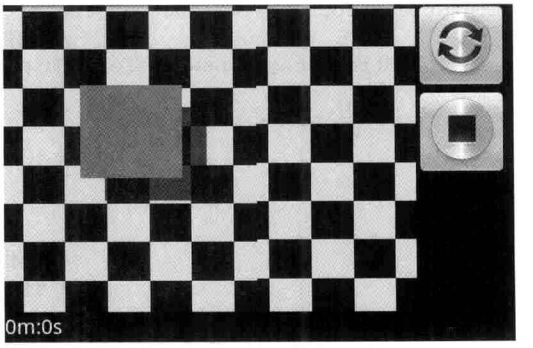


图 10-10　视频的录制

提示：本程序主要是讲解视频的采集。请读者朋友注意，在采集视频时需要使用真机，模拟器可能无法录制。

【实现过程】

由 MediaRecorder 创建的多媒体录制器对象设置录制视频的属性，其中 MediaRecorder.setPreviewDisplay()表示将 SurfaceView 设置为预览所用，setVideoSource()表示设置视频输入设备，MediaRecorder.setOutputFormat()设置录制视频的格式，并为界面中 ImageButton 添加监听器。

同样在进行实际开发前需要为其添加相应的权限声明，在本软件中设置的权限依次为，录音权限：<uses-permission android:name="android.permission.RECORD_AUDIO"/>，拍照权限：<uses-permission android:name="android.permission.CAMERA"/>和向存储卡中写入数据的权限：<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>。

【代码解析】

经过上面的理论介绍，接下来要介绍的是该软件的核心代码部分，首先要介绍的是 AndroidManifest.xml 文件，在该文件中主要完成的是权限的设置，代码如下。

代码位置：见光盘源代码/第 10 章/Sample10_10/AndroidManifest.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>                    <!--版本号和编码方式-->
2   <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3       package="com.bn.chap10.video"
4       android:versionCode="1"
5       android:versionName="1.0">                            <!--版本名称-->
6      <application android:icon="@drawable/icon" android:label="@string/app_name">
7         <activity android:name=".Sample10_10_Activity"
8                 android:label="@string/app_name">          <!--显示的程序名称-->
9            <intent-filter>
10              <action android:name="android.intent.action.MAIN" />
                                                              <!--标识程序开始-->
11              <category android:name="android.intent.category.LAUNCHER" />
12           </intent-filter>
13        </activity>
14     </application>
15     <uses-sdk android:minSdkVersion="7" />                 <!--设置 SDK 的值-->
16     <!-- 声明需要录音权限 -->
17     <uses-permission android:name="android.permission.RECORD_AUDIO"/>
18     <!-- 声明需要拍照权限 -->
19     <uses-permission android:name="android.permission.CAMERA"/>
20     <!-- 声明需要写存储卡权限 -->
21     <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
22  </manifest>
```

提示：上述代码中第 17 行为添加允许录音的权限，第 19 行为添加允许拍照的权限，第 21 行为添加允许向存储卡写入数据的权限。

权限设置完毕后，下一步要做的是界面布局的设计与实现，完成界面布局设置的文件是 main.xml，该文件代码的代码如下所列。

代码位置：见光盘源代码/第 10 章/Sample10_10/ res/layout/main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>                    <!--版本号和编码方式-->
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3                 android:orientation="horizontal"
4                 android:layout_width="fill_parent"
5                 android:layout_height="fill_parent"
6      >                                                      <!--LinearLayout 的属性设置-->
```

```
7       <LinearLayout
8               android:id="@+id/LinearLayout01"
9               android:layout_width="wrap_content"
10              android:layout_height="wrap_content"
11              android:orientation="vertical"
12      >                                       <!--LinearLayout 的属性设置-->
13        <SurfaceView
14              android:id="@+id/mySurfaceView"
15              android:gravity="center_horizontal"
16              android:layout_width="388px"
17              android:layout_height="291px"
18       />                                     <!--SurfaceView 的属性设置-->
19       <TextView
20              android:id="@+id/tvTime"
21              android:layout_width="wrap_content"
22              android:layout_height="wrap_content"
23              android:text="0m:0s"
24              android:textColor="@color/white"
25              android:textSize="20dip"
26       />                                     <!--TextView 的属性设置-->
27      </LinearLayout>                         <!--LinearLayout 的属性设置-->
28      <LinearLayout
29              android:id="@+id/LinearLayout02"
30              android:layout_width="wrap_content"
31              android:layout_height="wrap_content"
32              android:orientation="vertical"
33       >                                      <!--LinearLayout 的属性设置-->
34         <ImageButton
35                 android:id="@+id/ImageButton01"
36                 android:layout_width="wrap_content"
37                 android:layout_height="wrap_content"
38                 android:src="@drawable/record"
39           >                                  <!-- ImageButton 的属性设置-->
40           </ImageButton>
41           <ImageButton
42             android:id="@+id/ImageButton02"
43             android:layout_width="wrap_content"
44             android:layout_height="wrap_content"
45             android:src="@drawable/stop"
46           >                                  <!-- ImageButton 的属性设置-->
47           </ImageButton>
48      </LinearLayout>                         <!-- LinearLayout 的属性设置-->
49  </LinearLayout>
```

其中：

- 第 2～6 行为界面中最外层 LinearLayout 的属性设置，此 LinearLayout 设置为充满整个手机屏幕。
- 第 7～27 行为内层 LinearLayout 属性设置，在该 LinearLayout 中添加 SurfaceView 和 TextView，并设置大小和 ID 等属性。
- 第 28～48 行为内层 LinearLayout 属性设置，在该 LinearLayout 中添加两个 ImageButton，分别设置其显示的图片为@drawable/record 和@drawable/stop，大小为包裹内容。

设置完主界面的布局后，接下来要进行的是主类的开发，在详细介绍该类之前，首先介绍的是 Sample10_10_Activity 类的实现框架，代码如下。

代码位置：见光盘源代码/第 10 章/Sample10_10/src/com/bn/ex10j/Sample10_10_Activity.java。

```
1   package com.bn.chap10.video;                                    //声明包
2   ……//该处省略了部分类的导入，读者可自行查阅随书光盘中源代码
3   public class Sample10_10_Activity extends Activity implements OnClickListener{
```

```
4       public static final int UPDATE_TIME=0;                    //更新录音时间的消息编号
5       SurfaceView mSurfaceView;                                 //摄像预览用的 SurfaceView
6       private SurfaceHolder mSurfaceHolder;
7       private ImageButton  buttonStart;                         //开始录制按钮
8       private ImageButton  buttonStop;                          //停止录制按钮
9       private MediaRecorder recorder;                           //多媒体录制器
10      Handler hd;                                               //消息处理器
11      int countSecond=0;                                        //录制的秒数
12      private TextView tvTime;                                  //显示录制时间的文本 View
13      public void onCreate(Bundle savedInstanceState) {    //重写的 onCreate 方法
14        super.onCreate(savedInstanceState);                     //调用父类
15        this.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
16        requestWindowFeature(Window.FEATURE_NO_TITLE);      //设置无标题
17        getWindow().setFlags(
18              WindowManager.LayoutParams.FLAG_FULLSCREEN ,
19              WindowManager.LayoutParams.FLAG_FULLSCREEN
20        );                                                      //设置为全屏
21        setContentView(R.layout.main);                          //加载主界面
22        //对摄像预览用的 SurfaceView 及其 SurfaceHolder 进行初始化
23        mSurfaceView=(SurfaceView)this.findViewById(R.id.mySurfaceView);
24        mSurfaceHolder=mSurfaceView.getHolder();                //得到 Holder 的引用
25        mSurfaceHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
26        buttonStart=(ImageButton)this.findViewById(R.id.ImageButton01); //录制按钮
27        buttonStop=(ImageButton)this.findViewById(R.id.ImageButton02); //停止按钮
28        buttonStart.setOnClickListener(this);                   //给录制按钮添加监听器
29        buttonStop.setOnClickListener(this);                    //给停止按钮添加监听器
30        tvTime=(TextView)this.findViewById(R.id.tvTime); //显示录制时长的文本框引用
31        hd=new Handler(){                                       //线程中创建一个 Handler
32          public void handleMessage(Message msg){
33              super.handleMessage(msg);                         //调用父类处理
34              switch(msg.what){                                 //判断接收的 msg
35                case UPDATE_TIME:
36                    Bundle b=msg.getData();                     //获取消息中的数据
37                    String msgStr=b.getString("msg");           //获取内容字符串
38                    tvTime.setText(msgStr);                     //设置字符串到文本框中
39                break;
40              }}};}
41      public void onClick(View v) {                             //重写的 onClick 方法
42          ……//该处省略了部分代码，在后面将对其进行介绍。
43      }
44      protected void onPause() {                                //重写的 onPause 方法
45        super.onPause();                                        //调用父类
46        if(recorder!= null){                                    //若对象不为 null
47              recorder.stop();
48              recorder.release();                               //释放录相机
49              recorder = null;                                  //设置为 null
50          }}
51      public void setTime(){                                    //设置显示时间的方法
52        ……//该处省略了部分代码，在后面将对其进行介绍。
53      }}
```

其中：

- 第 4～12 行为该类中的主要成员变量的声明，其中第 9 行为多媒体录制器引用的声明。
- 第 13～40 行为重写 onCreste()方法，在该方法中首先将程序设置为横屏和全屏显示，将 main.xml 设置为当前主界面，随后对摄像预览用的 SurfaceView 及其 SurfaceHolder 进行初始化，为按钮添加监听器，创建 Handler 对象，根据消息 what 编号的不同，执行

不同的业务逻辑。

- 第 40～53 行为写 onClick()方法、onPause()方法以及自定义 setTime()方法，其中 onClick()方法和 setTime()方法在后面将对其进行介绍，onPause()方法则表示若多媒体录制器引用不为 null，则停止录像并将其释放，设置多媒体录制器引用为 null。

（1）下面要介绍 onClick()方法的实现，在该方法中主要完成的是根据按钮执行不同的业务逻辑，将下列代码插入到 Sample10_10_Activity 类的实现框架的第 42 行。

代码位置：见光盘源代码/第 10 章/Sample10_10/src/com/bn/ex10j/Sample10_10_Activity.java。

```
1   if(v== buttonStart){                                          //按下开始录制按钮
2       if(recorder!=null){                                       //若录制器不为空
3               Toast.makeText(                                   //创建 Toast
4                   this,
5                   "请停止本次拍摄再开始新的拍摄！",              //提示信息内容
6                   Toast.LENGTH_SHORT
7               ).show();                                         //显示 Toast
8               return;
9           }
10          try{
11              recorder = new MediaRecorder();                   //创建录制器对象
12              File outf=File.createTempFile(                    //创建输出文件对象
13                      "aas",                                    //名称
14                      ".mp4",                                   //格式
15                      Environment.getExternalStorageDirectory()
16      );
17      recorder.setPreviewDisplay(mSurfaceHolder.getSurface());//设置预览的View
18      recorder.setVideoSource(MediaRecorder.VideoSource.CAMERA); //设置视频输入设备
19          recorder.setAudioSource(MediaRecorder.AudioSource.MIC);
                                                                  //设置录音源为麦克风
20          recorder.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);
                                                                  //设置输出格式为 mp4
21          recorder.setVideoSize(640, 480);                      //设置视频大小
22          recorder.setVideoFrameRate(20);                       //设置视频帧速率
23      recorder.setVideoEncoder(MediaRecorder.VideoEncoder.H264);//设置视频编码
24      recorder.setAudioEncoder(MediaRecorder.AudioEncoder.DEFAULT); //设置音频编码
25      recorder.setMaxDuration(10000);                           //设置最大时长
26      recorder.setOutputFile(outf.getAbsolutePath());           //设置输出文件
27          recorder.prepare();                                   //准备录制器
28          recorder.start();                                     //开始录制
29          new Thread(){                                         //启动一个线程进行计时
30              public void run(){
31                  while(recorder!=null){
32                      countSecond++;                            //计时器加 1
33                      setTime();                                //调用方法设置新时长
34                          try {
35                              Thread.sleep(1000);               //休息 1000ms
36                          } catch (InterruptedException e) {
37                              e.printStackTrace();              //打印异常
38                          }}}
39              }.start();
40          }
41          catch(Exception e){
42              e.printStackTrace();                              //打印异常
43              Toast.makeText(                                   //创建 Toast
44                  this,
45                  "录制故障："+e.toString(),
46                  Toast.LENGTH_SHORT
```

```
47                ).show();                                    //显示 Toast
48            }}
49        else if(v==buttonStop){                              //按下的是停止按钮
50                recorder.stop();                             //停止录制
51            recorder.release();                              //释放录制器
52            recorder=null;                                   //清空录制器引用
53                countSecond=0;                               //计时器清 0
54                setTime();                                   //调用方法设置新时长
55        }
```

其中：

- 第 1～48 行为按下录制按钮时执行的代码，在代码中首先判断录制器引用是否为 null，若为空则创建录制器对象和输出文件对象，然后设置用于预览的 View、视频的输入设备、录音源与输出的格式，并制定视屏的分辨率、帧速率、视频编码以及音频编码。
- 第 49～55 行为按下停止按钮时执行的代码，在代码中执行停止录制的命令，并释放录制器，将其引用设置为 null，计时器清 0，调用自定义 setTime 的方法。

（2）最后介绍自定方法 setTime()的开发，该方法主要完成的是时间的设置，将下列代码插入到 Sample10_10_Activity 类的实现框架的第 52 行。

代码位置：见光盘源代码/第 10 章/Sample10_10/src/com/bn/ex10j/Sample10_10_Activity.java。

```
1    int second=countSecond%60;                               //计算分钟和秒数
2    int minute=countSecond/60;
3    String msgStr=minute+"m:"+second+"s";                    //创建内容字符串
4    Bundle b=new Bundle();                                   //创建消息数据 Bundle
5    b.putString("msg", msgStr);                              //将字符串放进 Bundle
6    Message msg=new Message();                               //创建消息对象
7    msg.setData(b);                                          //设置数据绑定到消息
8    msg.what=UPDATE_TIME;                                    //设置消息的 what 值
9    hd.sendMessage(msg);                                     //发送消息
```

其中：

- 第 1～3 行为计算分钟和秒数，并组装为新的字符串。
- 第 4～9 行为创建 Bundle 对象，将字符串信息放进 Bundle 中，创建消息对象，将 Bundle 放入消息中，指定消息编号为 UPDATE_TIME，通过 Handler 发送消息。

实例 11　视频播放器

手机不仅仅有录制视频的功能，还拥有播放视频的功能，在本节将介绍如何实现自定义的视频播放。

【实例描述】

本软件的主界面中含有 3 个图片按钮，依次表示开始播放、暂停播放和停止播放。在播放视频之前首先要确定手机 SD 卡中含有 aa.3gp 格式的视频，否则将无法播放。在视频播放的过程中，单击暂停按钮，视频停止播放，再次单击播放按钮视频继续播放。单击停止按钮，则停止视频的播放。

本实例的运行效果如 10-11 所示。

提示：图 10-11 中从左至右依次为软件开始运行和运行中的效果图，以及为播放视频时的效果图。由于模拟器具有局限性，所以建议读者在运行本程序时尽量使用真机。

图 10-11　视频播放效果图

【实现过程】

在软件中使用 MediaPlayer 对视频进行播放，通过 setDataSource()方法设定路径，以及使用 prepare()方法对要播放的视频进行准备，准备完成后使用 start()方法播放视频。显示播放视频所用的是 SurfaceView，通过 SurfaceHolde 对其进行设定，显示视频播放时的画面。

【代码解析】

经过上面的理论介绍，接下来要介绍的是该软件的核心代码部分，首先要介绍的是 main.xml 文件的实现，该文件的代码如下所列。

代码位置：见光盘源代码/第 10 章/Sample10_11/ res/layout/main.xml。

```
<?xml version="1.0" encoding="utf-8"?>      <!--版本号和编码方式-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:orientation="vertical"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
    >                                       <!-- LinearLayout 的属性设置-->
    <SurfaceView
          android:id="@+id/VideoView01"
          android:layout_width="320dip"
      android:layout_height="240dip">       <!-- SurfaceView 的大小-->
    </SurfaceView>                          <!--SurfaceView 的属性设置-->
    <TextView
          android:id="@+id/TextView01"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:textColor="@color/white"
          android:textSize="20dip"
          android:text="0m:0s"
    >                                       <!--TextView 的属性设置-->
    </TextView>
    <SeekBar
          android:id="@+id/SeekBar01"
          android:layout_width="fill_parent"
      android:layout_height="wrap_content">
```

```
25      </SeekBar>                                  <!--SeekBar 的属性设置-->
26      <MediaController
27            android:id="@+id/MediaController01"
28            android:layout_width="wrap_content"
29            android:layout_height="wrap_content"
30      />                                          <!--MediaController 的属性设置-->
31      <LinearLayout
32            android:id="@+id/LinearLayout01"
33                android:layout_width="wrap_content"
34            android:layout_height="wrap_content"
35            android:orientation="horizontal"
36      >                                           <!--LinearLayout 的属性设置-->
37       <ImageButton
38                 android:id="@+id/ImageButton01"
39                 android:layout_width="wrap_content"
40                 android:layout_height="wrap_content"
41                 android:src="@drawable/play"
42       >                                          <!--ImageButton 的属性设置-->
43       </ImageButton>                             <!--ImageButton 的属性设置-->
44       <ImageButton
45                 android:id="@+id/ImageButton02"
46                 android:layout_width="wrap_content"
47                 android:layout_height="wrap_content"
48                 android:src="@drawable/pause"
49       >
50       </ImageButton>                             <!--ImageButton 的属性设置-->
51       <ImageButton
52                 android:id="@+id/ImageButton03"
53                 android:layout_width="wrap_content"
54                 android:layout_height="wrap_content"
55                 android:src="@drawable/stop"
56       >                                          <!--ImageButton 的属性设置-->
57       </ImageButton>
58      </LinearLayout>                             <!-- LinearLayout 的属性设置-->
59  </LinearLayout>                                 <!-- LinearLayout 的属性设置-->
```

其中：

- 第 7～30 行为在 LinearLayout 中添加 SurfaceView、TextView、SeekBar 和 MediaController，分别设置 4 种控件的属性。
- 第 31～58 行为内层 LinearLayout 的属性设置，在此 LinearLayout 中添加 3 个 ImageButton 按钮，分别设置显示的图片。

上述代码介绍的是本程序的主界面的实现，接下来介绍 Sample10_11_Activity 类的设计与实现，首先将对该类的框架进行介绍，代码如下。

代码位置：见光盘源代码/第 10 章/Sample10_11/src/com/bn/ex10k/Sample10_11_Activity.java。

```
1   package com.bn.chap10.play;                          //声明包
2   ……//该处省略了部分类的导入，读者可自行查看随书光盘中源代码
3   public class Sample10_11_Activity extends Activity
4   implements OnClickListener,OnSeekBarChangeListener{
5       public static final int UPDATE_TIME=0;           //更新播放时间的消息编号
6       public static final String currentPlay="/sdcard/aa.3gp";     //播放路径
7       ImageButton play;                                            //播放按钮
8       ImageButton pause;                                           //暂停按钮
9       ImageButton stop;                                            //停止按钮
10      SurfaceView sv;                                  //播放显示用的 SurfaceView
11      SurfaceHolder sh;                                //播放用的 SurfaceHolder
12      MediaPlayer mp;                                  //媒体播放器
```

```
13      SeekBar sb;                                          //进度显示拖拉条
14      TextView tvTime;                                     //时间长度显示
15      Handler hd;                                          //消息处理器
16      int state=0;                                         //播放状态指示
17    public void onCreate(Bundle savedInstanceState){
18       super.onCreate(savedInstanceState);
19       setContentView(R.layout.main);                      //设置主界面
20       play=(ImageButton)this.findViewById(R.id.ImageButton01);   //播放按钮
21       pause=(ImageButton)this.findViewById(R.id.ImageButton02);  //暂停按钮
22       stop=(ImageButton)this.findViewById(R.id.ImageButton03);   //停止按钮
23       play.setOnClickListener(this);                      //为播放按钮添加监听器
24       pause.setOnClickListener(this);                     //为暂停按钮添加监听器
25       stop.setOnClickListener(this);                      //为停止按钮添加监听器
26       //初始化播放用的 SurfaceView 及 SurfaceHolder
27       sv=(SurfaceView)this.findViewById(R.id.VideoView01);
28       sh=sv.getHolder();                                  //得到 Holder 的对象
29       sh.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
30       sb=(SeekBar)this.findViewById(R.id.SeekBar01);      //初始化进度拖拉条引用
31       sb.setEnabled(false);                               //不在暂停状态禁用拖拉条
32       sb.setOnSeekBarChangeListener(this);                //给进度拖拉条添加监听器
33       tvTime=(TextView)findViewById(R.id.TextView01);     //显示播放时长的文本框
34       hd=new Handler() {                                  //线程中创建一个 Handler
35          public void handleMessage(Message msg){
36              super.handleMessage(msg);                    //调用父类处理
37              switch(msg.what){
38                 case UPDATE_TIME:                         //msg 为 UPDATE_TIME
39                     Bundle b=msg.getData();               //获取消息中的数据
40                     String msgStr=b.getString("msg");     //获取内容字符串
41                     tvTime.setText(msgStr);               //显示录音时长的文本框
42                 break;
43              }}};}
44     public void onClick(View v) {                         //重写的 onClick 方法
45         ……//该处省略了部分代码，在后面将会详细介绍。
46     }
47    public void setTime(int countSecond){                  //设置显示时间的方法
48         ……//该处省略了部分代码，在后面将会详细介绍。
49    }
50     public void onProgressChanged(SeekBar seekBar, int progress,
51              boolean fromUser) {                          //实现进度拖拉的监听方法
52         if(mp!=null&&state==2){                           //进度拖拉到指定位置
53              mp.seekTo(progress*mp.getDuration()/sb.getMax());
54         }}
55     public void onStartTrackingTouch(SeekBar seekBar) {}     //重写的方法
56     public void onStopTrackingTouch(SeekBar seekBar) { }     //重写的方法
57  }
```

其中：

- 第 5～16 行为声明该类中的成员变量，其中的 state 表示播放状态的标志位，0 表示未准备，1 表示播放中，2 表示暂停。
- 第 17～43 行为重写 onCreste()方法，在该方法中设置 main.xml 为当前主界面，初始化图片按钮、SurfaceView 和进度拖拉条，为图片按钮和拖拉条设置监听器。最后创建 Handler 对象，在 handleMessage()方法中根据消息 what 编号的不同，执行不同的业务逻辑。
- 第 44～49 行为重写 onClick()方法、自定义 setTime()方法，这两个方法将在后面详细给出。
- 第 50～57 行为进度拖拉条的监听器，允许拖拉条在播放状态中进行拖拉。

（1）接下来要介绍的是 onClick()方法，该方法中主要实现的是图片按钮的监听功能，根据不同的按钮处理不同的业务逻辑，将下列代码插入到上述 Sample10_11_Activity 类框架的第 45 行。

代码位置：见光盘源代码/第 10 章/Sample10_11/src/com/bn/ex10k/Sample10_11_Activity.java。

```
if(v==play){                                                    //按下播放按钮
    if(state==1){                                               //若当前已经是播放状态
        Toast.makeText(                                         //创建 Toast
            this,
            "播放中，请结束本次播放再开始新的播放！",
            Toast.LENGTH_SHORT
        ).show();                                               //显示 Toast
        return;
    }
    else if(state==0){                                          //若当前是未准备状态
        mp=new MediaPlayer();                                   //创建媒体播放器对象
        mp.setAudioStreamType(AudioManager.STREAM_MUSIC); //设置音频流格式
        mp.setDisplay(sh);                                      //设置显示用 SurfaceView
        mp.setOnCompletionListener(                             //视频播放结束添加的监听器
          new OnCompletionListener(){
              public void onCompletion(MediaPlayer arg0) {
                                                                //歌曲播放结束停止播放
                  state=2;                                      //更新界面状态
    }});
    try{
        mp.setDataSource(currentPlay);                          //设置播放路径
        mp.prepare();                                           //准备播放
        }
      catch(Exception e){
        e.printStackTrace();                                    //打印异常
      }
      sb.setMax(mp.getDuration()/1000);                         //根据片长设置拖拉条最大值
      }
      mp.start();                                               //开始播放
      state=1;                                                  //状态设置为 1 表示播放中
      sb.setEnabled(false);                                     //禁用拖拉条
      new Thread(){                                             //定时更新进度条及时间
        public void run(){
          while(state==1){                                      //播放状态
            sb.setProgress(mp.getCurrentPosition()/1000); //设置拖拉条
            setTime(mp.getCurrentPosition()/1000);    //设置时间
            try{
              Thread.sleep(1000);                               //线程睡眠 1 秒钟
            }
            catch(Exception e){
              e.printStackTrace();                              //打印异常
            }}}
      }.start();
    }
    else if(v==pause){                                          //按下暂停按钮
      if(state!=1){                                             //若当前不是播放状态
        Toast.makeText(                                         //创建 Toast
          this,
          "请在播放状态再暂停！",
          Toast.LENGTH_SHORT
        ).show();                                               //显示 Toast 提示
        return;
      }
```

```
53            mp.pause();                                    //暂停
54            state=2;                                       //设置状态为 2 表示暂停
55            sb.setEnabled(true);                           //启用拖拉条
56        }
57    else if(v==stop){                                      //按下停止按钮
58        if(state==0){                                      //若处于停止状态
59            return;                                        //返回
60        }
61        state=0;                                           //设为停止状态
62        mp.stop();                                         //停止播放
63        mp.release();                                      //释放播放器
64        mp=null;                                           //清空引用
65        sb.setEnabled(false);                              //禁用拖拉条
66        sb.setProgress(0);                                 //设置进度为 0
67        setTime(0);                                        //设置时间为 0
68    }
```

其中：

- 第 1～43 行为播放按钮添加监听器，首先判断当前状态是否是播放状态，之后判断是否为停止状态，并设置音频流格式。为视频播放结束添加监听器，歌曲播放结束，更新界面；设置播放路径，播放歌曲，并将标志位设置为播放状态，禁止使用拖拉条，更新时间和拖拉条。
- 第 44～56 行为按下暂停按钮，首先判断当前状态是否为播放状态，若是，则暂停播放音乐，将状态值设为 2，启用拖拉条；若不是则弹出 Toast 进行信息提示。
- 第 57～68 行为按下停止按钮，首先判断当前状态是否为停止状态，若是则直接返回，若不是则将状态值设为 0，释放播放器，清空引用，将时间和拖拉条进度设为 0。

（2）最后要介绍 setTime()方法的实现，在该方法中主要实现的是时间的设置，将下列代码插入到上述 Sample10_11_Activity 类框架的第 45 行。

代码位置：见光盘源代码/第 10 章/Sample10_11/src/com/bn/ex10k/Sample10_11_Activity.java。

```
1    int second=countSecond%60;                             //计算分钟和秒数
2    int minute=countSecond/60;
3    String msgStr=minute+"m:"+second+"s";                  //创建内容字符串
4    Bundle b=new Bundle();                                 //创建消息数据 Bundle
5    b.putString("msg", msgStr);                            //将内容放进数据 Bundle 中
6    Message msg=new Message();                             //创建消息对象
7    msg.setData(b);                                        //设置数据 Bundle 到消息中
8    msg.what=UPDATE_TIME;                                  //设置消息的 what 值
9    hd.sendMessage(msg);                                   //发送消息
```

其中：

- 第 1～3 行为计算分钟和秒数，并组装为新的字符串。
- 第 4～9 行为创建 Bundle 对象，将字符串信息放进 Bundle 中，创建消息对象，将 Bundle 放入消息中，指定消息编号为 UPDATE_TIME，通过 Handler 发送消息。

实例 12　自定义动画效果

在手机应用程序的开发中，有时候需要使用一些动画。复杂的动画效果需要使用对其进行详细规划，但简单的动画可以通过简单的途径实现，在本节将对其进行详细介绍。

【实例描述】

本软件运行时，图片会从从小变大，从半透明到不透明，顺时针方向旋转，当运行结束时，图片以原大小尺寸显示在主界面上。本软件描述了 Android 平台为开发人员提供的开发简单动画的简单途径——渐变动画。

本实例的运行效果如图 10-12 所示。

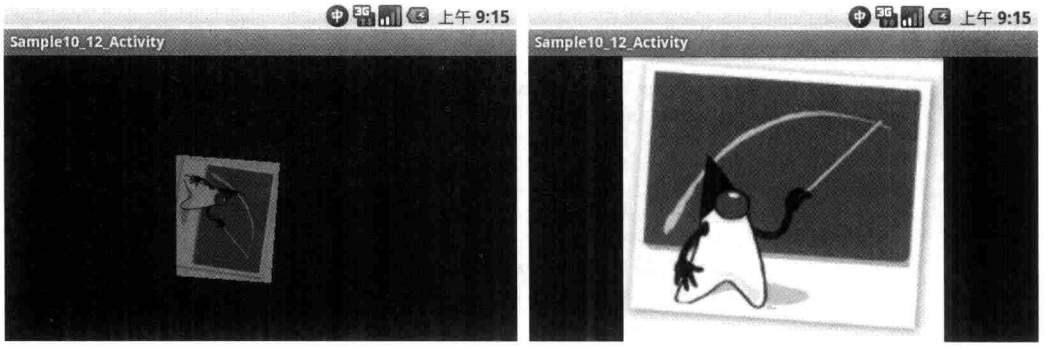


图 10-12　自定义动画运行效果

提示：在图 10-12 中依次为开始运行时的效果图，以及运行结束时的效果图。

【实现过程】

Android 平台下的渐变动画有 4 种要素组成，依次为 Alpha（透明度）、scale（尺寸伸缩）、translate（位置变化）和 rotate（图形旋转）。渐变动画中 4 种要素的变化情况是使用 xml 配置文件来描述的。

在新建项目后首先在 res 目录下新建 anim 子目录，然后在 anim 子目录下创建动画的 sml 描述文件，如在本软件中使用的文件名称为“myanmi.xml”。

【代码解析】

通过上面的理论介绍，相信读者对于渐变动画已经有所了解，在本部分将对该软件的核心代码进行详细介绍，首先要介绍的是 mian.xml 文件，代码如下。

代码位置：见光盘源代码/第 10 章/Sample10_11/ res/layout/main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>      <!--版本号和编码方式-->
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3                 android:orientation="vertical"
4                 android:layout_width="fill_parent"
5                 android:layout_height="fill_parent"
6   >                                           <!-- LinearLayout 的属性设置-->
7       <ImageView
8                     android:id="@+id/myImageView"
9                     android:layout_width="fill_parent"
10                    android:layout_height="fill_parent"
11                    android:src="@drawable/duke"
12      />                                      <!--ImageView 的属性设置-->
13  </LinearLayout>
```

上述代码介绍本程序主界面的布局，接下来要介绍的是 myanmi.xml 文件，该文件实现了

图片的渐变动画效果，该文件代码如下所列。

代码位置：见光盘源代码/第 10 章/Sample10_11/ res/anmi/myanmi.xml。

```
<?xml version="1.0" encoding="utf-8"?>          <!-- XML 的版本以及编码方式 -->
<set xmlns:android="http://schemas.android.com/apk/res/android">
  <alpha
            android:fromAlpha="0.1"
            android:toAlpha="1.0"
            android:duration="2000"
  />                                            <!-- 透明度的变换 -->
  <scale
     android:interpolator= "@android:anim/accelerate_decelerate_interpolator"
            android:fromXScale="0.0"
            android:toXScale="1.4"
            android:fromYScale="0.0"
            android:toYScale="1.4"
            android:pivotX="50%"
            android:pivotY="50%"
            android:fillAfter="false"
            android:duration="3000"
  />                                            <!-- 尺寸的变换 -->
  <translate
            android:fromXDelta="30"
            android:toXDelta="0"
            android:fromYDelta="30"
            android:toYDelta="50"
            android:duration="3000"
  />                                            <!-- 位置的变换 -->
  <rotate
     android:interpolator="@android:anim/accelerate_decelerate_interpolator"
            android:fromDegrees="0"
            android:toDegrees="+350"
            android:pivotX="50%"
            android:pivotY="50%"
            android:duration="3000"
  />                                            <!-- 旋转变换 -->
</set>
```

其中：

- 第 3～7 行为透明度的变换，其中 fromAlpha 属性为动画起始时的透明度，toAlpha 属性为动画结束时的透明度，duration 为动画持续的时间。
- 第 8～18 行为尺寸的变换，在属性设置中，fromXScale 为动画起始时 x 坐标，toXScale 为动画结束时 x 坐标，fromYScale 为动画开始时 y 坐标，toYScale 为动画结束时 y 坐标，pivotX 和 pivotY 设置动画相对于自身的位置，fillAfter 表示动画的转换在动画结束后是否被应用。
- 第 19～25 行为位置的变换，在属性设置中，fromXDelta 属性为动画起始时 x 坐标上的位置，toXDelta 属性为动画结束时 x 坐标上的位置，fromYDelta 属性为动画起始时 y 坐标上的位置，toYDelta 属性为动画结束时 y 坐标上的位置。
- 第 26～33 行为旋转变换，在属性设置中，interpolator 同样为一个动画的插入器，fromDegrees 属性为动画起始时物件的角度，toDegrees 属性为动画结束时物件的旋转角度。

上述代码介绍的是图片的渐变动画效果的设置，接下来要介绍的是 Sample10_12_Activity 类的设计与实现，该类代码如下所列。

代码位置：见光盘源代码/第 10 章/Sample10_12/src/com/bn/ex10l/Sample10_12_Activity.java。

```
package com.bn.chap10.zdydh;                    //声明包
```

```
2        ......//该处省略了部分类的导入，读者可自行查看随书光盘中源代码
6    import android.widget.ImageView;
7    public class Sample10_12_Activity extends Activity {
8        Animation myAnimation;                                    //动画的引用
9        ImageView myImageView;                                    //ImageView 的引用
10     public void onCreate(Bundle savedInstanceState) {
11       super.onCreate(savedInstanceState);                       //调用父类
12       setContentView(R.layout.main);                            //设置当前显示的 View
13       myAnimation= AnimationUtils.loadAnimation(this,R.anim.myanim);
                                                                   //加载动画
14       myImageView = (ImageView) this.findViewById(R.id.myImageView);
                                                                   //得到 ImageView 的引用
15       myImageView.startAnimation(myAnimation);                  //启动动画
16     }}
```

其中：

- 第 8～15 行为声明动画和 ImageView 的引用，设置当前显示的 View，加载动画，获取 ImageView 的引用后启动动画。

实例 13　小球游戏

手机中的游戏为玩家带来了不一样的感受，在 Android 平台上可以开发出各种各样的游戏，游戏运行是否流畅决定了该游戏的可玩性和对玩家的吸引力，在本节将对手机游戏的开发进行介绍。

【实例描述】

运行该游戏后，进入准备界面，在准备界面单击屏幕，游戏自动开始，小球开始沿一个方向运动，碰到壁时自动改变运动方向，若小球向下运动，板未能接住小球则游戏失败，若时间到达 40 秒，则赢取本局游戏。

本实例的运行效果如图 10-13 所示。

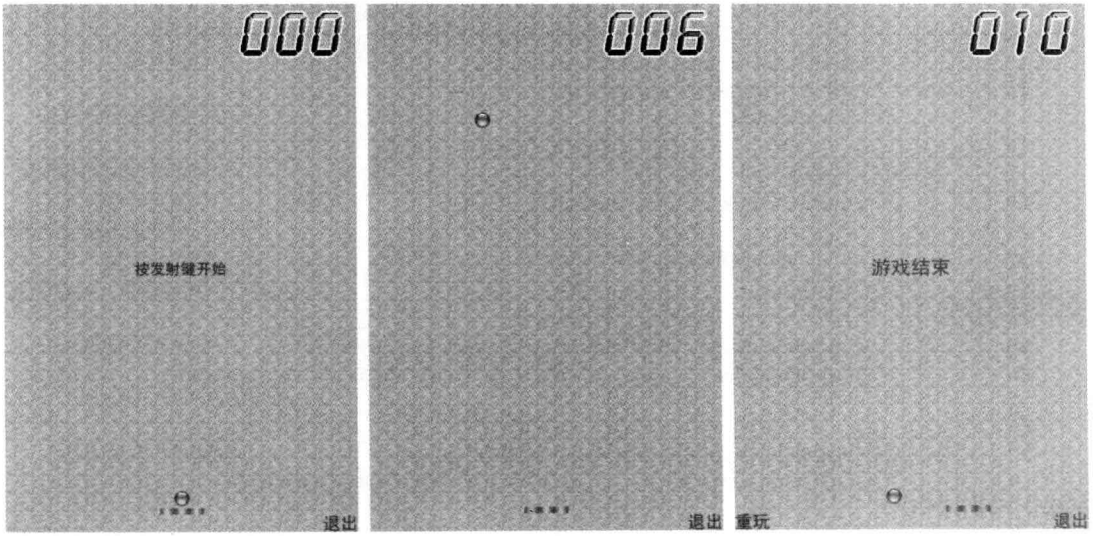


图 10-13　小球游戏截图

提示：在图 10-13 中从左至右依次为游戏未开始时的截图，游戏过程中的截图，以及游戏失败结束时的截图。

【实现过程】

该软件中使用平面贴图技术实现游戏界面的绘制，根据获取的手机分辨率，进行碰撞检测的开发。在游戏开发过程中，使用 onTouchEvevnt()方法实现游戏的开始，移动挡板和单击重玩或退出按钮。计时以及小球的运动和界面的重回是通过线程完成的。

在此游戏中使用了游戏声音，Android 平台下的游戏声音使用的是 SoundPool，使用 SoundPool 需要在游戏加载时将需要的声音资源加载，在需要的时候调用 playSound()方法播放即可，这样可以避免声音的延迟。在本游戏中使用 SoundPool 循环播放背景音乐。

【代码解析】

前面是该游戏的一些理论介绍，下面将对该游戏的个各类进行详细介绍，首先要介绍的是 Sample10_13_Activity 类，该类的代码如下所列。

代码位置：见光盘源代码/第 10 章/Sample10_13/src/com/bn/ex10m/Sample10_13_Activity.java。

```
1   package com.bn.chap10.ballgame;                                    //声明包
2   import java.util.HashMap;                                          //导入相关包
3   ......//该处省略了部分类的导入，读者可自行查看随书光盘中源代码
4   import android.view.WindowManager;                                 //导入相关包
5   public class Sample10_13_Activity extends Activity {
6       GameView gameView;
7       SoundPool soundPool;                                           //声音缓冲池
8       HashMap<Integer, Integer> soundPoolMap;                        //存放声音 ID 的 Map
9       public void onCreate(Bundle savedInstanceState) {
10          super.onCreate(savedInstanceState);
11          initSounds();                                              //初始化声音
12          requestWindowFeature(Window.FEATURE_NO_TITLE);             //设置全屏
13          getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN ,
14                  WindowManager.LayoutParams.FLAG_FULLSCREEN);
15          gameView = new GameView(this);                             //创建对象
16          setContentView(gameView);
17          playSound(1, -1);                                          //循环播放背景音乐
18      }
19      public void initSounds(){                                      //声音缓冲池的初始化
20          soundPool = new SoundPool(                                 //创建声音缓冲池
21              4,                                                     //同时能最多播放的个数
22              AudioManager.STREAM_MUSIC,                             //音频的类型
23              100                                                    //声音的播放质量，目前无效
24          );
25          soundPoolMap = new HashMap<Integer, Integer>();            //创建声音资源 Map
26          soundPoolMap.put(1, soundPool.load(this, R.raw.gamestart, 1));
                                                                       //将加载声音资源 id
27      }
28      public void playSound(int sound, int loop) {                   //播放声音的方法
29          AudioManager mgr = (AudioManager)this.getSystemService(Context.
AUDIO_SERVICE);
30          float streamVolumeCurrent = mgr.getStreamVolume(AudioManager.STREAM_
MUSIC);
31          float streamVolumeMax = mgr.getStreamMaxVolume(AudioManager.STREAM_
MUSIC);
```

```
32          float volume = streamVolumeCurrent / streamVolumeMax; //计算声音大小
33          soundPool.play(
34          soundPoolMap.get(sound),                    //声音资源 id
35          volume,                                     //左声道音量
36          volume,                                     //右声道音量
37          1,                                          //优先级
38          loop,                                       //循环次数 -1 代表永远循环
39          0.5f                                        //回放速度 0.5f～2.0f 之间
40        );}}
```

其中：

- 第 6～18 行为声明引用，在 onCreate()方法中初始化声音，设置为全屏，创建 GameView 对象，并将当前界面设置为 gameView，循环播放背景音乐。
- 第 19～27 行为初始化声音的方法，在该方法中创建声音缓冲池，设置同时能够播放 4 种声音，创建声音资源 Map 并加载声音资源。

上述代码介绍的是 Activity 类的开发，接下来介绍的是游戏界面 GameView 框架的开发，代码如下。

代码位置：见光盘源代码/第 10 章/Sample10_13/src/com/bn/ex10m/GameView.java。

```
1   package com.bn.chap10.ballgame;                          //声明包
2   ……//该处省略了部分类的导入代码，读者可自行查看随书光盘中源代码。
3   public class GameView extends SurfaceView implements SurfaceHolder.Callback{
4       Sample10_13_Activity activity;                       //Activity 引用
5       TimeThread tt;                                       //TimeThread 引用
6       BallGoThread bgt;                                    //BallGoThread 引用
7       int backSize=16;                                     //背景块尺寸
8       int screenWidth = 320;                               //屏幕宽度
9       int screenHeight = 480;                              //屏幕高度
10      int bannerWidth=40;                                  //板宽度
11      int bannerHeight=6;                                  //板高度
12      int bottomSpance=16;                                 //底下空白
13      int bannerSpan=5;                                    //板步进
14      int ballSpan=8;                                      //球步进
15      int ballSize=16;                                     //小球尺寸
16      int hintWidth=100;                                   //提示宽度
17      int hintHeight=20;                                   //提示高度
18      int status=0;                                        //状态值
19      int score=0;                                         //得分
20      int ballx;                                           //小球 x 坐标
21      int bally;                                           //小球 y 坐标
22      int direction=0;                                     //小球方向
23      int bannerX;                                         //板 X 坐标
24      int bannerY;                                         //板 Y 坐标
25      int scoreWidth = 32;
26      Bitmap iback;                                        //背景图
27      Bitmap[] iscore=new Bitmap[10];                      //得分图
28      Bitmap iball;                                        //球
29      Bitmap ibanner;                                      //板
30      Bitmap ibegin;                                       //开始
31      Bitmap igameover;                                    //游戏失败结束
32      Bitmap iwin;                                         //游戏获胜结束
33      Bitmap iexit;                                        //退出
34      Bitmap ireplay;                                      //重玩
35      GameViewDrawThread gameViewDrawThread;               //绘制线程
```

```
36      public GameView(Sample10_13_Activity activity) {
37          super(activity);
38          getHolder().addCallback(this);                          //注册回调接口
39          this.activity = activity;                               //获取引用
40          initBitmap();                                           //初始化位图
41          gameViewDrawThread = new GameViewDrawThread(this);      //创建对象
42      }
43      public void initBitmap(){                                   //将图片加载
44          iback=BitmapFactory.decodeResource(getResources(),R.drawable.back);
                                                                    //背景图
45          iscore[0]=BitmapFactory.decodeResource(getResources(),R.drawable.d0);
                                                                    //数字 0
46          iscore[1]=BitmapFactory.decodeResource(getResources(),R.drawable.d1);
                                                                    //数字 1
47          iscore[2]=BitmapFactory.decodeResource(getResources(),R.drawable.d2);
                                                                    //数字 2
48          iscore[3]=BitmapFactory.decodeResource(getResources(),R.drawable.d3);
                                                                    //数字 3
49          iscore[4]=BitmapFactory.decodeResource(getResources(),R.drawable.d4);
                                                                    //数字 4
50          iscore[5]=BitmapFactory.decodeResource(getResources(),R.drawable.d5);
                                                                    //数字 5
51          iscore[6]=BitmapFactory.decodeResource(getResources(),R.drawable.d6);
                                                                    //数字 6
52          iscore[7]=BitmapFactory.decodeResource(getResources(),R.drawable.d7);
                                                                    //数字 7
53          iscore[8]=BitmapFactory.decodeResource(getResources(),R.drawable.d8);
                                                                    //数字 8
54          iscore[9]=BitmapFactory.decodeResource(getResources(),R.drawable.d9);
                                                                    //数字 9
55          iball = BitmapFactory.decodeResource(getResources(), R.drawable.ball);
                                                                    //球
56          ibanner = BitmapFactory.decodeResource(getResources(), R.drawable.
banner);//板
57          ibegin = BitmapFactory.decodeResource(getResources(), R.drawable.
begin); //开始
58          igameover = BitmapFactory.decodeResource(getResources(), R.drawable.
gameover);
59          iwin = BitmapFactory.decodeResource(getResources(), R.drawable.win);
                                                                    //获胜
60          iexit       =       BitmapFactory.decodeResource(getResources(),
R.drawable.exit);     //退出
61          ireplay = BitmapFactory.decodeResource(getResources(), R.drawable.
replay); //重玩
62          initBallAndBanner();                                    //初始化小球位置及板 X 坐标
63      }
64      public void initBallAndBanner(){
65          bally=screenHeight-bottomSpance-bannerHeight-ballSize; //初始化小球位置
66          ballx=screenWidth/2-ballSize/2;
67          bannerX=screenWidth/2-bannerWidth/2;        //初始化板 X 坐标
68          bannerY=screenHeight-bottomSpance-bannerHeight; //初始化板 Y 坐标
69      }
70      public void replay(){
71          if(status==2||status==3){                   //初始化小球位置及板 X 坐标
72              initBallAndBanner();
73              score=0;                                //初始化时间
74              status=0;                               //初始化状态
75              direction=3;                            //初始化方向
```

```
76          }}
77      protected void onDraw(Canvas canvas) {
78          ……//该处省略了部分代码，在后面会详细介绍。
79          }
80      public boolean onTouchEvent(MotionEvent event) {
81          ……//该处省略了部分代码，在后面会详细介绍。
82      }
83      public void surfaceChanged(SurfaceHolder holder, int format, int width,
int height) {}
84      public void surfaceCreated(SurfaceHolder holder) {  //创建时启动相应进程
85        this.gameViewDrawThread.flag = true;              //标志位设为 true
86        gameViewDrawThread.start();                       //启动线程
87      }
88      public void surfaceDestroyed(SurfaceHolder holder) {//摧毁时释放相应进程
89        boolean retry = true;                             //标志位设为 true
90        this.gameViewDrawThread.flag =  false;            //线程标志位设为 false
91        while (retry) {
92           try {
93                   gameViewDrawThread.join();             //调用 join 方法
94               retry = false;                             //标志位设为 false
95           }
96           catch (InterruptedException e) {}      //不断地循环，直到刷帧线程结束
97        }}}
```

其中：

- 第 4～35 行为该类中成员变量的声明，其中第 18 行为游戏状态值，0 表示等待开始，1 表示进行中，2 表示游戏结束，3 表示游戏胜利。
- 第 36～42 行为该类的构造器，在构造器内完成的业务逻辑为获取 Activity 引用，注册回调接口，初始化位图以及创建线程对象。
- 第 43～63 行为初始化图片的方法，其中第 62 行为调用初始化小球位置的方法。
- 第 64～69 行为初始化小球位置的方法，每次调用时将小球和板设置在初始位置。
- 第 70～76 行为自定义 replay()方法，若状态值处于 2 或 3，则初始化小球位置，将控制时间、状态和方向的变量初始化。
- 第 77～97 行为重写父类的方法，其中 onDraw()和 onTouchEvent()方法会在后面进行介绍，这里就不再一一赘述。在重写的 surfaceCreated()方法中将线程标志位设置为 true，并启动线程；在 surfaceDestroyed()方法中将线程标志位设为 false，使用 while 循环判断线程是否关闭，成功关闭时将控制 while 循环的标志位设为 false。

（1）下面要介绍的是 onDraw()方法的设计与实现，将下列代码插入到 GameView 框架的第 78 行。

代码位置：见光盘源代码/第 10 章/Sample10_13/src/com/bn/ex10m/GameView.java。

```
1   super.onDraw(canvas);                                                  //清背景
2   int cols=screenWidth/backSize+((screenWidth%backSize==0)?0:1);   //列数
3   int rows=screenHeight/backSize+((screenHeight%backSize==0)?0:1); //行数
4   for(int i=0;i<rows;i++){
5       for(int j=0;j<cols;j++){
6           canvas.drawBitmap(iback, 16*j,16*i, null);          //绘制背景
7       }
8   }
9   String scoreStr=score+"";                                              //时间字符串
10  int loop=3-scoreStr.length();                                          //获取时间长度
11  for(int i=0;i<loop;i++){                                               //循环
12      scoreStr="0"+scoreStr;
```

```
13  }
14  int startX=screenWidth-scoreWidth*3-10;                          //起始位置
15  for(int i=0;i<3;i++){                                            //循环
16      int tempScore=scoreStr.charAt(i)-'0';
17      canvas.drawBitmap(iscore[tempScore], startX+i*scoreWidth,5, null); //绘制时间
18  }
19  canvas.drawBitmap(iball,ballx,bally, null);                      //绘制小球
20  canvas.drawBitmap(ibanner,bannerX,bannerY, null);                //绘制板
21  if(status==0){                                                   //绘制开始提示
22     canvas.drawBitmap(
23          ibegin,
24          screenWidth/2-hintWidth/2,
25          screenHeight/2-hintHeight/2, null
26      );}
27  if(status==2){                                                   //绘制失败提示
28      canvas.drawBitmap(                                           //绘制方法
29          igameover,
30          screenWidth/2-hintWidth/2,                               //X 坐标
31          screenHeight/2-hintHeight/2, null
32      ); }
33  if(status==3){                                                   //绘制胜利
34      canvas.drawBitmap(                                           //绘制方法
35          iwin,
36              screenWidth/2-hintWidth/2,                           //X 坐标
37              screenHeight/2-hintHeight/2, null
38          );}
39      canvas.drawBitmap(                                           //绘制退出选项
40          iexit,
41              screenWidth-32,                                      //X 坐标
42             screenHeight-16,null
43         );
44  if(status==2||status==3){                                        //绘制重玩选项
45      canvas.drawBitmap(                                           //绘制方法
46              ireplay ,
47              0,                                                   //X 坐标
48                  screenHeight-16,null
49   );}
```

其中：

- 第 1～18 行主要完成背景和时间的绘制，背景的绘制是依据屏幕大小绘制的。
- 第 19～26 行表示对小球和板的绘制。
- 第 27～38 行主要完成开始、失败、胜利提示消息的绘制。
- 第 39～49 行主要完成退出选项和重玩选项的绘制。

（2）下面介绍的是 onTouchEvent()方法的设计与实现，将下列代码插入到 GameView 框架的第 81 行。

代码位置：见光盘源代码/第 10 章/Sample10_13/src/com/bn/ex10m/GameView.java。

```
1   int x = (int) event.getX();                                     //获取触摸点 X 坐标
2   int y = (int) event.getY();                                     //获取触摸点 Y 坐标
3   if(x<screenWidth&&x>screenWidth-32
4           &&y<screenHeight&&y>screenHeight-16){                   //按下退出选项
5       activity.soundPool.stop(1);                                 //播放退出游戏声音
6       System.exit(0);                                             //退出游戏
7    }
8   if(status == 0){                                                //等待状态
9           status=1;                                               //将状态值改为 1
```

```
10            tt=new TimeThread(this);                          //创建时间线程
11            bgt=new BallGoThread(this);                       //创建球运动线程
12            tt.start();                                       //启动线程
13            bgt.start();                                      //启动线程
14    }
15    else if(status == 1){                                     //游戏进行中
16        bannerX = x;                                          //板 x 位置
17    }
18    else if(status==2||status==3){
19    if(x<32&&x>0&&y<screenHeight&&y>screenHeight-16){         //按下重玩选项
20          replay();                                           //调用重玩方法
21         } }
22        return super.onTouchEvent(event);
23    }
```

其中：

- 第 1～17 行主要为获取触摸点的 X、Y 坐标，若单击的是退出按钮，则退出游戏；若为等待状态，则开始游戏，若为游戏进行中，则调整板的 X 位置。
- 第 19～22 行为单击重玩按钮时，调用重玩方法。

上述代码已经介绍了本程序的 GameView 类的开发，接下来介绍控制球运动的线程 BallGoThread 类的开发，代码如下。

代码位置：见光盘源代码/第 10 章/Sample10_13/src/com/bn/ex10m/BallGoThread.java。

```
1   package com.bn.chap10.ballgame;
2   public class BallGoThread extends Thread{                 //游戏过程中移动球的线程
3       GameView father;                                      //声明引用
4       boolean flag=true;                                    //标志位
5       public BallGoThread(GameView father){
6           this.father=father;                               //获取引用
7       }
8       public void run(){                                    //重写的方法
9           while(flag){
10              switch(father.direction){
11                  case 0:                                   //右上
12                    father.ballx=father.ballx+father.ballSpan; //按当前方向移动球
13                    father.bally=father.bally-father.ballSpan;
14                    if(father.ballx>=father.screenWidth-father.ballSize) {
                                                              //碰到右壁
15                      father.direction=3;                   //左上
16                    }
17                    else if(father.bally<=0){             //碰到上壁
18                      father.direction=1;                 //右下
19                    }
20                  break;
21                  case 3:                                   //左上
22                    father.ballx=father.ballx-father.ballSpan; //按当前方向移动球
23                    father.bally=father.bally-father.ballSpan;
24                    if(father.ballx<=0){                  //碰到左壁
25                      father.direction=0;                 //右上
26                    }
27                    else if(father.bally<=0) {            //碰到上壁
28                      father.direction=2;                 //左下
29                    }
30                  break;
31                  case 1:                                   //右下
32                    father.ballx=father.ballx+father.ballSpan; //按当前方向移动球
33                    father.bally=father.bally+father.ballSpan;
```

```
34                      if(father.bally>=
35                            father.screenHeight-father.bannerHeight
36                            -father.bottomSpance-father.ballSize){  //碰到下壁
37                          checkCollision(1);                        //检测是否在板上
38                      }
39                      else if(father.ballx>=father.screenWidth-father.ballSize)
{//碰到右壁
40                         father.direction=2;                        //左下
41                      }
42                    break;
43                    case 2:                                         //左下
44                      father.ballx=father.ballx-father.ballSpan;     //得到x坐标
45                      father.bally=father.bally+father.ballSpan;     //得到y坐标
46                      if(father.bally>=
47                            father.screenHeight-father.bannerHeight
48                            -father.bottomSpance-father.ballSize){  //碰到下壁
49                         checkCollision(2);                         //检测是否在板上
50                      }
51                      else if(father.ballx<=0){                     //碰到左壁
52                         father.direction=1;                        //右下
53                      }
54                    break;                                          //退出
55                }
56                try{
57                    Thread.sleep(80);                               //线程睡眠80毫秒
58                }
59                catch(Exception e){e.printStackTrace();}          //打印异常
60            }}
61        public void checkCollision(int direction){                //声明方法
62            if(father.ballx>=father.bannerX-father.ballSize&&
63              father.ballx<=
64                father.bannerX+father.bannerWidth){                 //碰到板
65                switch(direction){
66                    case 1:                                         //direction为1
67                      father.direction=0;                           //改变方向为右上
68                    break;
69                    case 2:                                         //direction为2
70                      father.direction=3;                           //改变方向为左上
71                    break;
72                }
73            }
74            else{                                    //没有碰到板,游戏失败
75                father.tt.flag=false;                //时间线程标志位设为false
76                father.bgt.flag=false;               //球运动线程标志位设为false
77                father.status=2;                     //设置为游戏结束
78            }}
79    }
```

其中：

- 第 11～30 行主要完成的是判断球向右上或左上移动，按照小球的方向移动小球。若小球向右上运动碰到右壁，改变小球方向为左上，若碰到上壁，改变小球方向为左下；若小球向左上运动碰到左壁，改变小球方向为右上，若碰到上壁，改变小球方向为右下。
- 第 31～54 行主要完成的是球向左下或右下移动，按照小球的方向移动小球。若小球向左下移动碰到左壁，改变小球方向为右下，然后判断是否在板范围内，若在范围内，则该小球运动方向为右上或左上；同理，小球向右下运动碰到右壁，改变小球方向为左下，然后判断是否在板的范围内，若在范围内，则该小球运动方向为右上或左上。

- 第 61～79 行为检测是否在板上的方法，若处于板的范围则改变球运动方向，若不在范围内，将时间线程标志位和球运动标志位设为 false，并将状态值设为 2，游戏结束。

上述代码介绍的是本程序的线程类的开发，接下来介绍绘制的线程类的开发，代码如下。

代码位置：见光盘源代码/第 10 章/Sample10_13/src/com/bn/ex10m/GameViewDrawThread.java。

```
1   package com.bn.chap10.ballgame;                                    //声明包
2   import android.graphics.Canvas;                                    //导入相关包
3   import android.view.SurfaceHolder;
4   public class GameViewDrawThread extends Thread{                    //创建继承 Thread 的类
5       boolean flag = true;                                           //线程标志位
6       int sleepSpan = 100;
7       GameView gameView;                                             //声明引用
8       SurfaceHolder surfaceHolder;                                   //声明引用
9       public GameViewDrawThread(GameView gameView){
10          this.gameView = gameView;                                  //获取引用
11          this.surfaceHolder = gameView.getHolder();                 //创建对象
12      }
13      public void run(){
14          Canvas c;                                                  //声明引用
15          while (this.flag) {
16           c = null;                                                 //Canvas 为空
17           try {
18               //锁定整个画布，在内存要求比较高的情况下，建议参数不要为 null
19             c = this.surfaceHolder.lockCanvas(null);
20             synchronized (this.surfaceHolder) {
21               gameView.onDraw(c);                                   //绘制
22             }
23             } finally {
24               if (c != null) {                                      //Canvas 不为空
25                   this.surfaceHolder.unlockCanvasAndPost(c);        //释放锁
26               }
27             }
28             try{
29                 Thread.sleep(sleepSpan);                           //指定睡眠毫秒数
30             }
31             catch(Exception e){                                    //捕获异常
32                 e.printStackTrace();                               //打印堆栈信息
33  }}}
```

其中：

- 第 1～12 行表示声明成员变量以及创建构造器，在构造器内获取 GameView 引用并创建 SurfaceHolder 对象。
- 第 13～33 行是对线程类的 run 方法的重写，主要完成的是获取和锁定画布后绘制界面，绘制结束后解除锁定，重复进行，直到标志位设为 false 为止。

上述代码介绍的是本程序的绘制线程类的开发，接下来介绍的是控制时间的线程类，代码如下。

代码位置：见光盘源代码/第 10 章/Sample10_13/src/com/bn/ex10m/TimeThread.java。

```
1   package com.bn.chap10.ballgame;
2   public class TimeThread extends Thread{                            //计算生存时间的线程
3       int highest = 40;                                              //胜利值
4       GameView gameView;                                             //声明引用
5       boolean flag=true;                                             //线程标志位
6       public TimeThread(GameView gameView){
7           this.gameView=gameView;                                    //获取引用
```

```
8       }
9       public void run(){                                    //重写的方法
10          while(flag){                                      //flag 为 true 时
11              gameView.score++;
12              if(gameView.score==highest){                  //判断 score
13                  gameView.status=3;                        //游戏胜利
14                  gameView.tt.flag=false;                   //标志位设为 false
15                  gameView.bgt.flag=false;                  //标志位设为 false
16              }
17              try{
18                  Thread.sleep(1000);                       //线程睡眠 1 秒
19              }
20              catch(Exception e){                           //捕获异常
21                  e.printStackTrace();                      //打印异常
22              }}}}
```

其中：

- 第 9～21 行主要完成的是增加游戏时间，若 score 等于最大时间时，则表示游戏成功，将时间线程标志位和球运动线程标志位设为 false。

实例 14　音乐播放器

在前面介绍了如何制作视频播放器，而音乐播放器又是手机中不可或缺的一项。在本节将对音乐播放器进行详细介绍。

【实例描述】

本软件的主界面包含两个图片按钮，其中一个能够播放音乐和暂停播放中的音乐，另一个按钮实现的是停止正在播放的音乐。在能够正确播放音乐之前，首先需要在手机 SD 卡中添加歌曲，否则播放器将无法播放音乐。

本实例的运行效果如 10-14-1 所示。

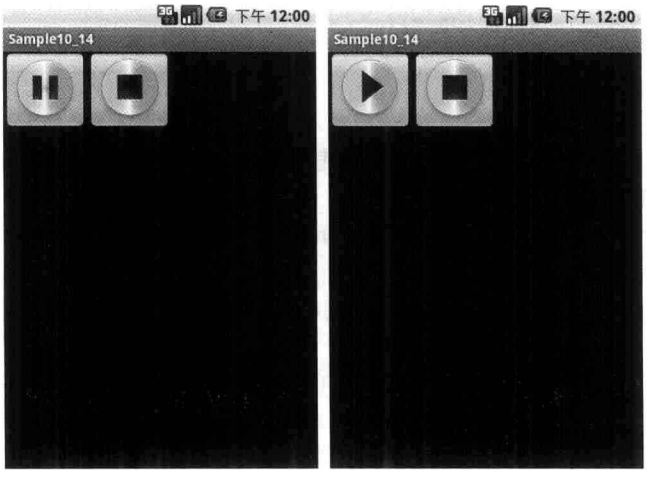


图 10-14-1　音乐播放器截图

提示：在图 10-14-1 中从左至右依次表示准备播放的界面，以及为播放音乐时的界面。

【实现过程】

在本软件中将对 Receiver 和 BroadcastReceiver 进行结合使用，首先在 AndroidManifest.xml 中注册 Receiver，实现能够在后台播放音乐的功能。

通过 CommandReceiver 类，接收前台 Activity 发送播放命令的 Intent。在 CommandReceiver 类中，有后台 Service 注册的接收前台控制命令 Intent 的 Receiver，并广播播放状态变化 Intent，UIUpdateReceiver 接收，从而更新 Activity 中界面的显示，其工作原理图如图 10-14-2 所示。

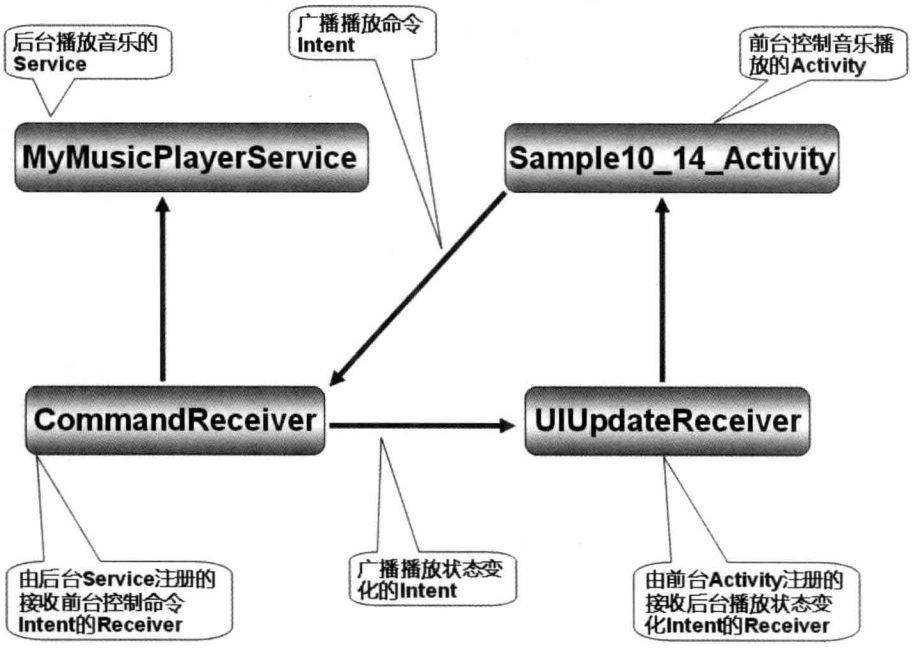


图 10-14-2　工作原理图

【代码解析】

经过上面的理论介绍，接下来要介绍的是该软件的核心代码，首先介绍的是 AndroidManifest.xml 文件，在该文件中需要注册 Service，否则软件将无法正常运行，其代码如下。

代码位置：见光盘源代码/第 10 章/Sample10_10/AndroidManifest.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>                    <!--版本号和编码方式-->
2   <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3       package="com.bn.ex10m"
4       android:versionCode="1"
5       android:versionName="1.0">                            <!--版本名称-->
6     <application android:icon="@drawable/icon" android:label="@string/app_name">
7       <activity android:name=".Sample10_14_Activity"
8               android:label="@string/app_name">             <!--显示的程序名称-->
9         <intent-filter>
10          <action android:name="android.intent.action.MAIN" />
                                                              <!--标识程序开始-->
11          <category android:name="android.intent.category.LAUNCHER" />
12        </intent-filter>
13      </activity>
14      <service android:name=".MyMusicPlayerService" android:process=":remote"/>
15    </application>
```

```
16      <uses-sdk android:minSdkVersion="7" />                    <!--设置 SDK 的值-->
17  </manifest>
```

提示：上述代码中第 14 行为声明 Service，并设置 process 为 remote。

上述代码介绍本程序的 AndroidManifest.xml 的开发，接下来介绍布局的开发，代码如下。

代码位置：见光盘源代码/第 10 章/Sample10_11/ res/layout/main.xml，详见源代码。

主界面 main.xml 中向 LinearLayout 中添加两个 ImageButton 按钮。

接下来要介绍的是该软件中的常量类，代码如下。

代码位置：见光盘源代码/第 10 章/Sample10_14/src/com/bn/ex10n/Constant.java。

```
1   package com.bn.chap10.mp;
2   public class Constant {                                    //包装各种常量的常量类
3     //表示不同命令的常量
4     public static final int COMMAND_PLAY=0;                  //播放命令
5     public static final int COMMAND_PAUSE=1;                 //暂停命令
6     public static final int COMMAND_STOP=2;                  //停止命令
7     //表示不同状态的常量
8     public static final int STATUS_PLAY=0;                   //播放状态
9     public static final int STATUS_PAUSE=1;                  //暂停状态
10    public static final int STATUS_STOP=2;                   //停止状态
11    public static final String MUSIC_CONTROL="TinyPlayer.ACTION_CONTROL";
                                                               //表示控制命令的 Action
12    public static final String UPDATE_STATUS="TinyPlayer.ACTION_UPDATE";
                                                               //表示更新界面的 Action
13  }
```

提示：常量类中各常量所代表的含义请参看源代码注释。

上述代码主要是对本程序用到的所有常量的声明，在开发中需要把常量全部放到该类中，接下来要介绍的是主类 Sample10_14_Activity 的实现，代码如下。

代码位置：见光盘源代码/第 10 章/Sample10_14/src/com/bn/ex10n/Sample10_14_Activity.java。

```
1   package com.bn.chap10.mp;                                  //声明包
2   import android.app.Activity;                               //导入相关包
3   ......// 该处省略了部分类的导入，读者可自行查看随书光盘中源代码
4   import android.widget.ImageButton;                         //导入相关包
5   public class Sample10_14_Activity extends Activity {
6     String currentPlay="/sdcard/zdan.mp3";                   //播放路径
7     UIUpdateReceiver uiur;                                   //界面更新 Intent 的接收者
8     public void onCreate(Bundle savedInstanceState) {
9       super.onCreate(savedInstanceState);
10      setContentView(R.layout.main);                         //设置当前界面
11      ImageButton
ibPlayPause=(ImageButton)findViewById(R.id.ImageButtonPlayPause); //获取引用
12      ImageButton ibStop=(ImageButton)findViewById(R.id.ImageButtonStop); //
获取停止按钮的引用
13      uiur=new UIUpdateReceiver(this);                       //创建界面更新 Intent 接收者
14      ibStop.setOnClickListener(                             //为停止按钮添加监听器
15        new OnClickListener(){
16          public void onClick(View v) {                      //重写的方法
17            Intent intent=new Intent(Constant.MUSIC_CONTROL); //创建对象
18            intent.putExtra("cmd", Constant.COMMAND_STOP);
19            Sample10_14_Activity.this.sendBroadcast(intent);
```

```
                                                        //发送停止命令广播
20                  }});
21          ibPlayPause.setOnClickListener(                 //播放/暂停按钮添加监听器
22            new OnClickListener(){                            //匿名内部类
23                public void onClick(View v) {                 //重写的方法
24                  if(uiur.status==Constant.STATUS_PLAY){ //若当前为播放状态
25                      Intent intent=new Intent(Constant.MUSIC_CONTROL);
26                      intent.putExtra("cmd", Constant.COMMAND_PAUSE);
27                      Sample10_14_Activity.this.sendBroadcast(intent);
                                                        //发出暂停命令
28                  }
29                  else if(uiur.status==Constant.STATUS_STOP) //若当前为停止状态
30                  {
31                      Intent intent=new Intent(Constant.MUSIC_CONTROL);
32                      intent.putExtra("cmd", Constant.COMMAND_PLAY);
                                                        //添加数据
33                      intent.putExtra("path", currentPlay);  //添加数据
34                      Sample10_14_Activity.this.sendBroadcast(intent);
                                                    //发出播放命令和播放路径
35                  }
36                  else{                                   //若当前为暂停状态
37                      Intent intent=new Intent(Constant.MUSIC_CONTROL);
38                      intent.putExtra("cmd", Constant.COMMAND_PLAY);
                                                        //添加数据
39                      Sample10_14_Activity.this.sendBroadcast(intent);
                                                        //发出继续播放命令
40                  }}});
41          //动态注册接收播放、暂停、停止状态更新 Intent 的 UIUpdateReceiver
42          IntentFilter filter=new IntentFilter();     //创建 Intent 对象
43          filter.addAction(Constant.UPDATE_STATUS);   //添加监听
44          this.registerReceiver(uiur, filter);
45          this.startService(new Intent(this,MyMusicPlayerService.java));
                                                    //发出后台 Service 的 Intent
46      }}
```

其中：

- 第 6～13 行为声明播放路径和界面更新 Intent 的接收者，设置当前界面为 main.xml，初始化图片按钮。
- 第 14～20 行为停止按钮监听器，当单击停止按钮时，发送停止播放的广播命令。
- 第 21～40 行为播放或暂停按钮监听器，当单击该按钮时判断是在播放状态还是在暂停状态，若处于播放状态则发送暂停命令，若处于暂停状态，则发送播放命令。
- 第 41～46 行为动态注册接收播放、暂停、停止状态更新 Intent 的 UIUpdateReceiver，并发出启动后台 Service 的 Intent。

上述代码主要是对本程序的 Activity 类的开发，接下来要介绍的是继承 BroadcastReceiver 类的 UIUpdateReceiver 类，该类的代码如下所列。

代码位置：见光盘源代码/第 10 章/Sample10_14/src/com/bn/ex10n/UIUpdateReceiver.java。

```
1   package com.bn.chap10.mp;                               //声明包
2   import android.app.Activity;                            //导入相关包
3   import android.content.BroadcastReceiver;               //导入相关包
4   import android.content.Context;                         //导入相关包
5   import android.content.Intent;                          //导入相关包
6   import android.widget.ImageButton;
7   public class UIUpdateReceiver extends BroadcastReceiver {
                                                    //更新界面显示的 Intent 接收者
```

```
8       int status=Constant.STATUS_STOP;                    //初始状态为停止状态
9       Activity ac;                                        //此 Receiver 对应的 Activity
10      public UIUpdateReceiver(Activity ac){
11       this.ac=ac;                                        //获取引用
12      }
13       public void onReceive(Context context, Intent intent) {
14           status=intent.getIntExtra("status", -1); //从接收的 Intent 中获取状态值
15           ImageButton ib=(ImageButton)ac.findViewById(R.id.
ImageButtonPlayPause);//获取按钮的引用
16           switch(status){
17            case Constant.STATUS_PLAY:                    //更新到播放状态
18                ib.setImageResource(R.drawable.pause);//将图片换为暂停的提示图片
19            break;
20            case Constant.STATUS_STOP:                    //更新到停止、暂停状态
21            case Constant.STATUS_PAUSE:
22                 ib.setImageResource(R.drawable.play);//则将图片换为播放的提示图片
23            break;
24           }}}
```

其中：

- 第 2～12 行为该类中相关包的导入，设置初始状态为停止状态，在构造器内获取 Activity 的引用。
- 第 13～23 行为重写 onReciver()方法，首先从接收的 Intent 中获取状态值，获取按钮的引用，然后判断若更新到播放状态则更换暂停图片，若更新到停止或暂停状态则将图片换为播放的提示图片。

上述代码主要是对本程序创建继承 BroadcastReceiver 类的开发，接下来介绍继承自 Service 类的 MyMusicPlayerService 类的设计与实现，代码如下。

代码位置：见光盘源代码/第 10 章/Sample10_14/src/com/bn/ex10n/MyMusicPlayerService.java。

```
1   package com.bn.chap10.mp;                               //声明包
2   import android.app.Service;                             //导入相关包
3   import android.content.Intent;                          //导入相关包
4   import android.content.IntentFilter;
5   import android.media.MediaPlayer;                       //导入相关包
6   import android.os.IBinder;                              //导入相关包
7   public class MyMusicPlayerService extends Service {
8       CommandReceiver cr;                                 //命令 Intent 接收者对象引用
9       public IBinder onBind(Intent intent){               //直接返回 null
10          return null;
11      }
12      public void onCreate(){                             //重写的方法
13          super.onCreate();                               //调用父类
14          cr=new CommandReceiver();                       //创建命令 Intent 接收者对象
15          cr.mp=new MediaPlayer();                        //创建媒体播放器对象
16          cr.status=Constant.STATUS_STOP;                 //初始状态为停止状态
17          //动态注册接收播放、暂停、停止命令 Intent 的 CommandReceiver
18          IntentFilter filter=new IntentFilter();         //创建 Intent 对象
19          filter.addAction(Constant.MUSIC_CONTROL);       //添加 Action
20          this.registerReceiver(cr, filter);
21      }
22      public void onDestroy(){                            //重写的方法
23          super.onDestroy();                              //调用父类
24          //取消注册接收播放、暂停、停止命令 Intent 的 CommandReceiver
25          this.unregisterReceiver(cr);
26          cr.mp.release();                                //释放播放器对象
```

```
27        }
28        public void onStart(Intent intent, int id){
29            cr.updateUI(this.getApplicationContext());       //更新界面状态
30        }}
```

其中：

- 第 9～11 行为重写 onBind()方法，因为在本例中不需要 Bind 功能，在这里直接返回 null。
- 第 12～27 行为重写 onCreate()方法和 onDestroy()方法，在 onCreate()方法中创建命令 Intent 接收者对象和创建媒体播放器对象，将初始状态设为停止状态，最后动态注册接收播放、暂停、停止命令 Intent 的 CommandReceiver。在 onDestroy()方法中取消注册接收播放、暂停、停止命令 Intent 的 CommandReceiver，释放播放器对象。
- 第 28～29 行为重写 onStart()方法，在该方法中主要完成更新界面的状态。

小结

本章主要介绍多媒体服务功能的开发。首先介绍的是图形图像的处理，然后介绍的是音频视频的处理，最后介绍的是一个简单的 2D 游戏以及简单的音乐播放软件。通过对本章的学习，读者应该具备自行开发简单 2D 游戏的能力。

第 11 章　Android 手机的 3D 世界

在本章中详细介绍了 Android 系统平台下手机基础 3D 实现的相关知识，下面将通过对几个小开发实例的讲解，来向读者介绍具体知识的应用开发。

实例 1　三角形的绘制

在本小节中，将向读者介绍 3D 世界中最基本物体三角形的绘制方法。

【实例描述】

本实例实现了两个在开启背面剪裁情况下的三角形的创建，通过对该物体创建过程的讲解，向读者介绍 3D 世界中最简单物体三角形的构造过程。

本实例运行效果如图 11-1、图 11-2 所示。

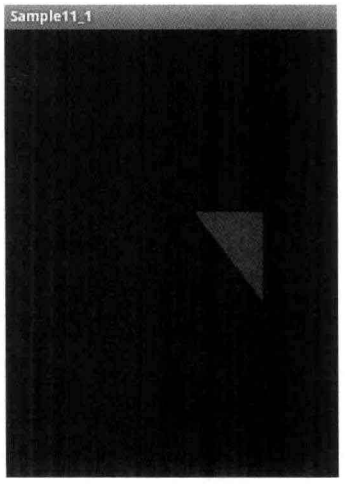


图 11-1　起始界面图

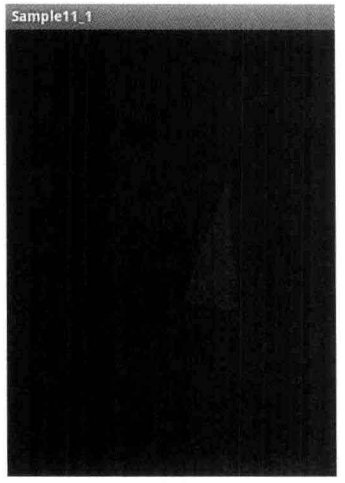


图 11-2　滑动界面后场景图

提示：当本实例运行时，首先进入的是如图 11-1 所示的起始界面，场景中显示了一个红色的三角形。当用户触控屏幕，用手指横向滑动屏幕时，场景会发生旋转，旋转一定角度时，场景中会显示另一个蓝色的三角形如图 11-2 所示。

【实现过程】

在 OpenGL ES 中只支持三角形，主要是出于性能的原因。从能力上来说只支持三角形与支持多种多边形是一样的。因为任何多边形都可以拆分成多个三角形，三角形的正反面是这样确定的，当面对一个三角形时，若顶点的顺序是逆时针的则位于三角形的正面，反之则在反面。

在本实例中，应用了背面剪裁技术，起始状态时，蓝色三角形不可见，因为显示的是蓝色三角形的背面，背面剪裁技术就是不显示物体背面效果，同时在本实例中应用了屏幕触控技术，OpenGL ES 程序在执行真正的绘制工作之前有很多准备工作，包括如下。

（1）清除颜色缓存与深度缓存

```
gl.glClear(GL10.GL_COLOR_BUFFER_BIT|GL10.GL_GL_DEPTH_BUFFER_BIT);
```

OpenGL ES 保存了一系列缓存（buffers），即用于绘图各方面的内存块。颜色缓存保存当前帧各像素的颜色。基本上就是用户在屏幕上看到的情况。深度缓存（有时也称为“z-buffer”），保存每个潜在像素离观察者距离的信息。使用此信息可以确定一个像素是否需要被绘制出来。这两个缓存是 OpenGL ES 中最常见的缓存。还有其他类型的一些缓存，如模板缓存和累积缓存。

（2）加载单位变换矩阵

```
gl.glMatrixMode(GL10.GL_MODELVIEW);//设置当前矩阵为模式矩阵，gl.glLoadIdentity();//设置当前矩阵为单位矩阵。
```

此工作将清除虚拟世界中的一切平移、旋转、缩放或其他变化，并将观察者置于原点。

【代码解析】

首先向读者介绍本项目开发的 Activity 类的开发过程。

代码位置：Sample11_1/src/com.bn.chap11.sjx/Sample11_1_Activity.java

```
1   package com.bn.chap11.sjx;                          //包名
2   import android.app.Activity;                        //包引用，该处有所省略，读者可自行查看源代码
3   import android.os.Bundle;
4   public class Sample11_1_Activity extends Activity {
5    private MySurfaceView mGLSurfaceView;                          //声明 3D 界面对象
6    protected void onCreate(Bundle savedInstanceState) {    //重写 onCreate()方法
7     super.onCreate(savedInstanceState);
8     mGLSurfaceView = new MySurfaceView(this);                     //创建 3D 界面对象
9     setContentView(mGLSurfaceView);                               //呈现 3D 界面
10    mGLSurfaceView.requestFocus();                                //获取焦点
11    mGLSurfaceView.setFocusableInTouchMode(true);                 //设置为可触控
12   }
13   protected void onResume() {                                    //重写 onResume()恢复方法
14    super.onResume();
15    mGLSurfaceView.onResume();                                    //界面恢复
16   }
17   protected void onPause() {                                     //重写 onPause()暂停方法
18     super.onPause();
19    mGLSurfaceView.onPause();                                     //界面暂停
20  }   }
```

其中：

- 第 5 行为声明 MySurfaceView 3D 界面对象。第 6～12 行为重写 onCreate()方法，在其中创建了界面对象，并通过调用 setContentView()方法跳转到 3D 界面。
- 第 13～16 行为重写的 onResume()方法，用于在软件运行暂停后恢复界面显示。第 17～20 行为重写 onPause()方法，用于在软件运行中暂停界面显示。

接下来向读者介绍本项目开发的界面实现类的开发过程。

代码位置：Sample11_1/src/com.bn.chap11.sjx/ MySurfaceView.java

```
1   package com.bn.chap11.sjx;                          //包名
2   import android.opengl.GLSurfaceView;    //包引用，该处有所省略，读者可自行查阅随书光盘中的源代码
```

```
class MySurfaceView extends GLSurfaceView {
  private final float TOUCH_SCALE_FACTOR = 180.0f/320;   //角度缩放比例
private SceneRenderer mRenderer;                          //场景渲染器
private float mPreviousY;                                 //上次的触控位置 Y 坐标
private float mPreviousX;                                 //上次的触控位置 X 坐标
public MySurfaceView(Context context) {
  super(context);
  mRenderer = new SceneRenderer();                        //创建场景渲染器
  setRenderer(mRenderer);                                 //设置渲染器
  setRenderMode(GLSurfaceView.RENDERMODE_CONTINUOUSLY);//设置渲染模式为主动渲染
}
public boolean onTouchEvent(MotionEvent e) {              //触摸事件回调方法
  float y = e.getY();                                     //获取当前触控点坐标
  float x = e.getX();
  switch (e.getAction()) {                                //判断触控事件
    case MotionEvent.ACTION_MOVE:                       //触控事件为屏幕滑动事件
      float dy = y - mPreviousY;                        //计算触控笔 Y 位移
      float dx = x - mPreviousX;                        //计算触控笔 X 位移
      mRenderer.tp.yAngle += dx * TOUCH_SCALE_FACTOR;//设置沿 x 轴旋转角度
      mRenderer.tp.zAngle+= dy * TOUCH_SCALE_FACTOR; //设置沿 z 轴旋转角度
      requestRender();                                  //重绘画面
}
mPreviousY = y;                                         //记录触控笔位置
mPreviousX = x;                                         //记录触控笔位置
return true;
}
private class SceneRenderer implements GLSurfaceView.Renderer{ //渲染器内部类
  TrianglePair tp=new TrianglePair();                   //创建三角形对象
  public SceneRenderer(){                               //空构造器
  }
  public void onDrawFrame(GL10 gl) {                    //重写 onDrawFrame()方法
    gl.glEnable(GL10.GL_CULL_FACE);                     //设置为打开背面剪裁
    gl.glShadeModel(GL10.GL_FLAT);                      //设置着色模型为不平滑着色
    gl.glClear(GL10.GL_COLOR_BUFFER_BIT
    GL 10.GL_DEPTH_BUFFER_BIT);                         //清除颜色缓存于深度缓存
    gl.glMatrixMode(GL10.GL_MODELVIEW);                 //设置当前矩阵为模式矩阵
    gl.glLoadIdentity();                                //设置当前矩阵为单位矩阵
    gl.glTranslatef(0, -1f, -3f);                       //平移操作
    tp.drawSelf(gl);                                    //绘制图形
}
public void onSurfaceChanged(GL10 gl, int width, int height) {
                                                      //重写 onSurfaceChanged()
  gl.glViewport(0, 0, width, height);                   //设置视窗大小及位置
  gl.glMatrixMode(GL10.GL_PROJECTION);                  //设置当前矩阵为投影矩阵
  gl.glLoadIdentity();                                  //设置当前矩阵为单位矩阵
  float ratio = (float) width / height;                 //计算透视投影的比例
  gl.glFrustumf(-ratio, ratio, -1, 1, 1, 10);           //透视投影矩阵
}
public void onSurfaceCreated(GL10 gl, EGLConfig config) {
                                                      //重写 onSurfaceCreated()
  gl.glDisable(GL10.GL_DITHER);                         //关闭抗抖动
  gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT,
  GL10.GL_FASTEST);              //设置特定 Hint 项目的模式，这里设置为使用快速模式
  gl.glClearColor(0,0,0,0);                             //设置屏幕背景色为黑色 RGBA
 gl.glEnable(GL10.GL_DEPTH_TEST);                       //启用深度测试
 } } }
```

其中：

- 第 4～7 行为声明成员变量及创建常量数据。其中 mPreviousX、mPreviousY 用于记录上次手机屏幕触控点的坐标位置。
- 第 8～13 行为该界面类的构造器部分，在其中创建了渲染器对象，并设置了渲染器属性。
- 第 14～28 行为屏幕触控事件回调方法，其中首先获取当前触控屏幕的坐标，然后根据屏幕滑动的距离计算场景中物体的旋转的角度。
- 第 39～56 行为渲染器内部类的构造方法。其中通过重写 onSurfaceCreated()方法实现了 3D 场景的创建，通过重写 onDrawFrame()方法实现了 3D 场景的绘制，通过 onSurfaceChanged()方法实现了界面变化时场景的设置。

然后向读者介绍本项目开发中物体构造类的开发过程。

代码位置：Sample11_1/src/com.bn.chap11.sjx/TrianglePair.java

```
1   package com.bn.chap11.sjx;                    //包名
2   import java.nio.ByteBuffer; //包引用，该处有所省略，读者可自行查阅随书光盘中的源代码
3   public class TrianglePair {
4     private IntBuffer  mVertexBuffer;           //顶点坐标数据缓冲
5     private IntBuffer  mColorBuffer;            //顶点着色数据缓冲
6     private ByteBuffer  mIndexBuffer;           //顶点构建索引数据缓冲
7     int vCount=0; int iCount=0;
8     float yAngle=0;                             //绕 y 轴旋转的角度
9     float zAngle=0;                             //绕 z 轴旋转的角度
10    public TrianglePair() {                     //顶点坐标数据的初始化
11      vCount=6;       final int UNIT_SIZE=10000;
12      int vertices[]=new int[]       {
13       8*UNIT_SIZE,10*UNIT_SIZE,0,     2*UNIT_SIZE,2*UNIT_SIZE,0,
14       8*UNIT_SIZE,2*UNIT_SIZE,0,      8*UNIT_SIZE,2*UNIT_SIZE,0,
15       8*UNIT_SIZE,10*UNIT_SIZE,0,     2*UNIT_SIZE,10*UNIT_SIZE,0
16  };
17    ByteBuffer vbb = ByteBuffer.allocateDirect(vertices.length*4);
                                                    //创建顶点坐标数据缓冲
18    vbb.order(ByteOrder.nativeOrder());           //设置字节顺序
19    mVertexBuffer = vbb.asIntBuffer();            //转换为 int 型缓冲
20    mVertexBuffer.put(vertices);                  //向缓冲区中放入顶点坐标数据
21    mVertexBuffer.position(0);                    //设置缓冲区起始位置
22    final int one = 65535;
23    int colors[]=new int[] {                      //顶点颜色值数组
24    one,one,one,0,  0,0,one,0,0,0,one,0,
25    one,one,one,0,one,0,0,0,one,0,0,0
26    };
27    ByteBuffer cbb = ByteBuffer.allocateDirect(colors.length*4);
                                                    //创建顶点着色数据缓冲
28    cbb.order(ByteOrder.nativeOrder());           //设置字节顺序
29    mColorBuffer = cbb.asIntBuffer();             //转换为 int 型缓冲
30    mColorBuffer.put(colors);                     //向缓冲区中放入顶点着色数据
31    mColorBuffer.position(0);                     //设置缓冲区起始位置
32    iCount=6;                                     //索引点数量
33    byte indices[]=new byte[]{                    //创建 byte 数组，用于存放索引值
34       0,1,2,3,4,5
35    };
36    mIndexBuffer = ByteBuffer.allocateDirect(indices.length);
                                                  //创建三角形构造索引数据缓冲
37    mIndexBuffer.put(indices);                    //向缓冲区中放入三角形构造索引数据
38    mIndexBuffer.position(0);                     //设置缓冲区起始位置
```

```
39   }
40   public void drawSelf(GL10 gl){
41     gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);      //启用顶点坐标数组
42     gl.glEnableClientState(GL10.GL_COLOR_ARRAY);       //启用顶点颜色数组
43     gl.glRotatef(yAngle, 0, 1, 0);
44     gl.glRotatef(zAngle, 0, 0, 1);                     //绘制图片
45     gl.glVertexPointer                                 //为画笔指定顶点坐标数据
46     (
47           3,                                        //每个顶点的坐标数量为 3
48           GL10.GL_FIXED,                            //顶点坐标值的类型为 GL_FIXED
49           0,                                        //连续顶点坐标数据之间的间隔
50           mVertexBuffer                             //顶点坐标数据
51     );
52   gl.glColorPointer                                  //为画笔指定顶点着色数据
53   (
54           4,                                  //设置颜色的组成成分，必须为 4RGBA
55           GL10.GL_FIXED,                            //顶点颜色值的类型为 GL_FIXED
56           0,                                        //连续顶点着色数据之间的间隔
57           mColorBuffer                              //顶点着色数据
58    );
59    gl.glDrawElements                                 //绘制图形
60    (
61           GL10.GL_TRIANGLES,                        //以三角形方式填充
62           iCount,                             //一共 icount/3 个三角形，iCount 个顶点
63           GL10.GL_UNSIGNED_BYTE,                    //索引值的尺寸
64           mIndexBuffer                              //索引值数据
65    );
66    }     }
```

其中：

- 第 4～9 行为声明成员变量，包括顶点数据缓存对象、顶点着色数据缓存对象和顶点索引数据缓存对象和顶点数量记录对象声明，顶点索引数量记录对象声明，以及场景选择角度对象声明。
- 第 10～39 行为该类构造的开发，其中，第 11～21 行为获取顶点数组并将其存入顶点数据缓存；第 22～31 行为获取顶点着色数组并将其存入顶点着色数据缓存；第 32～38 行为获取顶点索引数据并将其存入顶点索引数据缓存中。
- 第 40～66 行为绘制场景的方法。其中第 45～51 行为场景指定顶点坐标数据；第 52～58 行为场景指定顶点着色数据；第 59～65 行为场景指定顶点索引数据，并绘制场景。

实例 2　立方体的绘制

在本小节中，将向读者介绍 3D 世界中常用立方体物体的绘制方法。

【实例描述】

本实例实现了一个平滑着色后的立方体的创建，并通过对该物体的创建过程的讲解，向读者介绍立方体的构造过程。

本实例运行效果如图 11-3 所示。

提示：在图 11-3 中显示了一个红蓝相间的立方体，其可以用手指滑动手机屏幕的方式来改变物体的呈现角度，从而达到不同的视觉效果。

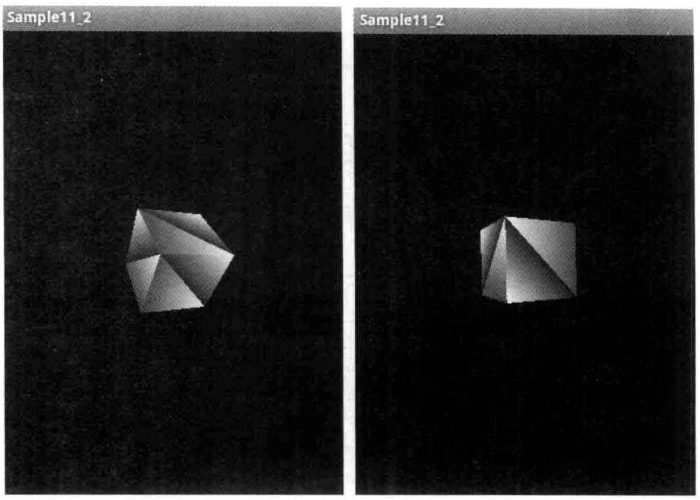


图 11-3　立方体场景图

【实现过程】

OpenGL ES 中只允许使用三角形进行填充。本实例中构造的是立方体，其有 6 个面，每个面是一个矩形，可以切分成两个三角形。因此，在 OpenGL ES 中，立方体可以用 12 个三角形填充。

同时在本实例中应用了屏幕触控技术。

【代码解析】

由于在本实例立方体的创建过程中，Activity 类与界面实现类与三角形的绘制方法相似，因此在这里不再赘述。

下面向读者介绍的是本项目开发中物体构造类的开发过程。

代码位置：Sample11_2/src/com.bn.chap11.mx/ Crate.java

```
1   package com.bn.chap11.mx;                    //包名
2   import java.nio.ByteBuffer;//包引用，该处有所省略，读者可自行查看随书光盘中的看源代码
3   public class Crate {
4     private IntBuffer  mVertexBuffer;    //顶点坐标数据缓冲
5     private IntBuffer  mColorBuffer;     //顶点着色数据缓冲
6     int vCount=0; float yAngle=0;        //绕 y 轴旋转的角度
7     float zAngle=0;                      //绕 z 轴旋转的角度
8     public Crate(){
9         vCount=36;int size=5;final int UNIT_SIZE=10000; //顶点数量及局部变量设置
10        int vertices[]=new int[]{        //指定顶点坐标数据
11          size*UNIT_SIZE,-size*UNIT_SIZE,-size*UNIT_SIZE, //第 1 个三角形顶点坐标
12          size*UNIT_SIZE,-size*UNIT_SIZE,size*UNIT_SIZE,  //第 1 个三角形顶点坐标
13          size*UNIT_SIZE,size*UNIT_SIZE,-size*UNIT_SIZE,  //第 1 个三角形顶点坐标
14          size*UNIT_SIZE,size*UNIT_SIZE,-size*UNIT_SIZE,-
15          size*UNIT_SIZE,-size*UNIT_SIZE,size*UNIT_SIZE,  //第 2 个三角形顶点坐标
16          size*UNIT_SIZE,size*UNIT_SIZE,size*UNIT_SIZE,   //第 2 个三角形顶点坐标
17          size*UNIT_SIZE,size*UNIT_SIZE,-size*UNIT_SIZE,
18          size*UNIT_SIZE,size*UNIT_SIZE,size*UNIT_SIZE,   //第 3 个三角形顶点坐标
19          size*UNIT_SIZE,size*UNIT_SIZE,-size*UNIT_SIZE,  //第 3 个三角形顶点坐标
20          size*UNIT_SIZE,size*UNIT_SIZE,-size*UNIT_SIZE,
```

```
21          size*UNIT_SIZE,size*UNIT_SIZE,size*UNIT_SIZE,   //第 4 个三角形顶点坐标
22          size*UNIT_SIZE,size*UNIT_SIZE,size*UNIT_SIZE,   //第 4 个三角形顶点坐标
23          size*UNIT_SIZE,size*UNIT_SIZE,-size*UNIT_SIZE,  //第 5 个三角形顶点坐标
24          size*UNIT_SIZE,size*UNIT_SIZE,size*UNIT_SIZE,   //第 5 个三角形顶点坐标
25          size*UNIT_SIZE,-size*UNIT_SIZE,-size*UNIT_SIZE, //第 5 个三角形顶点坐标
26          size*UNIT_SIZE,-size*UNIT_SIZE,-size*UNIT_SIZE,
27          size*UNIT_SIZE,size*UNIT_SIZE,size*UNIT_SIZE,   //第 6 个三角形顶点坐标
28          size*UNIT_SIZE,-size*UNIT_SIZE,size*UNIT_SIZE,  //第 6 个三角形顶点坐标
29          size*UNIT_SIZE,size*UNIT_SIZE,size*UNIT_SIZE,
30          size*UNIT_SIZE,-size*UNIT_SIZE,size*UNIT_SIZE,  //第 7 个三角形顶点坐标
31          size*UNIT_SIZE,size*UNIT_SIZE,size*UNIT_SIZE,   //第 7 个三角形顶点坐标
32          size*UNIT_SIZE,size*UNIT_SIZE,size*UNIT_SIZE,
33          size*UNIT_SIZE,-size*UNIT_SIZE,size*UNIT_SIZE,  //第 8 个三角形顶点坐标
34          size*UNIT_SIZE,-size*UNIT_SIZE,size*UNIT_SIZE,  //第 8 个三角形顶点坐标
35          size*UNIT_SIZE,-size*UNIT_SIZE,size*UNIT_SIZE,  //第 9 个三角形顶点坐标
36          size*UNIT_SIZE,-size*UNIT_SIZE,-size*UNIT_SIZE, //第 9 个三角形顶点坐标
37          size*UNIT_SIZE,-size*UNIT_SIZE,size*UNIT_SIZE,  //第 9 个三角形顶点坐标
38          size*UNIT_SIZE,-size*UNIT_SIZE,size*UNIT_SIZE,-
39          size*UNIT_SIZE,-size*UNIT_SIZE,-size*UNIT_SIZE,  //第10个三角形顶点坐标
40          size*UNIT_SIZE,-size*UNIT_SIZE,-size*UNIT_SIZE,  //第10个三角形顶点坐标
41          size*UNIT_SIZE,-size*UNIT_SIZE,-size*UNIT_SIZE,-
42          size*UNIT_SIZE,size*UNIT_SIZE,-size*UNIT_SIZE, //第 11 个三角形顶点坐标
43          size*UNIT_SIZE,-size*UNIT_SIZE,-size*UNIT_SIZE,//第 11 个三角形顶点坐标
44          size*UNIT_SIZE,-size*UNIT_SIZE,-size*UNIT_SIZE,-
45          size*UNIT_SIZE,size*UNIT_SIZE,-size*UNIT_SIZE,//第 12 个三角形顶点坐标
46          size*UNIT_SIZE,size*UNIT_SIZE,-size*UNIT_SIZE //第 12 个三角形顶点坐标
47  };
48  ByteBuffer vbb = ByteBuffer.allocateDirect(vertices.length*4);
                                                          //创建顶点坐标数据缓冲
49  vbb.order(ByteOrder.nativeOrder());                   //设置字节顺序
50  mVertexBuffer = vbb.asIntBuffer();                    //转换为 int 型缓冲
51  mVertexBuffer.put(vertices);                        //向缓冲区中放入顶点坐标数据
52  mVertexBuffer.position(0);                            //设置缓冲区起始位置
53  final int one = 65535;
54  int colors[]=new int[]{                               //顶点颜色值数组
55     one,one,one,0,0,0,one,0, 0,0,one,0,                //第 1 个三角形各顶点 RGBA 值
56     one,one,one,0, one,0,0,0, one,0,0,0,                //第 2 个三角形各顶点 RGBA 值
57     one,one,one,0, 0,0,one,0, 0,0,one,0,                //第 3 个三角形各顶点 RGBA 值
58     one,one,one,0, one,0,0,0, one,0,0,0,                //第 4 个三角形各顶点 RGBA 值
59     one,one,one,0, 0,0,one,0, 0,0,one,0,                //第 5 个三角形各顶点 RGBA 值
60     one,one,one,0, one,0,0,0, one,0,0,0,                //第 6 个三角形各顶点 RGBA 值
61     one,one,one,0, 0,0,one,0, 0,0,one,0,                //第 7 个三角形各顶点 RGBA 值
62     one,one,one,0, one,0,0,0, one,0,0,0,                //第 8 个三角形各顶点 RGBA 值
63     one,one,one,0, 0,0,one,0, 0,0,one,0,                //第 9 个三角形各顶点 RGBA 值
64     one,one,one,0, one,0,0,0, one,0,0,0,              //第 10 个三角形各顶点 RGBA 值
65     one,one,one,0, 0,0,one,0, 0,0,one,0,              //第 11 个三角形各顶点 RGBA 值
66     one,one,one,0, one,0,0,0, one,0,0,0               //第 12 个三角形各顶点 RGBA 值
67  };
68    ByteBuffer cbb = ByteBuffer.allocateDirect(colors.length*4);
                                                        //创建顶点着色数据缓冲
69    cbb.order(ByteOrder.nativeOrder());               //设置字节顺序
70    mColorBuffer = cbb.asIntBuffer();                 //转换为 int 型缓冲
71    mColorBuffer.put(colors);                         //向缓冲区中放入顶点着色数据
72    mColorBuffer.position(0);                         //设置缓冲区起始位置
73  }
74  public void drawSelf(GL10 gl)    {
```

```
75      gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);          //启用顶点坐标数组
76      gl.glEnableClientState(GL10.GL_COLOR_ARRAY);           //启用顶点颜色数组
77      gl.glRotatef(yAngle, 0, 1, 0); gl.glRotatef(zAngle, 0, 0, 1);
78      gl.glVertexPointer(3,GL10.GL_FIXED,0, mVertexBuffer); //为画笔指定顶点坐标数据
79      gl.glColorPointer (4,GL10.GL_FIXED, 0, mColorBuffer );
                                                          //为画笔指定顶点着色数据
80      gl.glDrawArrays(GL10.GL_TRIANGLES, 0, vCount);         //绘制图形
81  }   }
```

其中：

- 第 4～7 行为声明成员变量，包括顶点数据缓存对象、顶点着色数据缓存对象和顶点索引数据缓存对象和顶点数量记录对象声明，顶点索引数量记录对象声明，以及场景选择角度对象声明。
- 第 10～39 行为该类构造的开发，本场景是由顶点法实现的。其中，第 9～40 行为获取顶点数组并将其存入顶点数据缓存；第 41～60 行为获取顶点着色数组并将其存入顶点着色数据缓存。
- 第 62～67 行为绘制场景的方法。其中 65 行为场景指定顶点坐标数据；第 66 行为场景指定顶点着色数据；第 67 行为场景绘制。

实例 3　球体的绘制

在本小节中，将向读者介绍 3D 世界中常用物体球体的绘制方法。

【实例描述】

本实例实现了一个平滑着色后的球体创建，通过对该物体的创建过程的讲解，向读者介绍球体的构造过程。

本实例运行效果如图 11-4 所示。

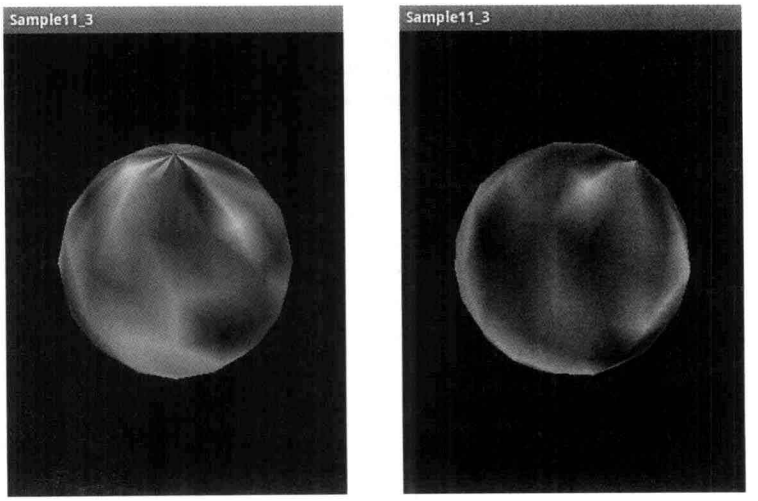


图 11-4　球体场景图

提示：在图 11-4 中显示了一个自动旋转的平滑球体，其可以用手指滑动手机屏幕的方式来改变物体的呈现角度，为用户提供不同球面的展示。

【实现过程】

球体是由多个三角形搭建而成的，因此需要开发人员自行编写生成组成球体的各个三角形顶点及索引的代码，代码的核心思想是首先将球体按一定的角度跨度进行经度与纬度方向的切分，然后依次计算每个特定经纬度处顶点的坐标，最后，将顶点按照一定的规律编排成三角形。

$x=r\cos\beta\cos\lambda$　　$y=r\cos\beta\sin\lambda$　　$z=r\sin\beta$　　$(-90<=\beta<=90;-180<=\lambda<=180)$　其中 β 为纬度，λ 为经度。

【代码解析】

由于在本实例立方体的创建过程中，Activity 类与三角形的绘制方法相似，因此在这里不再赘述。下面向读者介绍本项目开发中界面构造类的开发过程。

代码位置：Sample11_3/src/com.bn.chap11.qiu/ MySurfaceView.java

```
1   package com.bn.chap11.qiu;                    //包名
2   import android.opengl.GLSurfaceView;    //包引用，该处有所省略，读者可自行查看随书光盘中的源代码
3   class MySurfaceView extends GLSurfaceView {
4     private final float TOUCH_SCALE_FACTOR = 180.0f/320;  //角度缩放比例
5     private SceneRenderer mRenderer;                      //场景渲染器
6     private float mPreviousY;                             //上次的触控位置 Y 坐标
7     private float mPreviousX;                             //上次的触控位置 X 坐标
8     public MySurfaceView(Context context) {
9       super(context); mRenderer = new SceneRenderer();    //创建场景渲染器
10      setRenderer(mRenderer);                             //设置渲染器
11      setRenderMode(GLSurfaceView.RENDERMODE_CONTINUOUSLY);
                                                    //设置渲染模式为主动渲染
12    }
13    public boolean onTouchEvent(MotionEvent e) {          //触摸事件回调方法
14     float y = e.getY();      float x = e.getX();         //获取屏幕触控点坐标
15     switch (e.getAction()) {
16       case MotionEvent.ACTION_MOVE:                      //屏幕滑动事件
17         float dy = y - mPreviousY;float dx = x - mPreviousX; //计算触控笔 X、Y 位移
18         mRenderer.ball.mAngleY += dy * TOUCH_SCALE_FACTOR;//设置沿 Y 轴旋转角度
19         mRenderer.ball.mAngleZ += dx * TOUCH_SCALE_FACTOR;//设置沿 z 轴旋转角度
20         requestRender();                                 //重绘画面
21     }
22     mPreviousY =y;mPreviousX = x;                       //记录触控笔位置
23     return true;
24   }
25  private class SceneRenderer implements GLSurfaceView.Renderer{//渲染器内部类
26    Spheroid ball=new Spheroid(3);                          //创建球类对象
27    public SceneRenderer(){                                 //渲染器构造器
28       new Thread(){                                  //开启一个线程自动旋转球体
29         public void run(){                                //重写的方法
30           try {
31             Thread.sleep(1000);                    //休息 1000ms 再重绘
32  }
33  catch(Exception e) {                              //捕获异常
34    e.printStackTrace();                            //打印堆栈信息
35  }
36  while(true){
37    ball.mAngleX += 2 * TOUCH_SCALE_FACTOR;         //设置外围小 8 面体旋转角度
38    requestRender();                                //重绘画面
39    try{
```

```
40          Thread.sleep(10);                                    //休息 10ms 再重绘
41        }
42        catch(Exception e) {                                   //捕获异常
43           e.printStackTrace();                                //打印堆栈信息
44     }   }   }  }.start();                                    //开启线程
45    }
46    public void onDrawFrame(GL10 gl) {                         //绘制方法
47       gl.glShadeModel(GL10.GL_SMOOTH);
48       gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
                                                                //清除颜色缓存
49       gl.glMatrixMode(GL10.GL_MODELVIEW);                    //设置当前矩阵为模式矩阵
50       glLoadIdentity();                                      //设置当前矩阵为单位矩阵
51       glTranslatef(0, 0f, -2.0f);     ball.drawSelf(gl);
52    }
53    public void onSurfaceChanged(GL10 gl, int width, int height) {
54      gl.glViewport(0, 0, width, height);                     //设置视窗大小及位置
55      gl.glMatrixMode(GL10.GL_PROJECTION);                    //设置当前矩阵为投影矩阵
56      gl.glLoadIdentity();                                    //设置当前矩阵为单位矩阵
57      float ratio = (float) width / height;                   //计算透视投影的比例
58      gl.glFrustumf(-ratio, ratio, -1, 1, 1, 10);   //调用此方法计算产生透视投影矩阵
59    }
60    public void onSurfaceCreated(GL10 gl, EGLConfig config) {
61      gl.glDisable(GL10.GL_DITHER);                           //关闭抗抖动
62      gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT,
63      GL10.GL_FASTEST);                                       //设置特定 Hint 项目的模式
64      gl.glClearColor(0,0,0,0);                               //设置屏幕背景色为黑色 RGBA
65      gl.glShadeModel(GL10.GL_SMOOTH);                        //设置着色模型为平滑着色
66      gl.glEnable(GL10.GL_DEPTH_TEST);                        //启用深度测试
67    } }   }
```

其中：

- 第 4～7 行为声明成员变量及创建常量数据。其中 mPreviousX、mPreviousY 用于记录上次手机屏幕触控点的坐标位置。
- 第 8～12 行为该界面类的构造器部分，在其中创建了渲染器对象，并设置了渲染器属性。
- 第 13～24 行为屏幕触控事件回调方法，其中首先获取当前触控屏幕的坐标，然后根据屏幕滑动的距离计算场景中物体旋转的角度。
- 第 25～67 行为渲染器内部类的构造方法。其中通过重写 onSurfaceCreated()方法实现了 3D 场景的创建，通过重写 onDrawFrame()方法实现了 3D 场景的绘制，通过 onSurfaceChanged()方法实现了界面变化时场景的设置。第 28～44 行为开启一个线程，用于控制球的旋转。

然后向读者介绍本项目开发中物体构造类的开发过程。

代码位置：Sample11_3/src/com.bn.chap11.qiu/ Spheroid.java

```
1   package com.bn.chap11.qiu;                                  //包的声明
2   import java.nio.ByteBuffer;                                 //引入相关类
3   public class Spheroid {
4   private IntBuffer   mVertexBuffer;                          //顶点坐标数据缓冲
5   private IntBuffer   mColorBuffer;                           //顶点着色数据缓冲
6   private ByteBuffer  mIndexBuffer;                           //顶点构建索引数据缓冲
7   public float mAngleX; public float mAngleY; public float mAngleZ;//旋转角度
8   int vCount=0;    int iCount=0;
9   public Spheroid(int scale){
10      final double a=2;final double b=2;final double c=2;final int UNIT_SIZE=10000;
```

```
11      ArrayList<Integer>alVertix=newArrayList<Integer>(); //存放顶点坐标的ArrayList
12      final int angleSpan=18;                              //将球进行单位切分的角度
13      for(int vAngle=-90;vAngle<=90;vAngle=vAngle+angleSpan){
                                                             //垂直方向 angleSpan
14      for(int hAngle=0;hAngle<360;hAngle=hAngle+angleSpan)
                                                             //水平方向 angleSpan
15      {                                  //纵向横向各到一个角度后计算对应的此点在球面上的坐标
16      int x=(int)(a*scale*UNIT_SIZE*Math.cos(Math.toRadians(vAngle))
17      *Math.cos(Math.toRadians(hAngle)));
18      int y=(int)(b*scale*UNIT_SIZE*Math.cos(Math.toRadians(vAngle))
19      *Math.sin(Math.toRadians(hAngle)));
20      int z=(int)(c*scale*UNIT_SIZE*Math.sin(Math.toRadians(vAngle)));
21      alVertix.add(x);alVertix.add(y);alVertix.add(z);//将顶点坐标的 ArrayList
22  }   }
23      vCount=alVertix.size()/3;                            //定点数量
24      int vertices[]=new int[vCount*3];                   //坐标值转存到一个 int 数组中
25  for(int i=0;i<alVertix.size();i++){
26      vertices[i]=alVertix.get(i);
27  }
28  ByteBuffer vbb = ByteBuffer.allocateDirect(vertices.length*4);
                                                             //创建顶点坐标数据缓冲
29  vbb.order(ByteOrder.nativeOrder());                      //设置字节顺序
30  mVertexBuffer = vbb.asIntBuffer();                       //转换为 int 型缓冲
31  mVertexBuffer.put(vertices);                             //向缓冲区中放入顶点坐标数据
32  mVertexBuffer.position(0);                               //设置缓冲区起始位置
33  final int one = 65535;
34  int colors[]=new int[vCount*4];                          //顶点颜色值数组
35  for(int i=0;i<vCount;i++){                               //随机生成每个顶点的颜色
36      colors[i*4]=(int) (one*Math.random());  colors[i*4+1]=(int) (one*Math.
random());//色彩通道
37      colors[i*4+2]=(int) (one*Math.random());colors[i*4+3]=0;//alpha 色彩通道
38  }
39  ByteBuffer cbb = ByteBuffer.allocateDirect(colors.length*4);
                                                             //创建顶点着色数据缓冲
40  cbb.order(ByteOrder.nativeOrder());                      //设置字节顺序
41  mColorBuffer = cbb.asIntBuffer();                        //转换为 int 型缓冲
42  mColorBuffer.put(colors);                                //向缓冲区中放入顶点着色数据
43  mColorBuffer.position(0);                                //设置缓冲区起始位置
44  ArrayList<Integer> alIndex=new ArrayList<Integer>();
45  int row=(180/angleSpan)+1;                               //球面切分的行数
46  int col=360/angleSpan;                                   //球面切分的列数
47  for(int i=0;i<row;i++) {                                 //对每一行循环
48  if(i>0&&i<row-1){                                        //中间行
49  for(int j=-1;j<col;j++){                                 //构成三角形
50      int k=i*col+j;alIndex.add(k+col);alIndex.add(k+1);alIndex.add(k);     }
51      for(int j=0;j<col+1;j++){                            //构成三角形
52          int k=i*col+j;alIndex.add(k-col);alIndex.add(k-1);alIndex.add(k);
53  }   }   }
54  iCount=alIndex.size(); byte indices[]=new byte[alIndex.size()];
55  for(int i=0;i<alIndex.size();i++){
56      indices[i]=alIndex.get(i).byteValue();
57  }
58  mIndexBuffer = ByteBuffer.allocateDirect(indices.length);
                                                             //创建三角形构造索引数据缓冲
59  mIndexBuffer.put(indices);                              //向缓冲区中放入三角形构造索引数据
60  mIndexBuffer.position(0);                                //设置缓冲区起始位置
61  }
62  public void drawSelf(GL10 gl){
```

```
63        gl.glRotatef(mAngleZ, 0, 0, 1);                    //沿 Z 轴旋转
64        gl.glRotatef(mAngleY, 0, 1, 0);                    //沿 Y 轴旋转
65        gl.glRotatef(mAngleX, 1, 0, 0);                    //沿 X 轴旋转
66        gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
67        gl.glEnableClientState(GL10.GL_COLOR_ARRAY);
68        gl.glVertexPointer(3,GL10.GL_FIXED,0,mVertexBuffer);//为画笔指定顶点坐标数据
69        gl.glColorPointer(4, GL10.GL_FIXED, 0, mColorBuffer );
                                                             //为画笔指定顶点着色数据
70        gl.glDrawElements(GL10.GL_TRIANGLES, iCount,
71        GL10.GL_UNSIGNED_BYTE, mIndexBuffer);              //绘制图形
72    }   }
```

其中：

- 第 4～8 行为声明成员变量，包括顶点数据缓存对象、顶点着色数据缓存对象和顶点索引数据缓存对象和顶点数量记录对象声明，顶点索引数量记录对象声明，以及场景选择角度对象声明。
- 第 9～60 行为该类构造的开发，本场景是由顶点法实现的。其中，第 10～30 行为获取顶点数组并将其存入顶点数据缓存；第 31～41 行为获取顶点着色数组并将其存入顶点着色数据缓存。
- 第 42～59 行为获取顶点索引数据并将其存放到顶点索引数据缓存。
- 第 61～72 行为绘制场景的方法。其中第 67 行为场景指定顶点坐标数据；第 68 行为场景指定顶点着色数据；第 69 行为场景绘制。

实例 4　丰富多彩的光照世界

在本小节中，将向读者介绍 3D 世界中必不可少的光照效果应用。

【实例描述】

本实例中，应用了两个定位光源来向读者显示光照效果，一个光源为红色光，一个光源为绿色光，两个光源都在绕球体旋转。

本实例运行效果如图 11-5 所示。

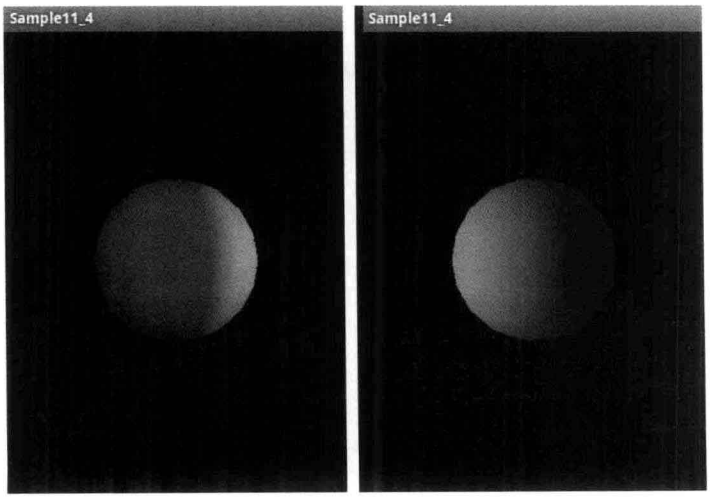


图 11-5　光照球效果图

提示：在图 11-5 中从左至右依次表示的是红色光源的照射物体的效果图，以及绿色光源照射物体的效果图。

【实现过程】

默认情况下，在 OpenGL ES 系统中灯光是关闭的；要想使用灯光，必须首先允许灯光并打开相应编号的灯。如下：gl.glEnable(GL10.GL_LIGHTING);//允许光,gl.glEnable(GL10.GL_LIGHT0);//打开 0 号灯。

在 OpenGL 中最多允许 8 盏灯，id 从“GL_LIGHT0”到“GL_LIGHT7”，OpenGL ES 中每盏灯的光分为 3 种组成元素：

- 环境光——AMBIENT；
- 散射光——DIFFUSE；
- 镜面反射光——SPECULAR。

【代码解析】

由于在本实例中立方体的创建过程中，Activity 类与三角形的绘制方法相似，因此在这里不再赘述。下面向读者介绍本项目开发中界面构造类的开发过程。

代码位置：Sample11_4/src/com.bn.chap11.gz/ MySurfaceView.java

```
1   package com.bn.chap11.gz;                    //包名
2   import android.opengl.GLSurfaceView;    //包引用，该处有所省略，读者可自行查看源代码
3   class MySurfaceView extends GLSurfaceView {
4     private final float TOUCH_SCALE_FACTOR = 180.0f/320;  //角度缩放比例
5     private SceneRenderer mRenderer;                      //场景渲染器
6     private float mPreviousY;                             //上次的触控位置 Y 坐标
7     private float mPreviousX;                             //上次的触控位置 X 坐标
8     private int lightAngleGreen=0;private int lightAngleRed=90;
9     public MySurfaceView(Context context) {
10    super(context);mRenderer = new SceneRenderer();       //创建场景渲染器
11    setRenderer(mRenderer);                               //设置渲染器
12  setRenderMode(GLSurfaceView.RENDERMODE_CONTINUOUSLY);//设置渲染模式为主动渲染
13  }
14  public boolean onTouchEvent(MotionEvent e) {          //触摸事件回调方法
15  ... ...//该处在上述软件实例中已详细介绍，在此不再赘述，读者可自行查看源代码
16  }
17  private class SceneRenderer implements GLSurfaceView.Renderer {
18    Ball ball=new Ball(4);
19    public SceneRenderer(){
20      new Thread() {                                  //开启一个线程自动旋转球体
21        public void run() {
22          try {
23              Thread.sleep(100);                      //休息 100ms 再重绘
24            }
25            catch(Exception e){
26              e.printStackTrace();                    //异常处理
27            }
28    while(true) {
29      lightAngleGreen+=5;lightAngleRed+=5; requestRender();   //重绘画面
30      try {
31        Thread.sleep(50);                             //休息 50ms 再重绘
32      }
```

```
33          catch(Exception e) {                                    //捕获异常
34          e.printStackTrace();                                    //打印信息
35    } } } }.start();                                              //开启线程
36    }
37    public void onDrawFrame(GL10 gl) {                            //绘制方法
38      gl.glShadeModel(GL10.GL_SMOOTH);
39      float lxGreen=(float)(7*Math.cos(Math.toRadians(lightAngleGreen)));
                                                                    //设定绿色光源的位置
40      float lzGreen=(float)(7*Math.sin(Math.toRadians(lightAngleGreen)));
41      float[] positionParamsGreen={lxGreen,0,lzGreen,1};    //最后的 1 表示定位光
42      gl.glLightfv(GL10.GL_LIGHT0, GL10.GL_POSITION, positionParamsGreen,0);
43      float lyRed=(float)(7*Math.cos(Math.toRadians(lightAngleRed)));
                                                                    //设定红色光源的位置
44      float lzRed=(float)(7*Math.sin(Math.toRadians(lightAngleRed)));
45      float[] positionParamsRed={0,lyRed,lzRed,1};          //最后的 1 表示定位光
46      gl.glLightfv(GL10.GL_LIGHT1, GL10.GL_POSITION, positionParamsRed,0);
47      glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT); //清除颜色缓存
48      gl.glMatrixMode(GL10.GL_MODELVIEW);                   //设置当前矩阵为模式矩阵
49      gl.glLoadIdentity();                                  //设置当前矩阵为单位矩阵
50      gl.glTranslatef(0, 0f, -1.8f); ball.drawSelf(gl);
51    }
52    public void onSurfaceChanged(GL10 gl, int width, int height) {      //重写的方法
53    ... ...//该处在上述软件实例中已详细介绍，在此不再赘述，读者可自行查看随书光盘中的源代码
54    }
55    public void onSurfaceCreated(GL10 gl, EGLConfig config) {
56      gl.glDisable(GL10.GL_DITHER);                         //关闭抗抖动
57      gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT,
58      GL10.GL_FASTEST);          //设置特定 Hint 项目的模式，这里设置为使用快速模式
59      gl.glClearColor(0,0,0,0);                             //设置屏幕背景色为黑色 RGBA
60      gl.glShadeModel(GL10.GL_SMOOTH);                      //设置着色模型为平滑着色
61      gl.glEnable(GL10.GL_DEPTH_TEST);                      //启用深度测试
62      gl.glEnable(GL10.GL_LIGHTING);                        //允许光照
63      initGreenLight(gl); initRedLight(gl);                 //初始化灯
64      initMaterial(gl);                                     //初始化材质
65    }   }
66    private void initGreenLight(GL10 gl){... ...}          /初始化绿色光源
67    private void initRedLight(GL10 gl){... ...}            //初始化红色光源
68    private void initMaterial(GL10 gl){... ...}
                                    //材质为白色时什么颜色的光照在上面就将体现出什么颜色
69    }
```

其中：

- 第 4～8 行为成员变量的声明部分。包括渲染器对象和用于存放屏幕上个触控点的坐标值，以及光源的光照角度值的初始化。
- 第 9～13 行为该界面实现类的构造器部分，在其中主要创建了用于渲染场景的渲染器对象，并对其进行了适当的属性设置。
- 第 14～16 行为屏幕触控回调方法。第 17～65 行为渲染器内部类的构造方法。其中重写 onSurfaceCreated()方法来创建场景，重写 onDrawFrame()方法来绘制场景，重写 onSurfaceChanged()方法用于当界面发生改变时设置当前场景。
- 第 66～68 行为初始化光源属性，以及初始化物体材质。

然后向读者介绍光照设置方法的具体实现过程。

代码位置：Sample11_4/src/com.bn.chap11.gz/ MySurfaceView.java

```
1   private void initGreenLight(GL10 gl){
2     gl.glEnable(GL10.GL_LIGHT0);                                  //打开 0 号灯
```

```
    float[] ambientParams={0.1f,0.1f,0.1f,1.0f};                    //环境光设置
    gl.glLightfv(GL10.GL_LIGHT0, GL10.GL_AMBIENT, ambientParams,0);
    float[] diffuseParams={0f,1f,0f,1.0f};                          //散射光设置
    .glLightfv(GL10.GL_LIGHT0, GL10.GL_DIFFUSE, diffuseParams,0);
     float[] specularParams={0.0f,1.0f,0.0f,1.0f};                  //反射光设置
    gl.glLightfv(GL10.GL_LIGHT0, GL10.GL_SPECULAR, specularParams,0);
  }
  private void initRedLight(GL10 gl){
    gl.glEnable(GL10.GL_LIGHT1);                                    //打开 1 号灯
    float[] ambientParams={0.1f,0.1f,0.1f,1.0f};                    //环境光设置
    gl.glLightfv(GL10.GL_LIGHT1, GL10.GL_AMBIENT, ambientParams,0);
    float[] diffuseParams={1f,0f,0f,1.0f};                          //散射光设置
    gl.glLightfv(GL10.GL_LIGHT1, GL10.GL_DIFFUSE, diffuseParams,0);
    float[] specularParams={1f,0f,0f,1.0f};                         //反射光设置
    gl.glLightfv(GL10.GL_LIGHT1, GL10.GL_SPECULAR, specularParams,0);
  }
  private void initMaterial(GL10 gl){
                                  //材质为白色时什么颜色的光照在上面就将体现出什么颜色
    float ambientMaterial[] = {0.4f, 0.4f, 0.4f, 1.0f};   //环境光为白色材质
    gl.glMaterialfv(GL10.GL_FRONT_AND_BACK, GL10.GL_AMBIENT, ambientMaterial,0);
    float diffuseMaterial[] = {0.8f, 0.8f, 0.8f, 1.0f};   //散射光为白色材质
    gl.glMaterialfv(GL10.GL_FRONT_AND_BACK, GL10.GL_DIFFUSE, diffuseMaterial,0);
    float specularMaterial[] = {1.0f, 1.0f, 1.0f, 1.0f};  //高光材质为白色
    gl.glMaterialfv(GL10.GL_FRONT_AND_BACK, GL10.GL_SPECULAR, specularMaterial,0);
    float shininessMaterial[] = {1.5f};       //高光反射区域,数越大高亮区域越小越暗
    gl.glMaterialfv(GL10.GL_FRONT_AND_BACK, GL10.GL_SHININESS,
shininessMaterial,0);
  }
```

提示：第 1～18 行为设置绿色和红色光的属性，包括打开的灯、环境光设置、散射光设置、反射光设置。第 19～28 行为设置球体的材质，包括环境光材质、散射光材质、高光材质和反射光材质。

实例 5　制作简易小木箱

在本小节中，将向读者介绍 3D 世界中必不可少的纹理贴图技术。

【实例描述】

本实例构造了一个 3D 木箱物体，该物体在游戏场景中频繁应用，因此借助对该物体的构造，来向读者介绍纹理贴图技术。

本实例运行效果如图 11-6、图 11-7 所示。

提示：运行实例时，首先显示的效果如图 11-6 所示，并且用户可以用手指滑动手机屏幕的方式来改变物体的呈现角度。木箱的立体效果图如图 11-7 所示。

【实现过程】

在本节中应用了纹理映射技术。纹理映射允许将指定的纹理素材映射到三维物体指定的多边形面上。要注意的是 OpenGL ES 中仅支持三角形，因此指定纹理时也要注意这一点，纹理坐标范围为 0.0～1.0，S 轴表示纹理的横向坐标，T 轴表示纹理的纵向坐标。

OpenGL ES 中进行纹理映射时对纹理图片的尺寸是有规定的，其要求纹理图片的宽度与高度必须是 2^n(2 的 n 次方)，即 32x32/256x512 等。

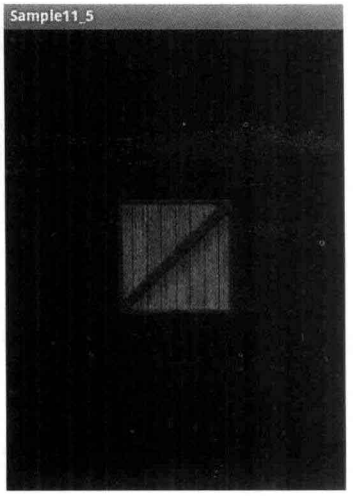


图 11-6　起始效果图

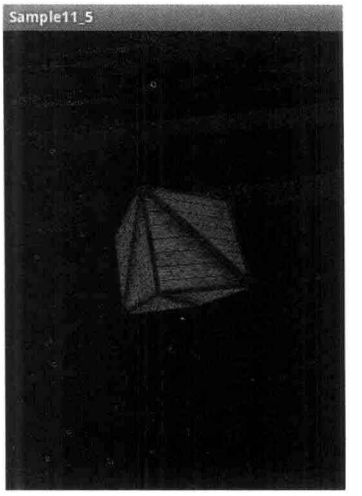


图 11-7　立体效果图

【代码解析】

由于在本实例中立方体的创建过程中，Activity 类与三角形的绘制方法相似，因此在这里不再赘述。

下面向读者介绍的是本项目开发中界面构造类的开发过程。

代码位置：Sample11_5/src/com.bn.chap11.crate/ MySurfaceView.java

```
1   package com.bn.chap11.crate;                    //包名
2   import java.io.IOException;                     //包引用，该处有所省略，读者可自行查看源代码
3   class MySurfaceView extends GLSurfaceView {
4     private final float TOUCH_SCALE_FACTOR = 180.0f/320;  //角度缩放比例
5     private SceneRenderer mRenderer;                      //场景渲染器
6     private float mPreviousY;                             //上次的触控位置 Y 坐标
7   private float mPreviousX;                               //上次的触控位置 X 坐标
8   int crateTextureId;                                     //木箱纹理 Id
9   public MySurfaceView(Context context) {
10    super(context);
11    mRenderer = new SceneRenderer();                      //创建场景渲染器
12    setRenderer(mRenderer);                               //设置渲染器
13    setRenderMode(GLSurfaceView.RENDERMODE_CONTINUOUSLY);//设置渲染模式为主动渲染
14  }
15  public boolean onTouchEvent(MotionEvent e) {          //触摸事件回调方法
16  ... ...//该处详细代码已在上述章节做了详细介绍，在此就不再赘述。
17  }
18  private class SceneRenderer implements GLSurfaceView.Renderer  {
19     Crate tp=new Crate();
20    public SceneRenderer(){
21    }
22    public void onDrawFrame(GL10 gl) {
23      gl.glDisable(GL10.GL_CULL_FACE);                    //设置为关闭背面剪裁
24      gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
25      gl.glMatrixMode(GL10.GL_MODELVIEW);                 //设置当前矩阵为模式矩阵
26      gl.glLoadIdentity();                                //设置当前矩阵为单位矩阵
```

```
27      gl.glTranslatef(0, 0f, -5f);                          //平移操作
28      tp.drawSelf(gl,crateTextureId);                       //绘制场景
29    }
30    public void onSurfaceChanged(GL10 gl, int width, int height) {
                                                              //设置视窗大小及位置
31      gl.glViewport(0, 0, width, height);
32      gl.glMatrixMode(GL10.GL_PROJECTION);                  //设置当前矩阵为投影矩阵
33      gl.glLoadIdentity();                                  //设置当前矩阵为单位矩阵
34      float ratio = (float) width / height;                 //计算透视投影的比例
35      gl.glFrustumf(-ratio, ratio, -1, 1, 1, 10);   //调用此方法计算产生透视投影矩阵
36    }
37    public void onSurfaceCreated(GL10 gl, EGLConfig config){
38      crateTextureId=initTexture(gl,R.drawable.crate);
39      gl.glDisable(GL10.GL_DITHER);    //关闭抗抖动
40      gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT,
41      GL10.GL_FASTEST);            //设置特定 Hint 项目的模式，这里设置为使用快速模式
42      gl.glClearColor(0,0,0,0);                             //设置屏幕背景色为黑色 RGBA
43      gl.glEnable(GL10.GL_DEPTH_TEST);                      //启用深度测试
44    }    }
45    public int initTexture(GL10 gl,int drawableId)//textureId{    //初始化纹理
46      int[] textures = new int[1];                                //生成纹理 Id
47      gl.glGenTextures(1, textures, 0);                     //自动获取一个纹理名称
48      int currTextureId=textures[0];                        //将当前纹理存放到数组中
49      gl.glBindTexture(GL10.GL_TEXTURE_2D, currTextureId);        //绑定当前纹理
50      gl.glTexParameterf(GL10.GL_TEXTURE_2D,GL10.GL_TEXTURE_MIN_FILTER,GL10.
GL_NEAREST);
51      gl.glTexParameterf(GL10.GL_TEXTURE_2D,
52      GL10.GL_TEXTURE_MAG_FILTER,GL10.GL_LINEAR);
53      gl.glTexParameterf(GL10.GL_TEXTURE_2D,
54      GL10.GL_TEXTURE_WRAP_S,GL10.GL_CLAMP_TO_EDGE);
55      gl.glTexParameterf(GL10.GL_TEXTURE_2D,
55      GL10.GL_TEXTURE_WRAP_T,GL10.GL_CLAMP_TO_EDGE);
56     InputStream is = this.getResources().openRawResource(drawableId);
                                                              //图片资源流
57     Bitmap bitmapTmp;                                      //声明位图变量
58    try{
59      bitmapTmp = BitmapFactory.decodeStream(is);           //将流数据转换成位图文件
60    }
61    finally {
62     try {
63       is.close();                                          //关闭流
64      }
65      catch(IOException e) {
66        e.printStackTrace();                                //异常处理
67       }      }
68     GLUtils.texImage2D(GL10.GL_TEXTURE_2D, 0, bitmapTmp, 0);//指定当前纹理位图
69     bitmapTmp.recycle();                                   /回收位图资源
70     return currTextureId;
71    }       }
```

其中：

- 第 4～8 行为成员变量的声明部分，包括渲染器对象和用于存放屏幕上个触控点的坐标值。
- 第 9～14 行为该界面实现类的构造器部分，在其中主要创建了用于渲染场景的渲染器对象，并对其进行了适当的属性设置。
- 第 15～17 行为屏幕触控回调方法。第 18～44 行为渲染器内部类的构造方法。其中重写 onSurfaceCreated()方法来创建场景，重写 onDrawFrame()方法来绘制场景，重写

onSurfaceChanged()方法，用于当界面发生改变时设置当前场景。

- 第 45～71 行为获取应用位图纹理 Id，其中，第 50～53 行为设置当前纹理位图资源的显示属性。

下面向读者介绍本项目开发中物体构造类的开发过程。

代码位置：Sample11_5/src/com.bn.chap11.crate/ Crate.java

```
1   package com.bn.chap11.crate;                    //包名
2   import java.nio.ByteBuffer;                     //包引用，该处有所省略，读者可自行查看源代码
3   import javax.microedition.khronos.opengles.GL10;
4   public class Crate {
5     private FloatBuffer  mVertexBuffer; //顶点坐标数据缓冲
6     private FloatBuffer  mTextureBuffer;//顶点纹理数据缓冲
7     int vCount=0;
8     float yAngle=0;                               //绕 y 轴旋转的角度
9     float zAngle=0;                               //绕 z 轴旋转的角度
10    public Crate(){
11    vCount=36;                                    //顶点数量
12    float size=1.0f;                              //单位常量
13    final int UNIT_SIZE=1;
14  float vertices[]=new float[]{                   //用于给出顶点坐标数据
15   ...//立方体顶点数据已经在上述实例中给出，在此不再赘述
16  };
17  ByteBuffer vbb = ByteBuffer.allocateDirect(vertices.length*4);
                                                    //创建顶点坐标数据缓冲
18  vbb.order(ByteOrder.nativeOrder());             //设置字节顺序
19  mVertexBuffer = vbb.asFloatBuffer();            //转换为 int 型缓冲
20  mVertexBuffer.put(vertices);                    //向缓冲区中放入顶点坐标数据
21  mVertexBuffer.position(0);                      //设置缓冲区起始位置
22  float textures[]=new float[] {                  //顶点颜色值数组，每个顶点 4 个色彩值 RGBA
23      0,0, 0,1, 1,0,                              //第 1 个三角形顶点纹理坐标数据
24      1,0, 0,1, 1,1,                              //第 2 个三角形顶点纹理坐标数据
25      0,0, 0,1, 1,0,                              //第 3 个三角形顶点纹理坐标数据
26      1,0, 0,1, 1,1,                              //第 4 个三角形顶点纹理坐标数据
27      0,0, 0,1, 1,0,                              //第 5 个三角形顶点纹理坐标数据
28      1,0, 0,1, 1,1,                              //第 6 个三角形顶点纹理坐标数据
29      0,0, 0,1, 1,0,                              //第 7 个三角形顶点纹理坐标数据
30      1,0, 0,1, 1,1,                              //第 8 个三角形顶点纹理坐标数据
31      0,0, 0,1, 1,0,                              //第 9 个三角形顶点纹理坐标数据
32      1,0, 0,1, 1,1,                              //第 10 个三角形顶点纹理坐标数据
33      0,0, 0,1, 1,0,                              //第 11 个三角形顶点纹理坐标数据
34      1,0, 0,1, 1,1                               //第 12 个三角形顶点纹理坐标数据
35      };
36      ByteBuffer cbb = ByteBuffer.allocateDirect(textures.length*4);
                                                    //创建顶点纹理数据缓冲
37      cbb.order(ByteOrder.nativeOrder());         //设置字节顺序
38      mTextureBuffer = cbb.asFloatBuffer();       //转换为 int 型缓冲
39      mTextureBuffer.put(textures);               //向缓冲区中放入顶点着色数据
40      mTextureBuffer.position(0);                 //设置缓冲区起始位置
41    }
42    public void drawSelf(GL10 gl,int texId){
43      gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);       //启用顶点坐标数组
44      gl.glRotatef(yAngle, 0, 1, 0); gl.glRotatef(zAngle, 0, 0, 1);
45      gl.glVertexPointer(3,GL10.GL_FLOAT,0, mVertexBuffer);
                                                    //为画笔指定顶点坐标数据
46      gl.glEnable(GL10.GL_TEXTURE_2D);            //开启纹理
47      gl.glEnableClientState(GL10.GL_TEXTURE_COORD_ARRAY);
```

```
                                                    //允许使用纹理 ST 坐标缓冲
48          gl.glTexCoordPointer(2, GL10.GL_FLOAT, 0, mTextureBuffer);
                                                    //为画笔指定纹理 ST 坐标缓冲
49          gl.glBindTexture(GL10.GL_TEXTURE_2D, texId);        //绑定当前纹理
50          gl.glDrawArrays (GL10.GL_TRIANGLES, 0, vCount );    //绘制图形
51          gl.glDisableClientState(GL10.GL_TEXTURE_COORD_ARRAY)    //关闭纹理
52          gl.glDisable(GL10.GL_TEXTURE_2D);
53          gl.glDisableClientState(GL10.GL_VERTEX_ARRAY);
54      }    }
```

其中：

- 第 5～9 行为成员变量的声明，包括顶点坐标数据缓存对像、顶点纹理坐标数据缓存对象和位图纹理 id，以及旋转角度的初始化。
- 第 10～41 行为该 3D 物体构造类的构造器的开发部分。其中，第 11～21 行为顶点坐标数据缓存设置，第 22～40 行为顶点纹理坐标数据缓存设置。
- 第 42～52 行为物体的绘制方法，在绘制前，首先指定顶点坐标数据缓冲和纹理坐标数据缓冲。在绘制完成后，关闭纹理缓冲和顶点缓冲。

实例 6　朦胧世界的雾景特效

在本小节中，将向读者介绍 3D 世界中的雾特效应用，可以使场景更加逼真。

【实例描述】

本实例的开发是在上一实例的基础上实现的，为木箱添加而来雾特效。使木箱看起很朦胧，近处的较为清楚，远处的不清楚。使读者充分体验雾特效的效果。

本实例运行效果如图 11-8 所示。

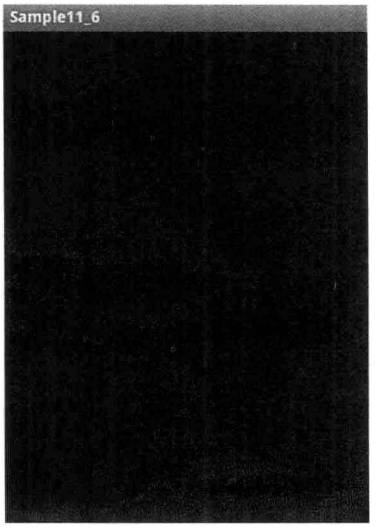


图 11-8　木箱的雾效果图

提示：在图 11-8 中为木箱添加了雾效果，它是在实例 5 的基础上实现的。

【实现过程】

本小节用到了雾特效技术。雾是一个通用术语，描述了类似大气效果的各种形态，其可以拥有模拟薄雾烟雾和污染。雾特效被启用后，离视点较远的物体将融入到雾的颜色中。可以控制雾的浓度，其决定了物体的淡化速度和雾的颜色。

大多数仿真程序中，使用雾可以提高性能。因为启动雾效果后无须绘制由于雾的影响而看不见的物体。

【代码解析】

由于在本实例中立方体的创建过程中，Activity 类与三角形的绘制方法相似，因此在这里不再赘述。

下面向读者介绍的是本项目开发中界面构造类的开发过程。

代码位置：Sample11_6/src/com.bn.chap11.wu/ MySurfaceView.java

```
1   package com.bn.chap11.wu;
2   import java.io.IOException;
3   class MySurfaceView extends GLSurfaceView {
4     private final float TOUCH_SCALE_FACTOR = 180.0f/320;       //角度缩放比例
5     private SceneRenderer mRenderer;                           //场景渲染器
6     private float mPreviousY;                                  //上次的触控位置 Y 坐标
7     private float mPreviousX;                                  //上次的触控位置 X 坐标
8     private int lightAngle=90;
9     int crateTextureId;                                        //木箱纹理 Id
10    public MySurfaceView(Context context) {
11      super(context);
12      mRenderer = new SceneRenderer();                         //创建场景渲染器
13      setRenderer(mRenderer);                                  //设置渲染器
14      setRenderMode(GLSurfaceView.RENDERMODE_CONTINUOUSLY); //设置渲染模式为主动渲染
15  }
16  public boolean onTouchEvent(MotionEvent e) {                 //触摸事件回调方法
17  float y = e.getY(); float x = e.getX();
18  switch (e.getAction()) {
19    case MotionEvent.ACTION_MOVE:
20      float dy = y - mPreviousY;loat dx = x - mPreviousX;  //计算触控笔位移
21      mRenderer.tp.yAngle += dx * TOUCH_SCALE_FACTOR;      //设置沿 x 轴旋转角度
22      mRenderer.tp.zAngle+= dy * TOUCH_SCALE_FACTOR;       //设置沿 Y 轴旋转角度
23      requestRender();                                     //重绘画面
24   }
25   mPreviousY = y; mPreviousX = x;                         //记录触控笔位置
26   return true;
27  }
28  private class SceneRenderer implements GLSurfaceView.Renderer {
29    Crate tp=new Crate();
30    public SceneRenderer(){
31    }
32    public void onDrawFrame(GL10 gl) {
33      float lx=(float)(10*Math.cos(Math.toRadians(lightAngle)));
                                                             //设定绿色光源的位置
34      float lz=(float)(10*Math.sin(Math.toRadians(lightAngle)));
35      float[] positionParamsGreen={lx,0,lz,1};          //最后的 1 表示定位光
36      gl.glLightfv(GL10.GL_LIGHT0, GL10.GL_POSITION, positionParamsGreen,0);
37      gl.glDisable(GL10.GL_CULL_FACE);                  //设置为关闭背面剪裁
38      gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
39      gl.glMatrixMode(GL10.GL_MODELVIEW);               //设置当前矩阵为模式矩阵
40      gl.glLoadIdentity();                              //设置当前矩阵为单位矩阵
```

```
41        gl.glTranslatef(0, 0f, -5f);
42       tp.drawSelf(gl,crateTextureId);
43   }
44   public void onSurfaceChanged(GL10 gl, int width, int height) {
45      gl.glViewport(0, 0, width, height);                    //设置视窗大小及位置
46      gl.glMatrixMode(GL10.GL_PROJECTION);                   //设置当前矩阵为投影矩阵
47      gl.glLoadIdentity();                                   //设置当前矩阵为单位矩阵
48      float ratio = (float) width / height;                  //计算透视投影的比例
49      gl.glFrustumf(-ratio, ratio, -1, 1, 1, 100);
                                          //调用此方法计算产生透视投影矩阵        }
50   public void onSurfaceCreated(GL10 gl, EGLConfig config) {
51      gl.glDisable(GL10.GL_DITHER);                          //关闭抗抖动
52      gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT,GL10.GL_FASTEST);
53      gl.glClearColor(0,0,0,0);                              //设置屏幕背景色为黑色 RGBA
54      gl.glEnable(GL10.GL_DEPTH_TEST);//启用深度测试
55      rateTextureId=initTexture(gl,R.drawable.crate);
56      gl.glEnable(GL10.GL_FOG);                              //允许雾
57      initFog(gl);                                           //初始化雾
58   }     }
59   public int initTexture(GL10 gl,int drawableId)//textureId{
60   ... ...//初始化纹理
61   }
62   public void initFog(GL10 gl){                             //初始化雾
63      float[] fogColor={220/255,220/255,220/255,0};          //雾的颜色
64      gl.glFogfv(GL10.GL_FOG_COLOR, fogColor, 0)             ;//设置雾的颜色
65      gl.glFogx(GL10.GL_FOG_MODE, GL10.GL_EXP2);             //设置雾的模式
66      gl.glFogf(GL10.GL_FOG_DENSITY, 0.3f);                  //设置雾的浓度
67   gl.glFogf(GL10.GL_FOG_START, 0.5f);                       //设置雾的开始距离
68   gl.glFogf(GL10.GL_FOG_END, 100.0f);                       //设置雾的结束距离
69   }     }
```

其中：

- 第 10～15 行为 MySurfaceView 界面类的构造器部分，在其中创建了场景渲染器对象，并设置其属性。
- 第 28～58 行为渲染器内部类的构造方法。其中重写 onSurfaceCreated()方法来创建场景，重写 onDrawFrame()方法来绘制场景，重写 onSurfaceChanged()方法，用于当界面发生改变时设置当前场景。
- 第 59～61 行为获取应用位图纹理 id。该处省略，读者可自行查看随书光盘中的源代码。
- 第 62～69 行为初始化雾的方法。其中包括设置雾的颜色，雾的模式，雾的浓度，雾的起始距离和结束距离。

实例 7　透过玻璃看风景

在本小节中，将向读者介绍 3D 世界中的混合技术的应用，可以使场景更加逼真。

【实例描述】

本实例中，通过应用 3D 混合技术，实现了透过玻璃图片来欣赏浏览风景图片的效果，相似于在真实世界中透过玻璃看窗外风景的感受。本实例运行效果如图 11-9 所示。

提示：本实例效果如图 1-9 所示，它较为真实地体现了透过玻璃来观看外部世界的朦胧效果。

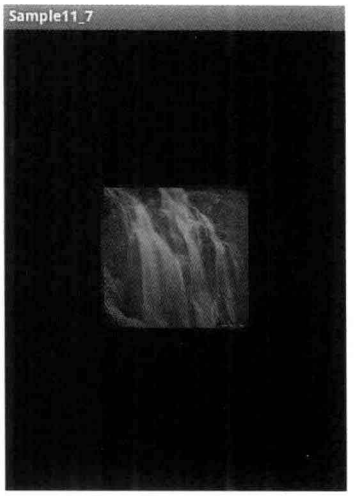


图 11-9　透过玻璃看风景效果图

【实现过程】

本小节用了混合技术。启用 Alpha 混合后，alpha 值用于将正在处理的片元 RGB 颜色值与存储在帧缓存中的像素 RGB 颜色值进行合并。混合功能被禁用时，使用新片元覆盖帧缓存中的颜色值，就产生了片元不透明的效果。

通过 Alpha 值在混合操作中可以控制新片元的颜色值与原有颜色值的合并权重。因此，通过 Alpha 混合，可以创建半透明效果的片元。Alpha 颜色混合是诸如透明度，数字合成等技术的核心。

本实例中应用的源因子和目标因子分别为 GL_SRC_ALPHA 和 GL_ONE_MINUS_SRC_ALPHA。

【代码解析】

由于在本实例中立方体的创建过程中，Activity 类与三角形的绘制方法相似，因此在这里不再赘述。

下面向读者介绍的是本项目开发中界面构造类的开发过程。

代码位置：Sample11_6/src/com.bn.chap11.hh/ MySurfaceView.java

```
1   package com.bn.chap11.hh;                    //包名
2   import java.io.IOException; //包引用，该处有所省略，读者可自行查看随书光盘中的源代码
3   class MySurfaceView extends GLSurfaceView {
4     private SceneRenderer mRenderer;                    //场景渲染器
5     public MySurfaceView(Context context) {
6       super(context);mRenderer = new SceneRenderer();//创建场景渲染器
7       setRenderer(mRenderer);                          //设置渲染器
8   setRenderMode(GLSurfaceView.RENDERMODE_CONTINUOUSLY);//设置渲染模式为主动渲染
9   }
10  private class SceneRenderer implements GLSurfaceView.Renderer {
11    int baseTextureId;                                //最底层矩形的不透明纹理的纹理 Id
12    int topTextureId;                                 //顶层透明纹理的纹理 Id
13    TextureRect t1;                                   //风景/纹理
14    TextureRect t2;                                   //玻璃纹理
15  public void onDrawFrame(GL10 gl) {
```

```
    gl.glShadeModel(GL10.GL_SMOOTH);                    //采用平滑着色
    gl.glClear(GL10.GL_COLOR_BUFFER_BIT
    GL10.GL_DEPTH_BUFFER_BIT);                          //清除颜色缓存于深度缓存
    gl.glMatrixMode(GL10.GL_MODELVIEW);                 //设置当前矩阵为模式矩阵
    gl.glLoadIdentity();                                //设置当前矩阵为单位矩阵
   gl.glPushMatrix();gl.glTranslatef(0, 0f, -2f); t1.drawSelf(gl);gl.glPopMatrix();
                                                        //绘制底层纹理矩形
    gl.glPushMatrix();gl.glTranslatef(0, 0f, -1.9f);t2.drawSelf(gl);gl.glPop
Matrix(); //绘制上层纹理矩形
  }
  public void onSurfaceChanged(GL10 gl, int width, int height) {
                                                        //设置视窗大小及位置
   gl.glViewport(0, 0, width, height);                  //设置当前矩阵为投影矩阵
   gl.glMatrixMode(GL10.GL_PROJECTION);                 //设置当前矩阵为单位矩阵
   gl.glLoadIdentity();                                 //计算透视投影的比例
   float ratio = (float) width / height;                //调用此方法计算产生透视投影矩阵
   gl.glFrustumf(-ratio, ratio, -1, 1, 1, 100);
  }
  public void onSurfaceCreated(GL10 gl, EGLConfig config) {
   gl.glDisable(GL10.GL_DITHER);                        //关闭抗抖动
   gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT,
   GL10.GL_FASTEST);              //设置特定 Hint 项目的模式，这里设置为使用快速模式
   gl.glClearColor(0,0,0,0);                            //设置屏幕背景色为黑色 RGBA
   gl.glEnable(GL10.GL_DEPTH_TEST);                     //启用深度测试
   gl.glEnable(GL10.GL_BLEND);                          //开启混合
   gl.glBlendFunc(GL10.GL_SRC_ALPHA, GL10.GL_ONE_MINUS_SRC_ALPHA);
   baseTextureId=initTexture(gl,R.drawable.fj1); //初始化纹理
   topTextureId=initTexture(gl,R.drawable.top);
   t1=new TextureRect(baseTextureId);                   //创建矩形
   t2=new TextureRect(topTextureId);
  }       }
  public int initTexture(GL10 gl,int drawableId)//textureId
  {                                                     //初始化纹理
    int[] textures = new int[1];                        //生成纹理 Id
    gl.glGenTextures(1, textures, 0);
    int currTextureId=textures[0];
    gl.glBindTexture(GL10.GL_TEXTURE_2D, currTextureId);
    gl.glTexParameterf(GL10.GL_TEXTURE_2D,GL10.GL_TEXTURE_MIN_FILTER,
GL10.GL_NEAREST);
    gl.glTexParameterf(GL10.GL_TEXTURE_2D,
    GL10.GL_TEXTURE_MAG_FILTER,GL10.GL_LINEAR);
    gl.glTexParameterf(GL10.GL_TEXTURE_2D,
    GL10.GL_TEXTURE_WRAP_S,GL10.GL_CLAMP_TO_EDGE);
    gl.glTexParameterf(GL10.GL_TEXTURE_2D,
    GL10.GL_TEXTURE_WRAP_T,GL10.GL_CLAMP_TO_EDGE);
    InputStream is = this.getResources().openRawResource(drawableId);
    Bitmap bitmapTmp;
    try{
        bitmapTmp = BitmapFactory.decodeStream(is);    //由数据流获取位图对象
     }
     finally {
         try {
            is.close();                                 //关闭流对象
         }
          catch(IOException e) {
              e.printStackTrace();                      //异常处理
    }     }
  GLUtils.texImage2D(GL10.GL_TEXTURE_2D, 0, bitmapTmp, 0);     //指定纹理
  bitmapTmp.recycle();                                  //回收位图资源
```

```
71   return currTextureId;
72   }    }
```

其中：

- 第 5～9 行为 MySurfaceView 界面类的构造器部分。在其中创建了场景渲染器对象，并设置其属性。
- 第 10～43 行为渲染器内部类的构造方法。其中重写 onSurfaceCreated()方法来创建场景，重写 onDrawFrame()方法来绘制场景，重写 onSurfaceChanged()方法用于当界面发生改变时设置当前场景。
- 第 44～72 行为获取应用位图纹理 id，其中，第 50～53 行为设置当前纹理位图资源的显示属性。

下面向读者介绍的是本项目开发中物体构造类的开发过程。

代码位置：Sample11_7/src/com.bn.chap11.hh/ TextureRect.java

```
1    package com.bn.chap11.hh;                              //包名
2    import java.nio.ByteBuffer;                            //包引用
3    public class TextureRect {
4      private IntBuffer   mVertexBuffer;                   //顶点坐标数据缓冲
5      private FloatBuffer   mTextureBuffer;                //顶点纹理数据缓冲
6      int vCount;int texId;                        //顶点数量记录成员变量，纹理 Id 成员变量
7    public TextureRect(int texId) {                        //构造器
8      this.texId=texId; vCount=6;final int UNIT_SIZE=40000;
9      int vertices[]=new int[]{                            //用于存放顶点坐标数据
10    -1*UNIT_SIZE,1*UNIT_SIZE,0,                           //第一个三角形的顶点坐标数据值
11    -1*UNIT_SIZE,-1*UNIT_SIZE,0,
12    1*UNIT_SIZE,1*UNIT_SIZE,0,
13    -1*UNIT_SIZE,-1*UNIT_SIZE,0,                          //第二个三角形的顶点坐标数据值
14    1*UNIT_SIZE,-1*UNIT_SIZE,0,
15    1*UNIT_SIZE,1*UNIT_SIZE,0
16    };
17    ByteBuffer vbb = ByteBuffer.allocateDirect(vertices.length*4);
                                                           //创建顶点坐标数据缓冲
18    vbb.order(ByteOrder.nativeOrder());                  //设置字节顺序
19    mVertexBuffer = vbb.asIntBuffer();                   //转换为 int 型缓冲
20    mVertexBuffer.put(vertices);                         //向缓冲区中放入顶点坐标数据
21    mVertexBuffer.position(0);                           //设置缓冲区起始位置
22    float textures[]=new float[] {
23        0,0,0,1,1,0,
24        0,1,1,1,1,0
25    };
26    ByteBuffer tbb = ByteBuffer.allocateDirect(textures.length*4);
                                                           //创建顶点纹理数据缓冲
27    tbb.order(ByteOrder.nativeOrder());                  //设置字节顺序
28    mTextureBuffer= tbb.asFloatBuffer();                 //转换为 Float 型缓冲
29    mTextureBuffer.put(textures);                        //向缓冲区中放入顶点着色数据
30    mTextureBuffer.position(0);                          //设置缓冲区起始位置
31    }
32    public void drawSelf(GL10 gl){
33        gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);        //启用顶点坐标数组
34        gl.glVertexPointer (3,GL10.GL_FIXED,0, mVertexBuffer );
                                                           //为画笔指定顶点坐标数据
35        gl.glEnable(GL10.GL_TEXTURE_2D);                 //开启纹理
36        gl.glEnableClientState(GL10.GL_TEXTURE_COORD_ARRAY);
                                                           //允许使用纹理 ST 坐标缓冲
37        gl.glTexCoordPointer(2, GL10.GL_FLOAT, 0, mTextureBuffer);
                                                           //为画笔指定纹理 ST 坐标缓冲
```

```
38          gl.glBindTexture(GL10.GL_TEXTURE_2D, texId);          //绑定当前纹理
39          gl.glDrawArrays(GL10.GL_TRIANGLES,0,vCount);          //绘制图形
40          gl.glDisableClientState(GL10.GL_TEXTURE_COORD_ARRAY);//关闭纹理
41          gl.glDisable(GL10.GL_TEXTURE_2D);
42      }    }
```

其中：

- 第 4～6 行为成员变量的声明，包括顶点坐标数据缓存对像、顶点纹理坐标数据缓存对象和位图纹理 Id。
- 第 7～31 行为该 3D 物体构造类的构造器的开发部分。其中，第 8～21 行为顶点坐标数据缓存设置；第 22～30 行为顶点纹理坐标数据缓存设置。
- 第 32～42 行为物体的绘制方法，在绘制前，首先要指定顶点坐标数据缓冲和纹理坐标数据缓冲。在绘制完成后，关闭纹理缓冲和顶点缓冲。

实例 8　3D 相册的制作

在本小节中，将向读者介绍一个综合性的实例，3D 相册制作。有助于读者将上述知识点融会贯通。

【实例描述】

本款软件打造了绚丽的 3D 效果，将相册置于立体空间中呈现，并且可以用手指横向滑动手机屏幕的方式来选择所有浏览的图片，在滑动过程中相册的旋转同时带有惯性因素，使场景效果更自然更人性化，并可以通过单击选择图片来显示当前图片。在浏览完成后，可以单击返回键，跳转回相册显示。

本实例运行效果如图 11-10、图 11-11 所示。

图 11-10　相册界面效果图

提示：在图 11-10 中从左至右依次为运行本软件进入主界面，通过滑动手机屏幕的方式改变物体的呈现角度。

提示：在本软件中，可通过单击操作来查看相册中的某一张图片，按返回键则会返回相册显示。单击图 11-10 右侧图中两张图片的显示效果如图 11-11 所示。

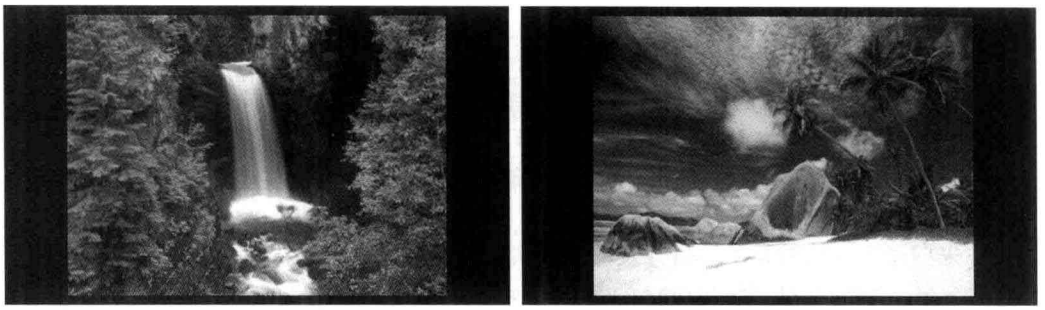


图 11-11　单击显示图片效果

【实现过程】

在本项目开发过程中，用了 OpenGL ES 中的图形绘制技术，包括纹理映射技术、物体旋转技术 glRotatef()、物体平移技术 glTranslatef()，并且应用了屏幕触控技术。

【代码解析】

首先向读者介绍的是本项目开发的 Activity 类的开发过程。

代码位置：Sample11_8/src/com.bn.chap11.xc/Sample11_8_Activity.java

```
1   package com.bn.chap11.xc;                    //包名
2   import android.app.Activity; //包引用，该处有所省略，读者可自行查看随书光盘中的源代码
3   public class Sample11_8_Activity extends Activity {
4       private MySurfaceView mGLSurfaceView;        //声明界面对象
5       public void onCreate(Bundle savedInstanceState) {
6       super.onCreate(savedInstanceState);          //继承父类的 onCreate()方法
7       requestWindowFeature(Window.FEATURE_NO_TITLE); //设置为全屏
8       getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN ,
9       WindowManager.LayoutParams.FLAG_FULLSCREEN);
10      this.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
                                                     //设置为横屏
11      DisplayMetrics dm=new DisplayMetrics();      //创建 DisplayMetrics 对象
12      getWindowManager().getDefaultDisplay().getMetrics(dm);
                                                //获取 DisplayMetrics 对象矩阵
13      mGLSurfaceView = new MySurfaceView(this);    //创建界面对象
14      mGLSurfaceView.screenHeight=dm.heightPixels;  //获取当前界面纵向分辨率
15      mGLSurfaceView.screenWidth=dm.widthPixels;    //获取当前界面横向分辨率
16
mGLSurfaceView.ratio=mGLSurfaceView.screenWidth/mGLSurfaceView.screenHeight;
17      mGLSurfaceView.requestFocus();               //获取焦点
18      mGLSurfaceView.setFocusableInTouchMode(true); //设置为可触控
19      setContentView(mGLSurfaceView);              //显示界面
20    }
21    protected void onResume() {                    //重写 onResume()方法
22      super.onResume();
23      GLSurfaceView.onResume();                    //界面恢复操作
25    }
26    protected void onPause() {                     //重写 onPause()方法
27      super.onPause();
28      Constant.threadWork=false;
29      mGLSurfaceView.onPause();                    //界面暂停操作
30    }   }
```

其中：

- 第 4 行为声明界面对象。第 5～20 行为重写 onCreate()方法，其中，第 7～9 行为全屏显示设置；第 10 行为横屏显示设置；第 11～15 行为获取当前屏幕的分辨率；第 16～19 行为获取屏幕焦点并使其可触控显示。
- 第 21～25 行为重写 onResume()方法，使界面从暂停状态恢复显示。第 26～30 行为重写 onPause()方法，是界面暂停显示。

接下来向读者介绍的是本项目开发的界面实现类的开发过程。

代码位置：Sample11_8/src/com.bn.chap11.xc/ MySurfaceView.java

```
1   package com.bn.chap11.xc;                    //包名
2   import java.io.IOException;                  //包引用，该处有所省略，读者可自行查看源代码
3   class MySurfaceView extends GLSurfaceView {
4       public float screenWidth;                          //屏幕宽带
5       public float screenHeight;                         //屏幕高度
6       public float ratio;                                //屏幕宽高比
7       private final float TOUCH_SCALE_FACTOR = 180.0f/320 //角度缩放比例
8       private SceneRenderer mRenderer;                   //场景渲染器
9       float mPreviousX;                                  //上次按下位置 x 坐标
10      float mPreviousY;                                  //上次按下位置 y 坐标
11      long previousTime;                                 //上次按下的时间
12      boolean isCheck=false;                             //是否单击查看图片
13      boolean isMove=false;                              //是否为屏幕滑动事件
14      int[] textureIds=new int[PHOTO_COUNT];             //n 张照片纹理 Id 数组
15      float yAngle=0;                                    //总场景旋转角度
16      int currIndex=0;                                   //当前选中的索引
17      float yAngleV=0;                                   //总场景角度变化速度
18  public MySurfaceView(Context context) {
19      super(context); mRenderer = new SceneRenderer();          //创建场景渲染器
20      setRenderer(mRenderer);                                   //设置渲染器
21      setRenderMode(GLSurfaceView.RENDERMODE_CONTINUOUSLY);
                                                           //设置渲染模式为主动渲染
22  threadWork=true;
23      new Thread(){
24      ... ...//启动一个线程根据当前的角速度旋转场景
25      }.start();
26  }
27  public boolean onKeyDown(int keyCode, KeyEvent e) {
28      if(keyCode==4){                                    //若按下的是返回键
29      if(isCheck){                                       //若在 detail 场景中则回主场景
30      isCheck=false;
31  }
32  else{                                                  //若在主场景中则退出程序
33      System.exit(0);
34  }   }
35  return true;
36  }
37  public boolean onTouchEvent(MotionEvent e){
38    .. ...                                               //触摸事件回调方法
39  }
40  private class SceneRenderer implements GLSurfaceView.Renderer{
41   ... ...                                               //渲染器内部类开发
42  }
43  public int initTexture(GL10 gl,int drawableId)//textureId{
44  ... ...                                                //初始化纹理
45  }     }
```

其中：

- 第 4～17 行为成员变量的声明，包括屏幕分辨率记录参数声明，触控标志位声明和渲染器对象声明等。
- 第 18～26 行为界面类的构造器部分，其中第 19～21 行为设置渲染器对象。第 23～25 行为根据当前的角速度旋转场景的线程。
- 第 27～36 行为按键单击事件。第 37～39 行为屏幕触控事件回调方法。第 40～42 行为渲染器内部类方法。第 43～45 行为初始化纹理的方法。

然后向读者介绍的是根据当前的角速度旋转场景的线程开发过程和纹理 Id 的初始化开发过程。

代码位置：Sample11_8/src/com.bn.chap11.xc/ MySurfaceView.java

```
1   new Thread(){                                    //启动一个线程根据当前的角速度旋转场景
2   public void run(){
3       while(threadWork){                           //循环判断
4           if(Float.isNaN(yAngle)||Float.isNaN(yAngleV)){
5           throw new RuntimeException("yangle "+yAngle+"yAngleV="+yAngleV);
6   }
7   yAngle+=yAngleV;
8   if(Float.isNaN(yAngle)){                         //根据角速度计算新的场景旋转角度
9       hrow new RuntimeException("yangle"+yAngle);
10  }
11  yAngle=(yAngle+360)%360;                         //将角度规格化到 0～360° 之间
12  if(Float.isNaN(yAngle)||Float.isNaN(yAngleV)){
13      throw new RuntimeException("yangle "+yAngle+"yAngleV="+yAngleV);
14  }
15  if(!isMove){                                     //若当前手指已经抬起则角速度衰减
16      yAngleV=yAngleV*0.7f;
17  }
18  if(Math.abs(yAngleV)<0.05){                      //若角速度小于阈值则归 0
19      yAngleV=0;
20  }
21  try {
22      Thread.sleep(50);
23  } catch (InterruptedException e) {
24      e.printStackTrace();                                          //异常处理
25  }  }  }.start();
26  public int initTexture(GL10 gl,int drawableId)//textureId{    //初始化纹理
27      int[] textures = new int[1];                                  //生成纹理 Id
28      gl.glGenTextures(1, textures, 0);                             //自动获取纹理 Id
29      int currTextureId=textures[0];          //将获取的纹理 Id 存放在纹理 Id 数组中
30      gl.glBindTexture(GL10.GL_TEXTURE_2D, currTextureId);          //绑定纹理 Id
31      gl.glTexParameterf(GL10.GL_TEXTURE_2D,
32      GL10.GL_TEXTURE_MIN_FILTER,GL10.GL_NEAREST);
33      gl.glTexParameterf(GL10.GL_TEXTURE_2D,
34      GL10.GL_TEXTURE_MAG_FILTER,GL10.GL_LINEAR);
35      gl.glTexParameterf(GL10.GL_TEXTURE_2D,
36      GL10.GL_TEXTURE_WRAP_S,GL10.GL_CLAMP_TO_EDGE);
37      gl.glTexParameterf(GL10.GL_TEXTURE_2D,
38      GL10.GL_TEXTURE_WRAP_T,GL10.GL_CLAMP_TO_EDGE);
39      InputStream is = this.getResources().openRawResource(drawableId);
                                                    //位图流数据
40      Bitmap bitmapTmp;
41      try {
42          bitmapTmp = BitmapFactory.decodeStream(is);           //得到位图对象
43      }
44      finally {
```

```
45            try {
46                is.close();                                    //关闭流
47              }
48             catch(IOException e) {
49                    e.printStackTrace();                      //异常处理
50        }      }
51    GLUtils.texImage2D(GL10.GL_TEXTURE_2D, 0, bitmapTmp, 0);//指定纹理 Id 对应位图
52        bitmapTmp.recycle();                                   //回收位图资源
53        return currTextureId;
54    }
```

其中：

- 第 1～25 行为根据当前的角速度旋转场景的线程。首先根据 threadWork 的值来判断是否进行线程控制。
- 第 26～54 行为纹理 Id 初始化方法。首先生成纹理 Id，并自动获取 Id 值，然后绑定纹理 Id 并设置纹理属性，最后通过资源流数据获取当前位图资源并将其与纹理 Id 绑定。进而用此纹理 Id 来表示当前位图对象。

之后向读者介绍的是屏幕触控事件的开发过程。

代码位置：Sample11_8/src/com.bn.chap11.xc/ MySurfaceView.java

```
1    public boolean onTouchEvent(MotionEvent e){      //触摸事件回调方法
2    if(isCheck){                                     //若在 detail 中不处理触控事件
3        return true;
4    }
5    float x = e.getX();                              //获取触控点 X 坐标
6    float y = e.getY();                              //获取触控点 Y 坐标
7    float dx = x - mPreviousX;                       //计算 X 向触控位移
8    float dy = y - mPreviousY;                       //计算 Y 向触控位移
9    long currTime=System.currentTimeMillis();        //获取当前时间戳
10   long timeSpan=(currTime-previousTime)/10;        //计算两次触控事件之间的时间差
11   switch (e.getAction()) {
12       case MotionEvent.ACTION_DOWN:                //按下事件
13           isMove=false;
14           break;
15       case MotionEvent.ACTION_MOVE:                //滑动事件
16           if(Math.abs(dx)>5||Math.abs(dy)>5) {     //触控位移大于阈值则进入移动状态
17               isMove=true;
18           }
19           if(isMove){                              //若在移动状态则计算角度变化速度
20               if(timeSpan!=0){
21                   yAngleV=dx * TOUCH_SCALE_FACTOR/timeSpan;
22     }   }
23       break;
24       case MotionEvent.ACTION_UP:                  //触控抬起事件
25           if(!isMove&&yAngleV==0){  //若在非移动状态且角度速度为 0，则看选中的是哪幅
照片来显示
26               float nearX=(x-screenWidth/2)*ratio/(screenWidth/2);
                                      //折算出触控点在 NEAR 面上的位置
27               float nearY=(screenHeight/2-y)/(screenHeight/2);
28                   if(x>screenWidth/2){  //先判断点下去的是左边还是右边，若是右边
29                       ArrayList<CandidateDis> al=new ArrayList<CandidateDis>();
30                       for(int i=0;i<PHOTO_COUNT;i++){
31                           float tempAngle=(i*PHOTO_ANGLE_SPAN+yAngle)%360;
32                           if(tempAngle>270&&tempAngle<360){
33                               al.add(new CandidateDis(tempAngle-270,
tempAngle,i));
34       }    }
```

```
35      Collections.sort(al);                          //根据与 270° 的夹角排序，夹角小的排在前面
36      currIndex=-1;                                  //遍历候选列表，谁在 X 范围内谁单独显示
37      for(CandidateDis cd:al){
38          if(cd.isInXRange(nearX,nearY)){
39              currIndex=cd.index;
40      break;
41      }   }   }
42      else{
43          ArrayList<CandidateDis> al=new ArrayList<CandidateDis>();
44          for(int i=0;i<PHOTO_COUNT;i++){
45              float tempAngle=(i*PHOTO_ANGLE_SPAN+yAngle)%360;
                                                       //计算此幅照片的角度
46              if(tempAngle>180&&tempAngle<270){
                                                  //若角度在 180° 到 270° 范围内，则在左边的前面
47              al.add(new CandidateDis(270-tempAngle,tempAngle,i));
48      }       }
49  Collections.sort(al);                              //根据与 270° 的夹角排序，夹角小的排在前
50  currIndex=-1;                                      //遍历候选列表谁在 X 范围内谁单独显示
51  for(CandidateDis cd:al) {                          //遍历
52      if(cd.isInXRange(nearX,nearY)){
53          currIndex=cd.index;
54  break;
55  }   }   }
56      if(currIndex!=-1){                             //若有选中照片，则设置进入 detail 显示状态
57          isCheck=true;
58  }     }
59  isMove=false;
60  break;
61  }
62  mPreviousX = x;                                    //记录触控笔位置
63  mPreviousY = y;                                    //记录触控笔位置
64  previousTime=currTime;                             //记录此次时间
65  return true;
66  }
```

其中：

- 第 2～3 行为表示若在 detail 中不处理触控事件。第 5～10 行为获取当前触控点坐标，并计算出当前触控点与上一触控点的距离，获取当前时间戳，进而计算两次触控事件之间的时间差。
- 第 12～14 行为屏幕触控单击事件处理方法。第 15～23 行为屏幕触控滑动事件处理方法。第 24～60 行为屏幕触控抬起事件处理方法。第 62～64 行为记录本次触控点位置和本次时间戳。

接下来向读者介绍的是渲染器内部类的开发过程。

代码位置：Sample11_8/src/com.bn.chap11.xc/ MySurfaceView.java

```
1   private class SceneRenderer implements GLSurfaceView.Renderer{
2   Board tp=new Board();                              //用于显示照片的纹理矩形
3   public void onDrawFrame(GL10 gl) {
4       gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
5       gl.glMatrixMode(GL10.GL_MODELVIEW);            //设置当前矩阵为模式矩阵
6       gl.glLoadIdentity();                           //设置当前矩阵为单位矩阵
7       if(isCheck){                                   //显示某一幅照片，detail 状态
8           gl.glPushMatrix();
9           gl.glTranslatef(0, 0f, -52f);              //平移操作
10          gl.glTranslatef(-Board.length-Board.width*0.5f, 0, 0);
11          tp.drawSelf(gl,textureIds[currIndex]); //绘制
12          gl.glPopMatrix();
```

```
13        }
14    else{                                                    //显示照片组，可触控旋转选择
15        gl.glPushMatrix();
16        gl.glTranslatef(0, 0f, CENTER_Z);
17        yAngle=yAngle%360;
18        gl.glRotatef(yAngle, 0, 1, 0);                       //旋转操作
19        for(int i=0;i<textureIds.length;i++){                //遍历纹理数组，显示每幅照片
20        gl.glPushMatrix();
21        gl.glRotatef(i*PHOTO_ANGLE_SPAN, 0, 1, 0);
22        tp.drawSelf(gl,textureIds[i]);                       //绘制
23        gl.glPopMatrix();
24    }
25    gl.glPopMatrix();
26    }     }
27    public void onSurfaceChanged(GL10 gl, int width, int height) {
28        gl.glViewport(0, 0, width, height);                  //设置视窗大小及位置
29        gl.glMatrixMode(GL10.GL_PROJECTION);                 //设置当前矩阵为投影矩阵
30        gl.glLoadIdentity();                                 //设置当前矩阵为单位矩阵
31        gl.glFrustumf(-ratio, ratio, -1, 1, NEAR, 100);
                                                    //调用此方法计算产生透视投影矩阵
32        gl.glDisable(GL10.GL_CULL_FACE);                     //设置为关闭背面剪裁
33    }
34    ublic void onSurfaceCreated(GL10 gl, EGLConfig config){
35        gl.glDisable(GL10.GL_DITHER);                        //关闭抗抖动
36        gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT,GL10.GL_FASTEST);
                                                             //设置快速模式
37        gl.glClearColor(0,0,0,0);                            //设置屏幕背景色为黑色 RGBA
38        gl.glEnable(GL10.GL_DEPTH_TEST);                     //启用深度测试
39        textureIds[0]=initTexture(gl,R.drawable.fj1);        //加载 n 幅纹理图，图片 1
40        textureIds[1]=initTexture(gl,R.drawable.fj2);                  //图片 2
41        extureIds[2]=initTexture(gl,R.drawable.fj3);                   //图片 3
42        textureIds[3]=initTexture(gl,R.drawable.fj4);                  //图片 4
43        textureIds[4]=initTexture(gl,R.drawable.fj5);                  //图片 5
44        textureIds[5]=initTexture(gl,R.drawable.fj6);                  //图片 6
45        textureIds[6]=initTexture(gl,R.drawable.fj7);                  //图片 7
46        textureIds[7]=initTexture(gl,R.drawable.fj8);                  //图片 8
47        textureIds[8]=initTexture(gl,R.drawable.fj9);                  //图片 9
48        textureIds[9]=initTexture(gl,R.drawable.fj10);                 //图片 10
49        textureIds[10]=initTexture(gl,R.drawable.fj11);                //图片 11
50        textureIds[11]=initTexture(gl,R.drawable.fj12);                //图片 12
51        textureIds[12]=initTexture(gl,R.drawable.fj13);                //图片 13
52        txtureIds[13]=initTexture(gl,R.drawable.fj14);                 //图片 14
53    }    }
```

其中：

- 第 2 行，为创建所要在场景中渲染的物体对象。第 3～26 行为重写 onDrawFrame()方法，用于物体的绘制。
- 第 27～33 行为重写 onSurfaceChanged()方法，用于当界面变化时物体的绘制设置。第 34～53 行为重写 onSurfaceCreated()方法，用于加载场景中要创建物体所应用的资源。

然后向读者介绍屏幕中图片的拾取操作实现类的开发过程。

代码位置：Sample11_8/src/com.bn.chap11.xc/ CandidateDis.java

```
1   package com.bn.chap11.xc;
2   import static com.bn.chap11.xc.Constant.*;
3   public class CandidateDis
4   implements Comparable<CandidateDis>{ //可能被拾取选中照片的描述信息存储对象所属类
```

```
5       float currAngleSpan;                                    //此幅照片与 270° 角的夹角
6       float currAngle;                                        //此幅照片的角度
7       int index;                                              //此幅照片的索引
8       public CandidateDis(float currAngleSpan,float currAngle,int index){
9           this.currAngleSpan=currAngleSpan;
10          this.currAngle=currAngle;
11          this.index=index;
12      }
13  public int compareTo(CandidateDis another) {
14      if(this.currAngleSpan<another.currAngleSpan){ //比较两幅照片中哪幅与 270° 的夹角小
15          return -1;
16      }
17      if(this.currAngleSpan==another.currAngleSpan){
18          return 0;
19      }
20      if(this.currAngleSpan>another.currAngleSpan){
21      return 1;
22  }
23      return 0;
24  }
25  public boolean isInXRange(float x,float y){//xy 为 NEAR 面上的触控坐标
26      double xn=Board.length*Math.cos
27      (Math.toRadians(currAngle));            //计算照片内侧点（靠近中心）的 X 坐标
28      double xw=(Board.length+Board.width)
29      *Math.cos(Math.toRadians(currAngle));  //计算照片外侧点（远离中心）的 X 坐标
30      double zn=-Board.length*Math.sin(Math.toRadians(currAngle))
31      +CENTER_Z;                              //计算照片内侧点（靠近中心）的 Z 坐标
32      double zw=-(Board.length+Board.width)
33      *Math.sin(Math.toRadians(currAngle))+CENTER_Z;
                                                //计算照片外侧点（远离中心）的 Z 坐标
34       double proj_xn=-NEAR*xn/zn;
                        //根据等比三角形原理分别计算两个点在 NEAR 面上的 X 坐标投影
35      double proj_xw=-NEAR*xw/zw;
36      double xmax=Math.max
37      proj_xn, proj_xw); //分别求出两个投影 X 坐标中大的和小的，有利于进行范围逻辑判断
38      double xmin=Math.min(proj_xn, proj_xw);
39      double k=x/NEAR;                        //触控处的 X 投影范围
40      double p=xn/(zn-CENTER_Z);
41      double zq=CENTER_Z*p/(p+k);
42      double xq=-k*zq;
43      double oa=Math.sqrt(x*x+NEAR*NEAR);
44      double ob=Math.sqrt(xq*xq+zq*zq);
45      double yq=oa*Board.height/ob;
46      if(x<xmax&&x>xmin){                     //若在范围内返回 true
47          if(y>-yq&&y<yq){
48              return true;
49       }
50      else{
51          return false;
52       }    }
53      else{                                   //若不在范围内返回 false
54          return false;
55  }   }    }
```

其中：

- 第 5～7 行为成员变量声明，包括当前图片的角度值和索引值声明。
- 第 8～12 行为该类的构造器部分，其中传入三个对象，此幅照片与 270° 角的夹角、此幅照片的角度和此幅照片的索引。
- 第 13～24 行为比较两幅照片哪幅与 270° 的夹角小的方法。第 25～55 行为判断是否触

控了当前图片。

接着向读者介绍的是本项目开发中物体构造类的开发过程。

代码位置：Sample11_8/src/com.bn.chap11.xc/ Board.java

```
1   package com.bn.chap11.xc;
2   import java.nio.ByteBuffer;
3   public class Board {                                    //显示照片的纹理矩形
4       private FloatBuffer  mVertexBuffer;                 //顶点坐标数据缓冲
5       private FloatBuffer  mTextureBuffer;                //顶点纹理数据缓冲
6       int vCount=0;//顶点数
7   public static final float length=3f;            //0° 位置图片左侧距离原点的横向距离
8   public static final float width=20f;                    //图片的宽
9    public static final float height=15f*0.5f;             //图片的半高度
10   public Board(){
11       vCount=6;
12       float vertices[]=new float[] {
13       length,height,0,
14       length,-height,0,
15       length+width,height,0,
16       length+width,height,0,
17       length,-height,0,
18       length+width,-height,0
19   };
20   ByteBuffer vbb = ByteBuffer.allocateDirect(vertices.length*4);
                                                            //创建顶点坐标数据缓冲
21   vbb.order(ByteOrder.nativeOrder());                    //设置字节顺序
22   mVertexBuffer = vbb.asFloatBuffer();                   //转换为 int 型缓冲
23   mVertexBuffer.put(vertices);                           //向缓冲区中放入顶点坐标数据
24   mVertexBuffer.position(0);                             //设置缓冲区起始位置
25   float textures[]=new float[] {                  //顶点颜色值数组，每个顶点 4 个色彩值 RGBA
26           0,0,
27           0,1,
28           1,0,
29           1,0,
30           0,1,
31           1,1
32    };
33    ByteBuffer cbb = ByteBuffer.allocateDirect(textures.length*4);
                                                  //创建顶点纹理数据缓冲
34    cbb.order(ByteOrder.nativeOrder());                   //设置字节顺序
35    mTextureBuffer = cbb.asFloatBuffer();                 //转换为 int 型缓冲
36    mTextureBuffer.put(textures);                         //向缓冲区中放入顶点着色数据
37    mTextureBuffer.position(0);                           //设置缓冲区起始位置
38   }
39   public void drawSelf(GL10 gl,int texId) {
40       gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);        //启用顶点坐标数组
41       gl.glVertexPointer(3,GL10.GL_FLOAT,0, mVertexBuffer);
                                                            //为画笔指定顶点坐标数据
42       gl.glEnable(GL10.GL_TEXTURE_2D);                   //开启纹理
43       gl.glEnableClientState(GL10.GL_TEXTURE_COORD_ARRAY);
                                                            //允许使用纹理 ST 坐标缓冲
44       gl.glTexCoordPointer(2, GL10.GL_FLOAT, 0, mTextureBuffer);
                                                            //为画笔指定纹理 ST 坐标缓冲
45       gl.glBindTexture(GL10.GL_TEXTURE_2D, texId);              //绑定当前纹理
46       gl.glDrawArrays(GL10.GL_TRIANGLES,0,vCount);              //绘制图形
47       gl.glDisableClientState(GL10.GL_TEXTURE_COORD_ARRAY);     //关闭纹理
48       gl.glDisable(GL10.GL_TEXTURE_2D);
49       gl.glDisableClientState(GL10.GL_VERTEX_ARRAY);
```

```
50    }    }
```

其中：

- 第 4～9 行为成员变量的声明。包括顶点坐标数据缓存对像、顶点纹理坐标数据缓存对象和位图纹理 Id。
- 第 10～38 行为该 3D 物体构造类的构造器的开发部分。其中，第 11～24 行为顶点坐标数据缓存设置；第 25～37 行为顶点纹理坐标数据缓存设置。
- 第 39～50 行为物体的绘制方法，在绘制前，首先要指定顶点坐标数据缓冲和纹理坐标数据缓冲。在绘制完成后，关闭纹理缓冲和顶点缓冲。

最后向读者介绍的是常量类 Constant.java 类的开发实现过程。

代码位置：Sample11_8/src/com.bn.chap11.xc/ Constant.java

```
1   package com.bn.chap11.xc;
2   public class Constant {
3       public static final float NEAR=6.5f;                    //摄像机的 Near 参数
4       public static final float CENTER_Z=-80f;                //场景中心的 Z 坐标
5       public static boolean threadWork;                       //惯性线程工作标志位
6       public static final int PHOTO_COUNT=14;                 //照片数量
7       public static final float PHOTO_ANGLE_SPAN=360.0f/PHOTO_COUNT;
                                                                //照片间角度跨度
8   }
```

提示：该类为常量类。用于存放提供一些常用的常量数据。常量类的应用，方便管理常量并且便于改动代码。

小结

本章主要介绍了 OpenGL ES 渲染 Android 手机的 3D 世界，首先以三角形的实例进行介绍，此三角形绘制采用的是 GL_TRIANGLES 模式，重要的是在既定的观察方向上渲染三角形，否则就可能显示不出图形的结果。

然后又介绍了 3D 中摄像机与雾、纹理映射等效果的开发，读者通过以上的学习可以熟练地编写 3D 功能各异的小程序。

第 12 章　手机特效开发

手机使用者在使用手机软件时，首先体验到的是应用软件的酷炫程度，然后是体验该软件的功能与性能。可见，在手机软件中软件的可观赏性的重要性。在本章中将要详细介绍基于 Android 平台的手机的特效开发，如何在手机软件开发中巧妙地运用这些特效将是本章重要内容。

实例 1　虚线特效的开发

本节将要介绍在 Android 手机中绘制自定义虚线多边形，通过继承 SurfaceView 绘制不同形状的三角形从而讲述多边形的绘制方法。

【实例描述】

在本软件中通过继承自 SurfaceView 绘制不同形状的三角形，首先创建出三角形的路径，然后初始化不同三角形的线形效果，最后绘制在手机屏幕上。本实例的运行效果如图 12-1、图 12-2 和图 12-3 所示。

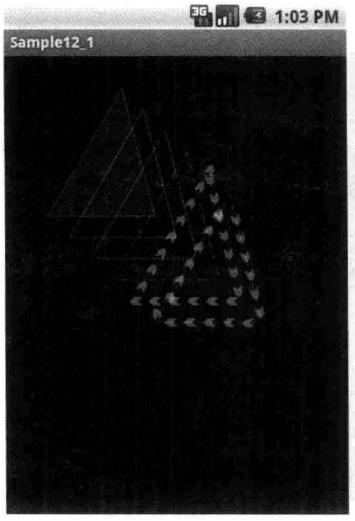


图 12-1　开始界面

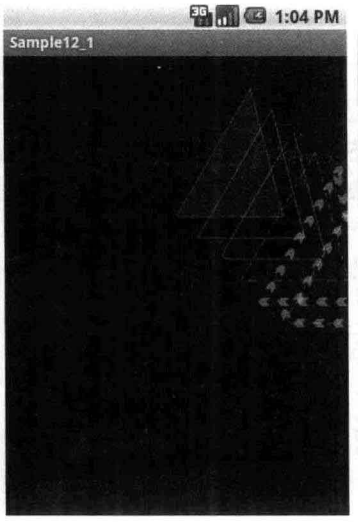


图 12-2　图片从右侧退出

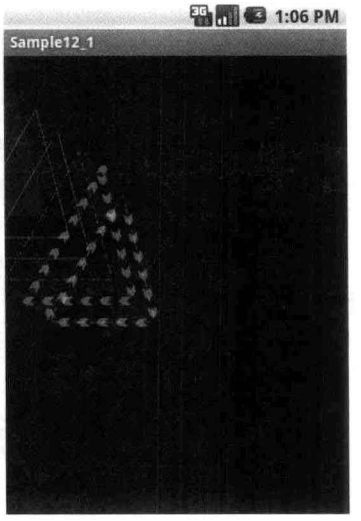


图 12-3　图片从左侧进入

提示：本程序的运行效果为图 12-1、图 12-2 和图 12-3，其中图 12-1 表示的是程序开始时的界面，图 12-2 表示的是图片运动至右侧退出的界面，图 12-3 表示的是图片从左侧进入的界面。

【实现过程】

本软件继承 SurfaceView 进行三角形的绘制，首先创建 Path 进行三角形路径设置，然后调

用 initPathEffect()方法设置三角形的线形效果。

【代码解析】

在本部分将详细介绍该软件实现过程，首先要介绍的是 Sample12_1_Activity 类，代码如下。

代码位置：见随书光盘中源代码/第 12 章/Sample12_1/src/com/bn/ex12a 目录下的 Sample12_1_Activity.java。

```
1   package com.bn.ex12a;                                          //包名声明
2   import android.app.Activity;                                   //相关类的引入
3   import android.os.Bundle;                                      //相关类的引入
4   public class Sample12_1_Activity extends Activity {
5       public void onCreate(Bundle savedInstanceState) {
6           super.onCreate(savedInstanceState);                    //调用父类
7           MySurfaceView mySurfaceView=new MySurfaceView(this);
                                                                   //创建 MySurfaceView
8           setContentView(mySurfaceView);                         //设置布局
9   }     }
```

提示：上述代码为主控制类的介绍，此类主要创建 MySurfaceView 对象引用并跳转到主界面。

上述代码介绍了本软件的 Sample12_1_Activity 类的开发，接下来介绍绘制界面类 MySurfaceView 框架的开发，代码如下。

代码位置：见随书光盘中源代码/第 12 章/Sample12_1/src/com/bn/ex12a 目录下的 MySurfaceView.java。

```
1   package com.bn.ex12a;                                          //声明包
2   import android.graphics.Canvas;                                //导入相关类
3   ......// 此处省略了部分类的引入代码，请读者自行查看随书光盘源代码
4   import android.view.SurfaceView;                               //导入相关类
5   public class MySurfaceView extends SurfaceView                 //继承 SurfaceView
6   implements SurfaceHolder.Callback                              //实现生命周期回调接口
7   {
8       Sample12_1_Activity activity;
9       Paint paint;                                               //画笔
10      Path mPatha = new Path();                                  //多边形路径-三角形 1
11      Path mPathb = new Path();                                  //多边形路径-三角形 2
12      Path mPathc = new Path();                                  //多边形路径-三角形 3
13      Path mPathd = new Path();                                  //多边形路径-三角形 4
14      Path mPathe = new Path();                                  //多边形路径-三角形 5
15      Path mPathf = new Path();                                  //多边形路径-三角形 6
16      Path mPathg = new Path();                                  //多边形路径-三角形 7
17      PathEffect[] peArray;                                      //声明 PathEffect 数组
18      boolean workFlag=true;
19      int xOffset=0;                                             //声明成员变量 xOffset
20      public MySurfaceView(Sample12_1_Activity activity) {       //构造器
21          super(activity);                                       //调用父类
22          this.activity = activity;                              //为成员变量赋值
23          this.getHolder().addCallback(this);                    //生命周期回调
24          paint = new Paint();                                   //创建画笔
25          paint.setAntiAlias(true);                              //打开抗锯齿
26          initPath(0);                                           //调用 initPath 方法
27         initPathEffect();                                       //调用 initPathEffect 方法
```

```
28          new Thread(){                                          //线程
29           public void run(){                                    //重写的方法
30               while(workFlag){                                  //workFlag 为 true 时
31                   repaint();                                    //刷帧方法
32                   xOffset=(xOffset+10)%320;                     //改变 xOffset 的值
33                   initPath(xOffset);                            //调用 initPath 方法
34                   try {
35                           Thread.sleep(3000);                   //休息 3000ms
36                       }catch (InterruptedException e){          //捕获异常
37                           e.printStackTrace();                  //打印堆栈信息
38          }}}}.start();                                          //线程开启
39      }
40      /*该处省略了设置路径的相关方法,, 将在下面的步骤给出*/
41      public void initPath(int xOffset){                         //声明方法
42          ......//该处省略了设置三角形的路径代码，读者可自行查看随书光盘源代码
43      }
44      /*此处省略了绘制图片的方法,, 将在下面的步骤给出*/
45      public void surfaceChanged(SurfaceHolder arg0, int arg1, int arg2, int
arg3){ }
46      public void surfaceCreated(SurfaceHolder holder){          //创建时被调用
47          repaint();                                             //调用刷帧方法
48      }
49      public void repaint(){                                     //刷帧方法
50          SurfaceHolder holder=this.getHolder();          //获取 SurfaceHolder 对象
51          Canvas canvas = holder.lockCanvas();                   //获取画布
52          try{
53              synchronized(holder){
54                  onDraw(canvas);                                //绘制
55              }
56          }catch(Exception e){                                   //捕获异常
57              e.printStackTrace();                               //打印堆栈信息
58          }finally{
59              if(canvas != null){                                //判断 canvas 是否为空
60                  holder.unlockCanvasAndPost(canvas);            //解除锁定
61      }}}
62      public void surfaceDestroyed(SurfaceHolder arg0) {         //销毁时被调用
63          workFlag=false;                                        //设置线程标志为 false
64  }}
```

其中：

- 第 8～19 行表示对该类的成员变量的声明。
- 第 20～39 行表示该类的构造器，在构造器中主要是初始化用到的数据，同时开启线程定时更新 xOffset 的值，使得图片移动。
- 第 46～48 行表示创建 SurfaceView 时最先调用的方法，并调用 repaint 方法。
- 第 49～61 行表示控制刷帧的方法，该方法首先锁定画布，并绘制，最后需要解除画布。

上述代码介绍了 MySurfaeView 类的框架的开发，接下来介绍初始化 6 种三角形线形效果的方法，将此方法插入到 MySurfaceView 第 40 行代码。

代码位置：见随书光盘中源代码/第 12 章/Sample12_1/src/com/bn/ex12a 目录下的 MySurfaceView.java。

```
1   private  Path makePathDash() {                                 //创建私有的方法
2       Path p = new Path();                                       //创建 Path 对象
3       p.moveTo(4, 0);                                            //设置路径
4       p.lineTo(0, -4);                                           //设置路径
5       p.lineTo(8, -4);
```

```
6         p.lineTo(12, 0);                                    //设置路径
7         p.lineTo(8, 4);                                     //设置路径
8         p.lineTo(0, 4);
9         return p;                                           //返回结果
10    }
11    public void initPathEffect() {                          //初始化 6 种线形效果的方法
12        peArray=new PathEffect[6];
13        peArray[0]=null;                                    //不使用效果
14        peArray[1]=new CornerPathEffect(10);                //圆角效果
15        peArray[2] = new DashPathEffect(                    //虚线效果
16            new float[] {5, 5, 10, 5},
17            1                                               //绘制下标
18        );
19        peArray[3] = new PathDashPathEffect(
20            makePathDash(),                                 //形状
21            18,                                             //间距
22            0,                                              //绘制偏移量
23            PathDashPathEffect.Style.ROTATE                 //样式
24        );
25        peArray[4] = new ComposePathEffect(peArray[2], peArray[1]);  //得到对象
26        peArray[5] = new ComposePathEffect(peArray[3], peArray[1]);  //得到对象
27    }
```

提示：上述两个方法初始化每个三角形的线形效果，第 25 和第 26 行是组合了两个三角形。

上述代码介绍初始化 6 种三角形线形效果的方法，接下来介绍绘制界面的方法，将此方法插入到 MySurfaceView 第 44 行代码如下。

代码位置：见随书光盘中源代码/第 12 章/Sample12_1/src/com/bn/ex12a 目录下的 MySurfaceView.java。

```
1   paint.setColor(Color.RED);                                //设置画笔颜色
2   paint.setStyle(Style.STROKE);                             //设置样式
3   paint.setStrokeWidth(3);
4   canvas.drawPath(mPatha, paint);                           //红色三角形
5   paint.setARGB(128, 8, 77, 253);                           //设置画笔颜色
6   paint.setStyle(Style.FILL);                               //设置样式
7   canvas.drawPath(mPatha, paint);
8   paint.setStyle(Style.STROKE);                             //设置样式
9   paint.setARGB(128, 255, 255, 240);                        //设置画笔颜色
10  paint.setPathEffect(peArray[0]);                          //绘制第一个虚线三角形
11  canvas.drawPath(mPathb, paint);
12  paint.setPathEffect(peArray[1]);                          //绘制第二个虚线三角形
13  canvas.drawPath(mPathc, paint);
14  paint.setPathEffect(peArray[2]);                          //绘制第三个虚线三角形
15  canvas.drawPath(mPathd, paint);
16  paint.setPathEffect(peArray[3]);                          //绘制第四个虚线三角形
17  canvas.drawPath(mPathe, paint);
18  paint.setPathEffect(peArray[4]);                          //绘制第五个虚线三角形
19  canvas.drawPath(mPathf, paint);
20  paint.setPathEffect(peArray[5]);                          //绘制第六个虚线三角形
21  canvas.drawPath(mPathg, paint);
22  paint.setPathEffect(null);                                //撤销线形效果设置
```

提示：上述为绘制主界面的方法，同时把 6 个不同的三角形绘制在界面中，最后撤销线形效果设置。

实例 2　切屏动画特效

本节将详细介绍切屏动画特效，在软件开发中加入切屏动画特效使软件更加人性化，没有加入切屏动画特效的软件会显得比较单调，降低本软件的可观赏性。

【实例描述】

进入本软件后，TextView 提示用户进入的是哪个界面，单击“下一界面”按钮进入子界面，子界面中布局与主界面类似，在子界面中单击“上一界面”返回主界面。

本实例的运行效果为如图 12-4、图 12-5、图 12-6 所示。

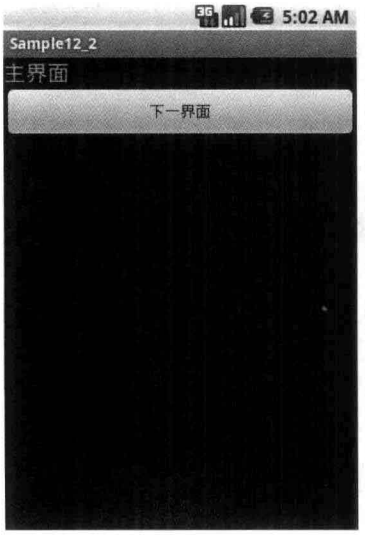


图 12-4　主界面

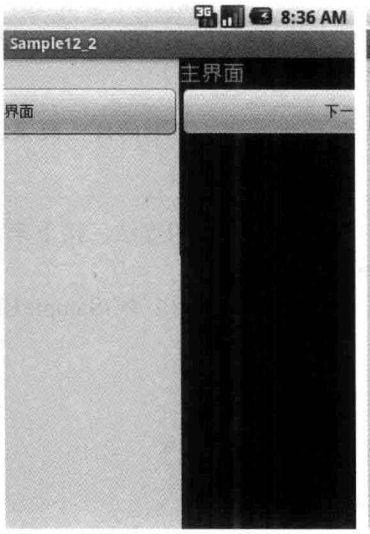


图 12-5　切换中界面

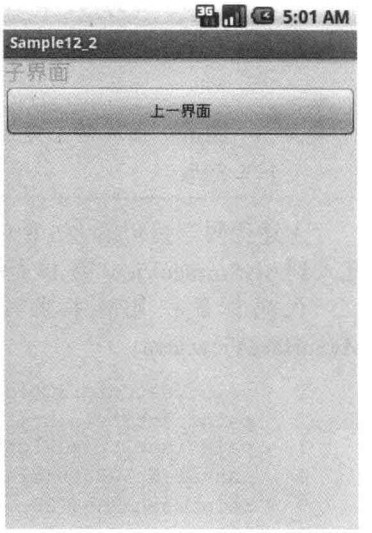


图 12-6　子界面

提示：本程序的运行效果为图 12-4、图 12-5、图 12-6，其中图 12-4 表示的是程序开始时的主界面，图 12-5 表示的是切换中的界面，图 12-6 表示的是由主界面切换进入子界面。

【实现过程】

本实例中运行软件，首先进入主界面。在 TextView 显示提示界面信息，并为 Button 按钮添加监听器，界面跳转的动画功能实现在项目目录中创建 anim 文件夹，然后为动画功能编写 xml 配置文件。

【代码解析】

在本部分会详细介绍本软件的实现过程。首先介绍主界面 xml 文件的设置，其代码如下。

代码位置：见随书光盘中源代码/第 12 章/Sample12_2/ res/layout 目录下的 main.xml。

```
<?xml version="1.0" encoding="utf-8"?>          <!--版本号编码方式-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
                    android:orientation="vertical"
                    android:layout_width="fill_parent"
                    android:layout_height="fill_parent"
                    android:id="@+id/layout01"
    >                                           <!--LinearLayout 的属性设置-->
    <TextView
                    android:layout_width="fill_parent"
                    android:layout_height="wrap_content"
                    android:text="主界面"
                    android:textSize="20dip"
    />                                          <!-- TextView 的属性设置-->
    <Button
                    android:text="下一界面"
                    android:id="@+id/Button01"
                    android:layout_width="fill_parent"
                    android:layout_height="wrap_content"
    >                                           <!-- Button 的属性设置-->
    </Button>
</LinearLayout>
```

提示：上述为主界面的配置文件，子界面和主界面类似，在这里不再赘述。

上述 main.xml 的代码介绍了本软件的主界面的配置，接下来介绍的是进入主界面动画特效的两个配置文件，代码如下。

代码位置：见随书光盘中源代码/第 12 章/Sample12_2/ res/anim 目录下的 myanim_visible_back.xml。

```
<?xml version="1.0" encoding="utf-8"?>          <!--xml 的版本以及编码方式 -->
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <translate
                    android:fromXDelta="100%p"
                    android:toXDelta="0"
                    android:fromYDelta="0"
                    android:toYDelta="0"
                    android:duration="300"
    />                                          <!-- 位置的变换 -->
</set>
```

提示：上述代码为 myanim_visible_back.xml 配置文件。

上述代码介绍了 myanim_visible_back.xml 的配置，接下来介绍的是 myanim_gone_back.xml 的配置，代码如下。

代码位置：见随书光盘中源代码/第 12 章/Sample12_2/ res/anim 目录下的 myanim_gone_back.xml。

```
<?xml version="1.0" encoding="utf-8"?>          <!--xml 的版本以及编码方式 -->
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <translate
                    android:fromXDelta="0"
                    android:toXDelta="-100%p"
                    android:fromYDelta="0"
                    android:toYDelta="0"
                    android:duration="300"
    />                                          <!-- 位置的变换 -->
</set>
```

提示：上述代码为 myanim_gone_back.xml 配置文件。

上述 myanim_visible_back.xml 与 myanim_gone_back.xml 的代码已经介绍了本软件的主界面动画特效的配置，接下来介绍的是进入子界面动画特效的两个配置文件，代码如下。

代码位置：见随书光盘中源代码/第 12 章/Sample12_2/ res/anim 目录下的 myanim_visible_go.xml。

```
<?xml version="1.0" encoding="utf-8"?>          <!--xml 的版本以及编码方式 -->
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <translate
                android:fromXDelta="-100%p"
                android:toXDelta="0"
                android:fromYDelta="0"
                android:toYDelta="0"
                android:duration="300"
    />                                          <!-- 位置的变换 -->
</set>
```

提示：上述代码为 myanim_visible_go.xml 配置文件。

上述代码为 myanim_visible_go.xml 的配置，接下来介绍 myanim_gone_go.xml 配置，代码如下。

代码位置：见随书光盘中源代码/第 12 章/Sample12_2/ res/anim 目录下的 myanim_gone_go.xml。

```
<?xml version="1.0" encoding="utf-8"?>          <!--xml 的版本以及编码方式 -->
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <translate
                android:fromXDelta="0"
                android:toXDelta="100%p"
                android:fromYDelta="0"
                android:toYDelta="0"
                android:duration="300"
    />                                          <!-- 位置的变换 -->
</set>
```

提示：上述代码为 myanim_gone_go.xml 配置文件。

上述 xml 的代码为本软件的所有界面特效的开发，接下来介绍本软件的控制类 Sample12_2_Activity 的开发，代码如下。

代码位置：见随书光盘中源代码/第 12 章/Sample12_2/src/com/bn/ex12b 目录下的 Sample12_2_Activity.java。

```
package com.bn.ex12b;                                   //包的声明
import android.app.Activity;                            //相关类的引入
......//此处省略了部分类的引入代码，请读者自行查看随书光盘源代码
import android.os.Bundle;                               //相关类的引入
public class Sample12_2_Activity extends Activity {
    public void onCreate(Bundle savedInstanceState){    //重写的 onCreate 方法
        super.onCreate(savedInstanceState);             //调用父类
        TransformUtil.initTransorfm(this);              //初始化切换动画
        setContentView(R.layout.main);                  //设置布局
        Button button01=(Button)findViewById(R.id.Button01);
        button01.setOnClickListener(                    //按钮监听
            new OnClickListener(){                      //匿名内部类
```

```
13                  public void onClick(View v) {                    //重写的方法
14                      gotoLast();                                  //去子界面
15      } } ); }
16      public void gotoLast() {                                     //去子界面的方法
17          final Activity activity=Sample12_2_Activity.this;
18          View layoutFrom=findViewById(R.id.layout01);             //取的子布局
19          TransformUtil.goTransorfm(layoutFrom,R.layout.mylayout,R.id.
layout02,activity);
20          Button button02=(Button)findViewById(R.id.Button01);
21          button02.setOnClickListener(                             //子界面按钮监听
22              new OnClickListener(){                               //匿名内部类
23                  public void onClick(View v){                     //重写的 onClick 方法
24                      gotoMain();                                  //去主界面
25      } } ); }
26      public void gotoMain() {                                     //去主界面的方法
27          final Activity activity=Sample12_2_Activity.this;        //常量值
28          View layoutFrom = findViewById(R.id.layout02);           //主界面布局
29
TransformUtil.backTransorfm(layoutFrom,R.layout.main,R.id.layout01,activity);
30          Button b1 = (Button) findViewById(R.id.Button01);        //得到引用
31          b1.setOnClickListener(                                   //为按钮添加监听器
32              new Button.OnClickListener() {
33                  public void onClick(View v) {                    //重写的方法
34                      gotoLast();                                  //去子界面
35      } } ); }
36  }
```

其中：

- 第 11-15 行表示对主界面中按钮添加监听器，单击该按钮跳转到子界面。
- 第 16-25 行表示跳转到子界面的方法，并为子界面中的“上一界面”按钮添加监听器，单击该按钮跳转到主界面。
- 第 26-35 行表示跳转到主界面的方法，并为主界面中的“下一界面”按钮添加监听器，单击该按钮跳转到子界面。

上述代码介绍本软件的控制类 Sample12_2_Activity 的开发，接下来介绍切换界面动画类 TransformUtil 的开发，代码如下。

代码位置：见随书光盘中源代码/第 12 章/Sample12_2/src/com/bn/ex12b 目录下的 TransformUtil.java。

```
1   package com.bn.ex12b;                                            //包的声明
2   import android.app.Activity;                                     //相关类的引入
3   ......//此处省略了部分类的引入代码，请读者自行查看随书光盘中的源代码
4   import android.content.Context;                                  //相关类的引入
5   public class TransformUtil {
6       static Animation visibleAnimationGo;                         //去动画中两个界面子动画
7       static Animation goneAnimationGo;                            //去动画中两个界面子动画
8       static Animation goneAnimationBack;                          //返回动画中两个界面子动画
9       static Animation visibleAnimationBack;                       //返回动画中两个界面子动画
10      public static void initTransorfm(Context context) {         //初始化新界面
11          visibleAnimationGo=AnimationUtils.loadAnimation(context,R.anim.
myanim_visible_go);
12          goneAnimationGo=AnimationUtils.loadAnimation(context,R.anim.
myanim_gone_go);
13          visibleAnimationGo.setFillAfter(false);                  //不可放大
14          goneAnimationGo.setFillAfter(false);                     //不可放大
15          visibleAnimationGo.setInterpolator(new AccelerateInterpolator());
```

```
                                                    //设置动画加速效果
    16          goneAnimationGo.setInterpolator(new AccelerateInterpolator());
                                                    //设置动画加速效果
    17          visibleAnimationBack =AnimationUtils.loadAnimation(context,R.anim.
myanim_visible_back);
    18          goneAnimationBack = AnimationUtils.loadAnimation(context,R.anim.
myanim_gone_back);
    19          goneAnimationBack.setFillAfter(false);      //不可放大
    20          visibleAnimationBack.setFillAfter(false);   //不可放大
    21          visibleAnimationBack.setInterpolator(new AccelerateInterpolator());
                                                    //设置动画加速效果
    22          goneAnimationBack.setInterpolator(new AccelerateInterpolator());
                                                    /设置动画加速效果
    23      }
    24      public static void goTransorfm (                //去子界面的切换方法
    25          View from,                                  //出发界面
    26          int layoutTo,                               //目的界面 Layout
    27          int toId,                                   //目的界面布局 View
    28          Activity activity
    29      ){
    30          from.startAnimation(goneAnimationGo);       //开始去子界面的动画
    31          activity.setContentView(layoutTo);          //设置子界面 Layout
    32          View layoutTarget=activity.findViewById(toId);      //子界面 View
    33          layoutTarget.startAnimation(visibleAnimationGo);    //开始子界面动画
    34      }
    35      public static void backTransorfm (              //返回主界面的切换方法
    36          View from,                                  //出发界面总 View
    37          int toLayout,                               //目标界面 Layout id
    38          int toId,                                   //目标界面总 View id
    39          Activity activity
    40      ){
    41          from.startAnimation(goneAnimationBack);     //开始去主界面动画
    42          activity.setContentView(toLayout);          //设置主界面 Layout
    43          final View layoutFrom =activity.findViewById(toId); //主界面 View
    44          layoutFrom.startAnimation(visibleAnimationBack);    //开始主界面动画
    45  } }
```

其中：

- 第 11～16 行是去子界面两个动画的初始化。
- 第 16～22 行是去主界面两个动画的初始化。
- 第 24～34 行是去子界面的切换方法，此方法接收目的界面的 Layout，布局 View。
- 第 35～45 行是返回主界面的切换方法，此方法接收目的界面的 Layout，布局 View。

实例 3　控件的抖动特效

Android 手机可以通过 xml 配置文件制作控件的抖动特效，在本节中将以 EditText 为实例详细介绍控件的抖动特效。

【实例描述】

在本软件中，首先 TextView 提示用户输入数字内容，然后用户单击“确定”按钮，获取 EditText 中内容进行匹配，匹配成功弹出 Toast 提示，否则 EditText 上下左右抖动。

本实例的运行效果为如图 12-7、图 12-8、图 12-9 所示。

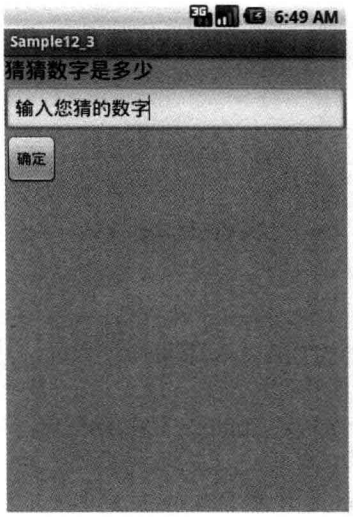


图 12-7　主界面

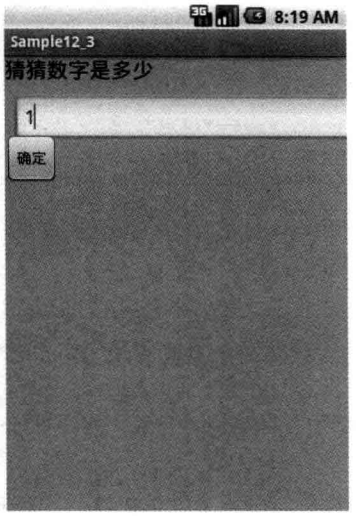


图 12-8　数字不匹配

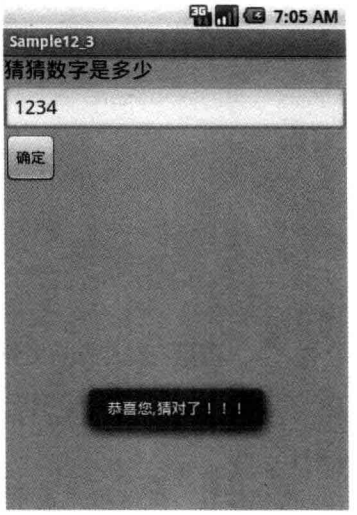


图 12-9　匹配成功

提示：本程序的运行效果为图 12-7、图 12-8、图 12-9，其中图 12-7 表示的是程序开始时的主界面，图 12-8 表示的是输入的数字不匹配，文本框抖动的界面，图 12-9 表示的是输入数字匹配成功的界面。

【实现过程】

本实例中首先单击“确定”按钮调用监听事件，获取用户输入的内容对输入的内容进行匹配，匹配成功弹出 Toast，否则 EditText 上下左右抖动，开始抖动特效则调用 Xml 配置文件。

【代码解析】

在本部分会详细介绍本程序的实现过程，首先介绍本软件的布局文件 main.xml，代码如下：

代码位置：见随书光盘中源代码/第 12 章/Sample12_3 res/layout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>          <!--版本号编码方式-->
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3                   android:orientation="vertical"
4                   android:layout_width="fill_parent"
5                   android:layout_height="fill_parent"
6                   android:background="#ADADAD"
7       >                                           <!--LinearLayout 的属性设置-->
8       <TextView
9                   android:text="猜猜数字是多少"
10                  android:id="@+id/TextView01"
11                  android:layout_width="wrap_content"
12                  android:textColor="#000000"
13                  android:layout_height="wrap_content"
14                  android:textSize="20dip"
15      >                                           <!-- TextView 的属性设置-->
16      </TextView>
17      <EditText
18                  android:text="输入你猜的数字"
19                  android:id="@+id/EditText01"
20                  android:layout_width="fill_parent"
```

```
                android:layout_height="wrap_content"
    >                                          <!-- EditText 的属性设置-->
    </EditText>
    <Button
                android:text="确定"
                android:id="@+id/Button01"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
    >                                          <!-- Button 的属性设置-->
    </Button>
</LinearLayout>
```

提示：上述 Xml 文件为本软件的主界面布局，其中用到了 TextView、EditText、Button。

上述 main.xml 代码介绍了本软件主界面的开发，接下来介绍的是 EditText 的抖动配置，代码如下。

代码位置：见随书光盘中源代码/第 12 章/Sample12_3 res/anim 目录下的 cycle.xml。

```
<?xml version="1.0" encoding="utf-8"?>              <!--版本号编码方式-->
    <cycleInterpolator
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:cycles="2"
    />                                              <!--设置循环-->
```

提示：上述代码为 cycle.xml 配置文件。。

上述 cycle.xml 代码介绍了抖动循环次数配置，接下来介绍的是 shake.xml 的配置，代码如下。

代码位置：见随书光盘中源代码/第 12 章/Sample12_3 res/anim 目录下的 shake.xml。

```
<?xml version="1.0" encoding="utf-8"?>              <!--版本号 编码方式-->
<set
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:interpolator="@anim/cycle"
>
    <translate
                android:fromXDelta="0"
                android:toXDelta="-10"
                android:fromYDelta="0"
                android:toYDelta="-10"
                android:duration="300"
    />                                              <!-- 左上子动画 -->
    <translate
                android:fromXDelta="0"
                android:toXDelta="10"
                android:fromYDelta="0"
                android:toYDelta="-10"
                android:startOffset="300"
                android:duration="300"
    />                                              <!-- 右上子动画 -->
    <translate
                android:fromXDelta="0"
                android:toXDelta="-10"
                android:fromYDelta="0"
                android:toYDelta="10"
                android:startOffset="600"
                android:duration="300"
    />                                              <!-- 左下子动画 -->
    <translate
```

```
30                  android:fromXDelta="0"
31                  android:toXDelta="10"
32                  android:fromYDelta="0"
33                  android:toYDelta="10"
34                  android:startOffset="900"
35                  android:duration="300"
36      />                                              <!-- 右下子动画 -->
37  </set>
```

提示：上述代码为 shake.xml 配置文件，此配置文件设置了 EditText 的动画效果。

上述 xml 代码主要是对 EditText 的配置，接下来介绍主控制类 Sample12_3_Activity 的开发，代码如下。

代码位置：见随书光盘中源代码/第 12 章/Sample12_3/src/com/bn/ex12c 目录下的 Sample12_3_Activity.java。

```
1   package com.bn.ex12c;                                   //包的声明
2   import android.app.Activity;                            //相关类的引入
3   ......//此处省略了部分类的引入，请读者自行查看随书光盘源代码
4   import android.os.Bundle;                               //相关类的引入
5   public class Sample12_3_Activity extends Activity{
6       public void onCreate(Bundle savedInstanceState) {   //重写的方法
7       super.onCreate(savedInstanceState);                 //调用父类
8       setContentView(R.layout.main);                      //设置布局
9       Button button01=(Button)findViewById(R.id.Button01);
10      button01.setOnClickListener(                        //确定按钮监听方法
11          new OnClickListener(){
12              public void onClick(View v){                //重写的 onClick 方法
13                EditText edit01=(EditText)findViewById(R.id.EditText01);
                                                            //EditText 声明
14                String math=edit01.getText().toString().trim();    //取出内容
15                if(math.equals("1234")){                  //判断是否匹配
16                   Toast.makeText(Sample12_3_Activity.this, "恭喜你,猜对了!!! ",
17                     Toast.LENGTH_SHORT).show();          //弹出 Toast
18                }
19                else{                                     //内容不匹配
20                   Animation ai=AnimationUtils.loadAnimation(Sample12_3_
Activity.this,R.anim.shake);
21                     edit01.startAnimation(ai);           //设置 EditText 动画
22  } } } ); } }
```

提示：上述代码中设置了本软件的布局，并对“确定”按钮进行了监听方法，此监听方法中，如果输入的内容与系统值匹配弹出 Toast，否则 Edittext 设置抖动动画。

实例 4　多点触控

Android 手机的多点触控为 Android 平台的一大亮点，本节将详细介绍多点触控技术，在本软件中对一张图片进行收缩、旋转操作实现多点触控，下面的内容是对多点触控的开发。

【实例描述】

运行本程序首先显示的是一张图片，手机可以通过用户对此图片按下的触控点进行计算，

并对图片进行任意的拉伸、缩小、旋转等操作。

本实例的运行效果为如图 12-10、图 12-11、图 12-12 所示。

图 12-10　开始界面

图 12-11　旋转 1

图 12-12　旋转 2

提示：本程序为多点触控，必须在真机上运行。本程序的运行效果为图 12-10、图 12-11、图 12-12，其中图 12-10 表示程序开始运行后图片缩放的界面，图 12-11 与图 12-12 分别表示旋转一定角度的图片。

【实现过程】

在本实例中，通过 SurfaceView 加载图片并绘制到屏幕上，调用 OnTouch 方法对用户触控屏幕事件进行监听，并记录对应的触控点，然后通过触控点计算片图的缩放比例和旋转角度。

【代码解析】

本部分将要介绍本实例的核心代码，首先介绍控制类 Sample12_4_Activity 的开发，代码如下。

代码位置：见随书光盘中源代码/第 12 章/Sample12_4/src/com/bn/ex12d 目录下的 Sample12_4_Activity.java。

```
1   package com.bn.ex12d;                                          //包的声明
2   import android.app.Activity;                                   //相关类的引入
3   import android.content.pm.ActivityInfo;                        //相关类的引入
4   import android.os.Bundle;                                      //相关类的引入
5   import android.util.DisplayMetrics;
6   import android.view.Window;                                    //相关类的引入
7   import android.view.WindowManager;                             //相关类的引入
8   public class Sample12_4_Activity extends Activity {
9       static float screenHeight;                                 //声明变量
10      static float screenWidth;                                  //静态成员变量
11      public void onCreate(Bundle savedInstanceState)  {
12          super.onCreate(savedInstanceState);                    //调用父类
13          requestWindowFeature(Window.FEATURE_NO_TITLE);         //去除标题
14          getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN ,
```

```
        WindowManager.LayoutParams.FLAG_FULLSCREEN);     //设置为全屏
        setRequestedOrientation
        (ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);      //强制为竖屏
        DisplayMetrics dm=new DisplayMetrics();          //获取屏幕宽度高度
        getWindowManager().getDefaultDisplay().getMetrics(dm);
        screenHeight=dm.heightPixels;                    //获取屏幕高度
        screenWidth=dm.widthPixels;                      //获取屏幕宽度
        MySurfaceView mySurfaceView=new MySurfaceView(this);
        this.setContentView(mySurfaceView);              //设置布局
} }
```

提示：上述代码主要是设置屏幕为全屏并且强制竖屏，同时获取屏幕的宽度、高度等操作。

上述代码介绍了本软件的控制类的开发，接下来详细介绍绘制类 MySurfaceView 框架的开发，理解该绘制类的框架有助于读者对本程序的学习，具体代码如下。

代码位置：见随书光盘中源代码/第 12 章/Sample12_4/src/com/bn/ex12d 目录下的 MySurfaceView。

```
package com.bn.ex12d;                                    //包的声明
import android.view.SurfaceHolder;
.....//此处省略了部分类的引入代码，请读者自行查看随书光盘源代码
import android.view.SurfaceView;                         //相关类的引入
public class MySurfaceView extends SurfaceView
 implements SurfaceHolder.Callback{                      //创建类
    Sample12_4_Activity activity;                        //成员变量
    Paint paint;
    Bitmap bitmapTmp;                                    //图片
    float PicHeight;                                     //图片高度
    float PicWidth;                                      //图片宽度
    float currScale=1;                                   //初始缩放比例
    float angle=0;                                       //初始旋转角度
    HashMap<Integer,BNPoint> hm=new HashMap<Integer,BNPoint>();
    float distance=0;                                    //两个触控点之间的距离
    public MySurfaceView(Sample12_4_Activity activity) {     //构造器
        super(activity);
        this.activity=activity;                          //为成员变量赋值
        this.getHolder().addCallback(this);      //设置生命周期回调接口的实现者
        paint=new Paint();                               //创建画笔
        paint.setAntiAlias(true);                        //打开抗锯齿
        bitmapTmp=BitmapFactory.decodeResource(getResources(),R.drawable.bg);
        PicHeight=bitmapTmp.getHeight();                 //图片的高度
        PicWidth=bitmapTmp.getWidth();                   //图片的宽度
    }
    public void onDraw(Canvas canvas){
        paint.setColor(Color.BLACK);                     //设置画笔颜色
        canvas.drawRect(0, 0, 480, 854, paint);          //绘制黑色填充矩形
        float picWidthTemp=PicWidth*currScale;       //动态计算图片的高度和宽度
        float picHeightTemp=PicHeight*currScale;
        float left=(screenWidth-picWidthTemp)/2;         //计算图片显示的坐标
        float top=(screenHeight-picHeightTemp)/2;
        Matrix m1=new Matrix();                          //旋转图片并绘制
        m1.setTranslate(left,top);                       //动态改变图片位置
        Matrix m2=new Matrix();                          //创建 Matrix 对象
        m2.setScale(currScale, currScale);
```

```
37          Matrix mz=new Matrix();                              //设置矩阵
38          mz.setConcat(m1, m2);
39          Matrix m3=new Matrix();                              //创建 Matrix 对象
40          m3.setRotate(angle, screenWidth/2, screenHeight/2); //动态旋转图片
41          Matrix mzz=new Matrix();                             //创建 Matrix 对象
42          mzz.setConcat(m3, mz);                               //设置矩阵
43          canvas.drawBitmap(bitmapTmp, mzz, paint);            //绘制图片
44      }
45      /*此处省略了屏幕触控事件方法，将在下面给出*/
46      public void surfaceChanged(SurfaceHolder holder,
47      int format, int width,int height) {                      //重写的方法
48          this.repaint();                                      //调用绘制方法
49      }
50      public void surfaceCreated(SurfaceHolder holder) { }     //重写的方法
51      public void surfaceDestroyed(SurfaceHolder holder){ }    //重写的方法
52      public void repaint(){                                   //重新绘制界面方法
53          SurfaceHolder holder=this.getHolder();
54          Canvas canvas=holder.lockCanvas();                   //获取画布
55          try{
56              synchronized(holder){                            //判断是否锁定
57                  onDraw(canvas);                              //绘制方法的调用
58              }
59          }catch(Exception e){                                 //捕获异常
60              e.printStackTrace();                             //打印信息
61          }finally{
62              if(canvas != null){                              //canvas 不为空
63                  holder.unlockCanvasAndPost(canvas);          //画布解锁
64  } } } }
```

其中：

- 第 7～15 行表示该类成员变量的声明。
- 第 16～25 行表示该类的构造器，主要是对成员变量的初始化。
- 第 26～44 行表示重写的 onDraw()方法，在其中主要是设置画笔颜色，并绘制图形。
- 第 52～64 行表示创建控制刷帧的方法。

上述代码介绍了本程序的绘制类的开发，接下来介绍的是屏幕触控事件的方法，将下列代码插入到 MySurfaceView 框架的第 45 行。

代码位置：见随书光盘中源代码/第 12 章/Sample12_4/src/com/bn/ex12d 目录下的 MySurfaceView。

```
1   public boolean onTouchEvent(MotionEvent e){
2       int action=e.getAction()&MotionEvent.ACTION_MASK;    //获取触控的动作编号
3       int id=(e.getAction()&MotionEvent.ACTION_POINTER_ID_MASK)>>>
4               MotionEvent.ACTION_POINTER_ID_SHIFT;         //得到 id
5       switch(action){
6           case MotionEvent.ACTION_DOWN:                    //主点 down
7           case MotionEvent.ACTION_POINTER_DOWN:            //辅点 down
8               hm.put(id, new BNPoint(e.getX(id),e.getY(id)));
                                                    //在 Map 中记录一个新点
9               if(hm.size()==2){                            //Map 的长度为 2
10                  BNPoint bpTempA=hm.get(0);               //得到第一个数据
11                  BNPoint bpTempB=hm.get(1);               //得到第二个数据
12                  distance=BNPoint.calDistance(bpTempA, bpTempB);
13              }
14          break;                                           //退出
15          case MotionEvent.ACTION_MOVE:                    //主、辅点 move
```

```
            Set<Integer> ks=hm.keySet();                                    //更新位置
                for(int i:ks){                                              //增强的 for 循环
                    hm.get(i).setLocation(e.getX(i), e.getY(i));
                }
                if(hm.size()==2) {                                          //同时计算旋转角度
                    BNPoint bpTempA=hm.get(0);                              //得到第一个数据
                    BNPoint bpTempB=hm.get(1);                              //得到第二个数据
                    float currDis=BNPoint.calDistance(bpTempA, bpTempB);
                                                                            //换算缩放系数
                    currScale=currScale+(currDis-distance)/200;
                    if(currScale>2||currScale<0.5){                         //收缩范围
                    currScale=currScale-(currDis-distance)/200;
                                                                            //为 currScale 赋值
                }
                distance=currDis;                                           //触控点之间距离
                if(bpTempA.hasOld||bpTempB.hasOld){                         //计算旋转角度
                    double alphaOld= Math.atan2((bpTempA.oldY- bpTempB.oldY),
                    (bpTempA.oldX- bpTempB.oldX));
                    double alphaNew=Math.atan2((bpTempA.y-bpTempB.y), (bpTempA.x-
bpTempB.x));
                    angle=angle+(float)Math.toDegrees(alphaNew-alphaOld);
                                                                            //增加 angle 的值
                }
                this.repaint();                                             //重绘画面
                }
            break;                                                          //退出
            case MotionEvent.ACTION_UP:                                     //主点 up
                hm.clear();                                                 //清除集合
            break;
            case MotionEvent.ACTION_POINTER_UP:                             //辅点 up
                hm.remove(id);                                            //Map 中删除对应 id 的辅点
            break;
        }
        return true;                                                        //返回 true
    }
```

提示：上述方法为监听触控方法，此方法中通过用户单击屏幕事件，计算触控点距离并换算为缩放系数，同时自定义图片的收缩比例范围。

上述代码介绍了本软件的控制类与绘制类的开发，接下来介绍的是根据触控点计算触控点距离的 BNPoint.java 类的开发，代码如下。

代码位置：见随书光盘中源代码/第 12 章/Sample12_4/src/com/bn/ex12d 目录下的 BNPoint.java。

```
package com.bn.ex12d;                                                   //声明包
public class BNPoint {
    float x;                                                            //触控的 X 坐标
    float y;                                                            //触控的 Y 坐标
    float oldX;                                                         //上一次触控的 X 坐标
    float oldY;                                                         //上一次触控的 Y 坐标
    boolean hasOld=false;                                               //是否存在上次触控位置
    public BNPoint(float x,float y) {
        this.x=x;                                                       //触控的 X 坐标
        this.y=y;                                                       //触控的 Y 坐标
    }
    public static float calDistance(BNPoint a,BNPoint b){//计算两个触控点的距离
        float result=0;
```

```
14            result=(float)Math.sqrt(                    //计算两点之间的距离
15            (a.x-b.x)*(a.x-b.x)+
16            (a.y-b.y)*(a.y-b.y)
17            );
18        return result;                                  //返回距离值
19        }
20        public void setLocation(float x,float y){       //记录上次触控与新的触控位置
21            oldX=this.x;                                //把原来位置记录为旧位置
22            oldY=this.y;
23            hasOld=true;                                //设置标志位
24            this.x=x;                                   //设置新位置
25            this.y=y;
26    } }
```

提示：此类库接收传递的参数，并记录上次触控点的位置和新触控点的位置，然后通过两个触控点的 X，Y 坐标计算两点之间的距离，返回结果。

实例 5　传感器探测者

由 Google 推出的 Android 的平台，其最吸引开发者的莫过于传感器。使用传感器的手机游戏，其操作性大大的增强，但是如何确定手机中传感器的类型成为一大难题。在本节将要介绍如何确定手机中传感器的数量以及每个传感器的具体参数。

【实例描述】

运行本程序进入主界面，在主界面单击“查看传感器类型”按钮，即可以在 EditText 文本框中显示手机中各个传感器的具体参数。

本实例的运行效果为如图 12-13、图 12-14 所示。

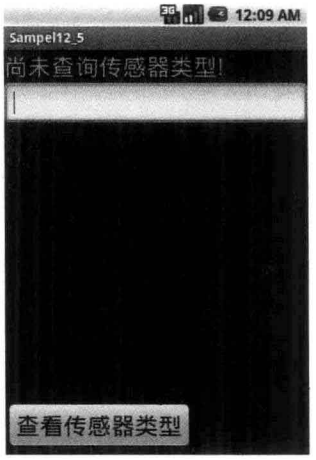


图 12-13　开始界面

图 12-14　查询结果

提示：本程序的运行效果为图 12-13、图 12-14，其中图 12-13 表示程序开始时进入的界面，图 12-14 表示单击“查看传感器类型”按钮查询得到传感器的具体参数。

【实现过程】

在本程序的开发过程中，主要用到 SensorManager 对象，通过其调用 getSensorList 方法，将得到的数据存入 List<Sensor>。创建 StringBuffer 对象，通过调用 getType、getName、getPower 与 getResolution 等方法得到传感器的数据，并将数据添加进 StringBuffer 对象中。

【代码解析】

在本部分会详细介绍本程序的实现过程，首先介绍本程序中 main.xml 文件的设置，代码如下。

代码位置：见随书光盘中源代码/第 12 章/Sample12_5/ res/layout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>          <!--版本号和编码方式-->
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3                 android:orientation="vertical"
4                 android:layout_width="fill_parent"
5                 android:layout_height="fill_parent"
6       >                                           <!--LinearLayout 的属性设置-->
7           <TextView  android:id="@+id/TextView01"
8                   android:text="尚未查询传感器类型!"
9                   android:textSize="25dip"
10                  android:layout_width="fill_parent"
11                  android:layout_height="wrap_content"
12          />                                      <!-- TextView 的属性设置-->
13      <ScrollView android:id="@+id/ScrollView01"
14          android:layout_width="fill_parent"
15          android:layout_height="340dip"
16      >                                           <!-- ScrollView 的属性设置-->
17          <EditText android:text=""
18                  android:id="@+id/EditText01"
19                  android:editable="false"
20                  android:layout_width="fill_parent"
21                  android:layout_height="wrap_content">
                                                    <!-- EditText 的大小-->
22          </EditText>                             <!-- EditText 的属性设置-->
23      </ScrollView>
24      <Button android:text="查看传感器类型"
25                  android:id="@+id/Button01"
26                  android:textSize="25dip"
27                  android:gravity="bottom|left"
28                  android:layout_width="wrap_content"
29                  android:layout_height="wrap_content">  <!--Button 的大小-->
30      </Button>                                   <!-- Button 的属性设置-->
31  </LinearLayout>
```

其中：

- 第 2～6 行表示设置总的 LinearLayout 的属性，其中设置的排放顺序为竖直排列。
- 第 7～12 行表示在 LinearLayout 中添加一个 TextView 控件，并设置其详细的属性。
- 第 13～23 行表示添加的 ScrollView 控件，设置其属性，并在其中再添加一个 EditText 控件。
- 第 24～30 行表示在 LinearLayout 中添加一个 Button 控件，并设置其详细的属性。

上述 main.xml 代码主要是对本程序主界面的搭建。接下来介绍的是如何查看手机中各个传感器的具体参数，代码如下。

代码位置：见本书随书光盘中源代码/第 12 章/Sample12_5/src/com/bn/ex12e 目录下的

Sample12_5_Activity。

```
1   package com.bn.ex12e;                                          //声明包
2   import java.util.List;                                         //导入相关类
3   ......//该处省略了部分类的导入代码，读者可自行查阅随书光盘源代码
4   import android.widget.TextView;                                //导入相关类
5   public class Sampel12_5_Activity extends Activity{             //创建继承 Activity 的类
6      @Override
7      public void onCreate(Bundle savedInstanceState){ //需要重写的方法
8         super.onCreate(savedInstanceState);                     //调用父类
9         setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
10        setContentView(R.layout.main);                          //跳转到主界面
11        Button button=(Button) findViewById(R.id.Button01);    //Button 引用
12        button.setOnClickListener(                              //添加监听器
13           new OnClickListener(){                               //匿名内部类
14                  @Override
15                  public void onClick(View v){                 //重写的方法
16                      TextView tv=(TextView)findViewById(R.id.TextView01);
17                      EditText et=(EditText)findViewById(R.id.EditText01);
18                      tv.setText("查询传感器的参数如下：");  //设置显示文字
19                      StringBuffer str=getSource();  //调用 getSource()方法
20                      et.setText(str);                        //显示文字
21              }
22        });
23     }
24     public StringBuffer getSource(){                           //声明 getSource()方法
25      SensorManager
tempSensorMsg=(SensorManager)getSystemService(SENSOR_SERVICE);
26      List<Sensor> lsmsg=tempSensorMsg.getSensorList(Sensor.TYPE_ALL);
                                                                   //得到的数据并存储
27      StringBuffer str=new StringBuffer();                       //创建 StringBuffer 对象
28      str.append("传感器的总数为：");                            //添加进 str 中
29      str.append(lsmsg.size());                                  //添加进 str 中
30      str.append("\n*****************************\n");    //添加进 str 中
31      for(Sensor s:lsmsg){                                       //循环遍历 lsmsg
32          str.append("传感器编号:");
33          str.append(s.getType()+"\n");                          //得到编号
34          str.append("传感器名称:");
35          str.append(s.getName()+"\n");                          //得到名称
36          str.append("工作电流:");                               //添加进 str 中
37          str.append(s.getPower()+"\n");                         //得到工作电流
38          str.append("分辨率:");                                 //添加进 str 中
39          str.append(s.getResolution()+"\n");                    //得到分辨率
40          str.append("最大测量值:");
41          str.append(s.getMaximumRange()+"\n");                  //得到最大测量值
42          str.append("供应商:");
43          str.append(s.getVendor()+"\n");                        //得到供应商
44          str.append("传感器的版本号:");                         //添加进 str 中
45          str.append(s.getVersion());                            //得到版本号
46      }
47      str.append("\n*****************************\n");          //添加进 str 中
48      return str;                                                //返回数据
49     }
50  }
```

其中：

- 第 12～20 行表示为 Button 按钮添加监听器，单击该按钮设置 TextView 的显示，调用

getSource()方法得到数据，并显示在 EditText 中。

- 第 24～49 行表示得到传感器的数据的方法，将得到的数据存放进 List<Sensor>集合中，创建 StringBuffer 对象，遍历 List<Sensor>将具体数据存入 StringBuffer 对象中，并返回得到的数据。

实例 6　小球游戏动态壁纸

伴随着 3G 网络发展，智能手机逐渐被手机使用者接纳，其逐渐成为手机购买者的首要选择。壁纸作为智能手机的一部分，很受手机使用者的喜爱。本节将要介绍如何开发手机的动态壁纸。

【实例描述】

本程序为手机壁纸，运行本程序。长按屏幕，弹出“Add to Home screen”对话框，在对话框中单击“Wallpapers”，弹出“Select wallpaper from”列表框，单击“live wallpapers”对话框，选中需要设置的桌面背景，即可实现动态壁纸。

本实例的运行效果为如图 12-15、图 12-16、图 12-17 所示。

图 12-15　“Add to Home screen”对话框

图 12-16　“Select wallpaper from”列表框

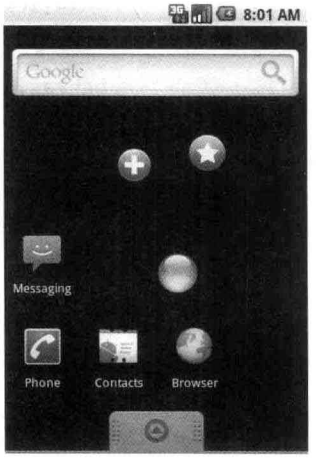


图 12-17　动态壁纸界面

提示：本程序的运行效果为图 12-15、图 12-16、图 12-17，其中图 12-15 表示长按屏幕弹出的“Add to Home screen”对话框，图 12-16 表示单击“Wallpapers”弹出“Select wallpaper from”列表框，图 12-17 表示设置小球完成动态壁纸界面。

【实现过程】

在本程序的开发过程中，需要继承 WallpaperService 类，并重写相应的方法。需要开启一个 BallGoThread 的线程类，定时更新球的位置。同时需要创建单个球的控制类 SingleBall，在其中绘制球，同时判断球的运动是否碰壁。创建一个管理所有球的类 AllBalls，随机地生成球的位置。

【代码解析】

在本部分详细介绍本程序具体的实现过程，首先介绍的是本程序中单个球的控制类，代码如下。

代码位置：见本书随书光盘中源代码/第 12 章/Sample12_6/src/com/bn/ex12f 目录下的 SingleBall。

```
1   package com.bn.ex12f;                                    //声明包
2   import android.graphics.Bitmap;                          //导入相关类
3   import android.graphics.Canvas;                          //导入相关类
4   import android.graphics.Paint;                           //导入相关类
5   public class SingleBall{
6   ......//该处省略成员变量的声明，读者可自行查阅随书光盘源代码
7       public SingleBall(int x,int y,int size,int direction,Bitmap bitmap,int
xSpan,int ySpan){
8           this.x=x;                                        //初始化坐标位置
9           this.y=y;
10          this.size=size;                                  //初始化球的尺寸
11          this.bitmap=bitmap;                              //初始化球的位图
12          this.direction=direction;                        //球的运动方向
13          this.xSpan=xSpan;                                //初始化 x 方向的步进
14          this.ySpan=ySpan;                                //初始化 y 方向的步进
15      }
16      void drawSelf(Canvas canvas, Paint paint){           //绘制单个球的方法
17          canvas.drawBitmap(bitmap, x,y, paint);           //调用 drawBitmap 方法
18      }
19      void go(){                                           //单个球运动的方法
20          int tempX,tempY;                                 //球的目标位置
21          switch(direction){
22              case DIRECTION_YS:                           //如果在向右上方运动
23                  tempX=x+xSpan;                           //计算目标位置坐标
24                  tempY=y-ySpan;
25                  if(isCollideWithRight(tempX,tempY)){     //到达屏幕右侧
26                      direction=DIRECTION_ZS;              //改变运动方向为左上
27                  }else if(isCollideWithUp(tempX,tempY)){      //到达屏幕上侧
28                      direction=DIRECTION_YX;              //改变运动方向为右下
29                  }else{                                   //如果没有碰撞
30                      x=tempX;                             //更新坐标位置
31                      y=tempY;
32                  }
33              break;
34              case DIRECTION_ZS:                           //如果在向左上方运动
35                  tempX=x-xSpan;                           //计算目标位置坐标
36                  tempY=y-ySpan;
37                  if(isCollideWithLeft(tempX,tempY)){      //到达屏幕左侧
38                      direction=DIRECTION_YS;              //改变运动方向为右上
39                  }else if(isCollideWithUp(tempX,tempY)){      //到达屏幕上侧
40                      direction=DIRECTION_ZX;              //改变运动方向为左下
41                  }else{                                   //如果没有碰撞
42                      x=tempX;                             //更新坐标位置
43                      y=tempY;
44                  }
45              break;
46              case DIRECTION_ZX:                           //如果在向左下方运动
47                  tempX=x-xSpan;                           //计算目标位置坐标
48                  tempY=y+ySpan;
```

```
49                  if(isCollideWithLeft(tempX,tempY)){      //到达屏幕左侧
50                      direction=DIRECTION_YX;              //改变运动方向为右下
51                  }else if(isCollideWithDown(tempX,tempY)){   //到达屏幕下侧
52                      direction=DIRECTION_ZS;              //改变运动方向为左上
53                  }else{                                   //如果没有碰撞
54                      x=tempX;                             //更新坐标位置
55                      y=tempY;
56                  }
57              break;
58              case DIRECTION_YX:                           //如果在向右下方运动
59                  tempX=x+xSpan;                           //计算目标位置坐标
60                  tempY=y+ySpan;
61                  if(isCollideWithRight(tempX,tempY)){     //到达屏幕右侧
62                      direction=DIRECTION_ZX;              //改变运动方向为左下
63                  }else if(isCollideWithDown(tempX,tempY)){   //到达屏幕下侧
64                      direction=DIRECTION_YS;              //改变运动方向为右上
65                  }else{                                   //如果没有碰撞
66                      x=tempX;                             //更新坐标位置
67                      y=tempY;
68                  }
69              break;                                       //退出
70              }
71      }
72      boolean isCollideWithRight(int tempX,int tempY){     //判断是否与屏右侧碰撞
73          return !(tempX>0&&tempX<Constant.SCREEN_WIDTH-this.size);
74      }
75      boolean isCollideWithUp(int tempX,int tempY){        //判断是否与屏上侧碰撞
76          return !(tempY>0);                               //返回值
77      }
78      boolean isCollideWithLeft(int tempX,int tempY){      //判断是否与屏左侧碰撞
79          return !(tempX>0);                               //返回值
80      }
81      boolean isCollideWithDown(int tempX,int tempY){      //判断是否与屏下侧碰撞
82          return !(tempY>0&&tempY<Constant.SCREEN_HEIGHT-this.size);
83  }   }
```

其中：

- 第 7～15 行表示该类的构造器，在其中为该类的成员变量赋值。
- 第 16～18 行表示绘制单个球的方法。
- 第 19～71 行表示判断球的方向，同时为每个对应方向的坐标赋值，并判断是否发生碰撞，如果没有发生碰撞，则小球继续运动。
- 第 72～74 行表示判断小球是否与屏幕右侧发生碰撞。
- 第 75～77 行表示判断小球是否与屏幕上侧发生碰撞。
- 第 78～80 行表示判断小球是否与屏幕左侧发生碰撞。
- 第 81～83 行表示判断小球是否与屏幕下侧发生碰撞。

上述代码已经介绍了单个球的控制类，接下来介绍的是管理所有小球的类的开发，代码如下。

代码位置：见本书随书光盘中源代码/第 12 章/Sample12_6/src/com/bn/ex12f 目录下的 AllBalls。

```
1   package com.bn.ex12f;                                    //声明包
2   import java.util.ArrayList;                              //导入相关类
3   ......// 该处省略了部分类的导入代码，读者可自行查阅随书光盘源代码
4   import android.graphics.Paint;                           //导入相关类
5   public class AllBalls{
```

```
6       ArrayList<SingleBall> alSingleBall=new ArrayList<SingleBall>();
                                                                    //单个球列表
7       Bitmap[] ballsBitmap;                                       //位图数组
8       int[] ballsSize;                                            //球尺寸数组
9       int[] ballsXSpan;                                           //球 x 方向步进数组
10      int[] ballsYSpan;                                           //球 y 方向步进数组
11      public AllBalls(int[] ballsSize,Bitmap[] ballsBitmap,int[] ballsXSpan,int[]
ballsYSpan){
12          this.ballsSize=ballsSize;                               //成员变量赋值
13          this.ballsBitmap=ballsBitmap;                           //成员变量赋值
14          this.ballsXSpan=ballsXSpan;                             //成员变量赋值
15          this.ballsYSpan=ballsYSpan;                             //成员变量赋值
16          for(int i=0;i<ballsSize.length;i++){                    //循环球尺寸数组
17          int x=(int) (Math.random()*(Constant.SCREEN_WIDTH-ballsSize[i]));
                                                                    //单个球位置
18          int y=(int) (Math.random()*(Constant.SCREEN_HEIGHT-ballsSize[i]));
                                                                    //单个球位置
19          int direction=(int) Math.random()*4;                    //的运动方向
20          alSingleBall.add(                                       //加入列表
21              new SingleBall(x,y,ballsSize[i],direction,ballsBitmap[i],
ballsXSpan[i],ballsYSpan[i]));
22      }   }
23      public void drawSelf(Canvas canvas, Paint paint){   //绘制所有球的方法
24          for(SingleBall sb:alSingleBall){                        //循环单个球列表
25              sb.drawSelf(canvas, paint);                         //绘制单个球
26      }   }
27      public void go(){                                           //使所有球运动的方法
28          for(SingleBall sb:alSingleBall){                        //循环单个球列表
29              sb.go();                                            //使单个球运动
30  }   }   }
```

其中：

- 第 6～10 行表示对该类的成员变量的声明。
- 第 11～22 行表示该类的构造器，在该构造器中需要对该类的成员变量赋初值，并且随机生成球的位置和运动方向，创建该球的对象并存入到 ArrayList 中。
- 第 23～26 行表示绘制所有球的方法，遍历球的列表，并调用 SingleBall 类的 drawSelf() 方法。
- 第 27～30 行表示所有球运动的方法，遍历球的列表，并调用 SingleBall 类的 go() 方法。

上述代码已经介绍了管理所有小球的类，接下来介绍的是控制小球运动的线程类的开发，代码如下。

代码位置：见本书随书光盘中源代码/第 12 章/Sample12_6/src/com/bn/ex12f 目录下的 BallGoThread。

```
1   package com.bn.ex12f;                                   //声明包
2   public class BallGoThread extends Thread{               //创建继承 Thread 的类
3       AllBalls allBalls;                                  //声明 AllBalls 的引用
4       public BallGoThread(AllBalls allBalls){             //构造器
5           this.allBalls=allBalls;                         //成员变量赋值
6       }
7       boolean ballGoFlag=true;                            //循环标志位
8       @Override
9       public void run(){                                  //重写 run 方法
10          while(ballGoFlag){                              //while 循环
```

```
                allBalls.go();                                  //调用使所有球运动的方法
                try{
                    Thread.sleep(Constant.MOVE_TIME);           //一段时间后再运动
                }catch(Exception e){                            //打印堆栈信息
                    e.printStackTrace();                        //打印异常
                }
}    }    }
```

提示：该类为继承 Thread 的线程类，其主要作用是定时调用 AllBalls 类的 go()方法。

上述代码已经介绍了定时更新界面的线程类，接下来介绍的是本程序的主控制类的开发，代码如下。

代码位置：见本书随书光盘中源代码/第 12 章/Sample12_6/src/com/bn/ex12f 目录下的 Sample12_6_WallPaper。

```
1   package com.bn.ex12f;                                       //声明包
2   import android.graphics.Bitmap;                             //导入相关类
3   ……//该处省略了部分类的导入代码，读者可自行查阅随书光盘源代码
4   import android.view.SurfaceHolder;                          //导入相关类
5   public class Sample12_6_WallPaper extends WallpaperService{  //创建类
6       ......//该处省略了该类的部分成员变量的声明，读者可自行查阅随书光盘源代码
7       @Override
8      public Engine onCreateEngine(){                          //重写 onCreateEngine 方法
9          ce=new BallEngine();                                 //创建 BallEngine 对象
10        return ce;                                            //返回创建的对象
11     }
12      public void initBitmap(){                               //初始化图片资源的方法
13          ......//该处省略了初始化图片的方法，读者可自行查阅随书光盘源代码
14     }
15   class BallEngine extends Engine{                           //创建内部类
16       Sample12_6_WallPaper father;
//MovingBallWallPaper 的引用
17      private final Paint paint = new Paint();                //创建画笔
18      boolean ifDraw;                                         //是否可见的标志位
19      BallGoThread bgThread;                                  //BallGoThread 引用
20      AllBalls allBalls;                                      // AllBalls 对象的引用
21      private final Runnable mDrawCube = new Runnable(){      //匿名内部类
22         public void run() {                                  //重写 run 方法
23            drawBalls();                                      //调用 drawBalls 方法
24         }
25      };
26      @Override
27      public void onCreate(SurfaceHolder surfaceHolder){      //重写 onCreate 方法
28         super.onCreate(surfaceHolder);                       //调用父类对应方法
29           paint.setAntiAlias(true);                          //打开抗锯齿
30           initBitmap();                                      //初始化位图资源
31       }
32      @Override
33      public void onDestroy(){                                //重写 onDestroy 方法
34         super.onDestroy();                                   //调用父类对应方法
35      }
36      @Override
37      public void onVisibilityChanged(boolean visible){
                                                  //重写 onVisibilityChanged 方法
38       ifDraw=visible;                                        //获得是否可见标志位
```

```
39          if(ifDraw){                                          //如果可见
40              bgThread=new BallGoThread(allBalls);             //创建 BallGoThread 线程
41              bgThread.start();                                //启动该线程
42              hd.postDelayed(mDrawCube, Constant.MOVE_TIME); //一定时间后绘制球
43            }else{                                             //如果不可见
44              bgThread.ballGoFlag=false;                       //停止 BallGoThread 线程
45        }}
46        @Override                                          //重写 onSurfaceChanged 方法
47        public void onSurfaceChanged(SurfaceHolder holder, int format, int width,
int height){
48            super.onSurfaceChanged(holder, format, width, height); //调用父类对应方法
49            if(width>height){                                //如果是横屏
50              Constant.SCREEN_HEIGHT=ConstantHP.SCREEN_HEIGHT;    //初始化横屏常量
51              Constant.SCREEN_WIDTH=ConstantHP.SCREEN_WIDTH;
52            }else{                                           //如果是竖屏
53              Constant.SCREEN_HEIGHT=ConstantSP.SCREEN_HEIGHT;    //初始化竖屏常量
54              Constant.SCREEN_WIDTH=ConstantSP.SCREEN_WIDTH;
55            }
56            int[] ballsSize={Constant.YELLOW_BALL_SIZE,  //所有球尺寸数组
57                             Constant.BLUE_BALL_SIZE,Constant.GREEN_BALL_SIZE};
58              Bitmap[] ballsBitmap={yellowBallBitmap,blueBallBitmap,
greenBallBitmap}; //所有球位图数组
59              int[] ballsXspan={Constant.YELLOW_XSPAN,        //所有球的 xSpan 数组
60                                Constant.BLUE_XSPAN,Constant.GREEN_XSPAN};
61              int[] ballsYspan={Constant.YELLOW_YSPAN,        //所有球的 ySpan 数组
62                             Constant.BLUE_YSPAN,Constant.GREEN_YSPAN};
63             allBalls=new AllBalls(ballsSize,ballsBitmap,ballsXspan,ballsYspan);
                                                               //创建 AllBalls 对象
64        }
65        @Override
66        public void onSurfaceCreated(SurfaceHolder holder){
                                                           //重写 onSurfaceCreated 方法
67            super.onSurfaceCreated(holder);                  //调用父类对应方法
68        }
69        @Override
70        public void onSurfaceDestroyed(SurfaceHolder holder){
                                                           //重写 onSurfaceDestroyed 方法
71            super.onSurfaceDestroyed(holder);            //调用父类对应方法
72        }
73        void drawBalls(){                                  //绘制所有球的方法
74            final SurfaceHolder holder = getSurfaceHolder();
                                                           //获得 SurfaceHolder 对象
75            Canvas canvas = null;                        //声明画布引用
76            try {
77              canvas = holder.lockCanvas();              //锁定并获得画布对象
78               if (canvas != null){                      //如果已得到画布对象
79                  canvas.drawColor(Color.argb(255, 0, 0, 0));     //擦空界面
80                  allBalls.drawSelf(canvas, paint);                //绘制所有的球
81            } } finally {
82             if (canvas != null) holder.unlockCanvasAndPost(canvas); //绘制完释放画布
83            }
84            if(ifDraw){                                          //如果桌面可见
85              hd.postDelayed(mDrawCube, Constant.MOVE_TIME); //一定时间后绘制球
86            }
87    }}}
```

其中：

- 第 8～11 行表示重写 onCreateEngine 方法，并返回创建的 BallEngine 对象。

- 第 12～14 行表示初始化位图资源的方法。
- 第 21～25 行表示基于 Runnable 接口的匿名内部类，并重写 run()方法。
- 第 27～31 行表示重写 onCreate 方法，打开抗锯齿，并初始化位图资源。
- 第 36～45 行表示重写 onVisibilityChanged 方法，主要是创建 BallGoThread 线程并启动线程。
- 第 46～64 行表示重写 onSurfaceChanged 方法，主要是判断手机当前的屏幕为横屏还是竖屏，初始化相应数据，同时创建 AllBalls 对象。
- 第 73～87 行表示创建绘制球的方法，主要是获得 SurfaceHolder 对象，同时锁定并获得画布对象。最后需要判断 canvas 是否为空，如果不为空则需要释放画布。

实例 7　自动完成输入框

本实例实现的是城市名称输入时的自动完成功能，用户需要输入期望城市名称的汉语拼音首字母，系统会根据其内容自动列出可能的城市，并以列表的形式显示出来。

【实例描述】

运行本软件显示 AutoCompleteTextView 控件提示用户输入城市名称的汉语拼音首字母，用户输入城市名称的首字母后，系统将自动检测出与此首字母相匹配的城市名称内容，并且以下拉列表的形式显示出来，若匹配的城市名称很多，下拉列表还可以用手指拖动方便选择。

本实例的运行效果为如图 12-18、图 12-19、图 12-20 所示。

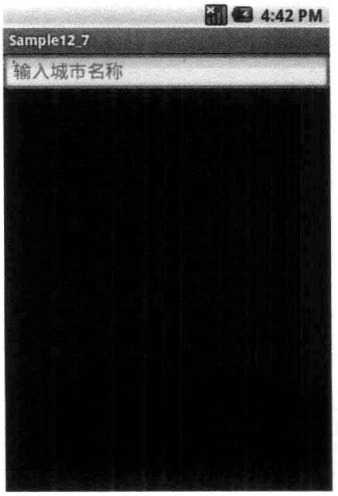


图 12-18　主界面

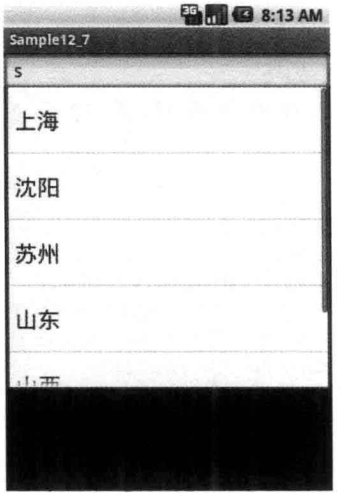


图 12-19　查询界面

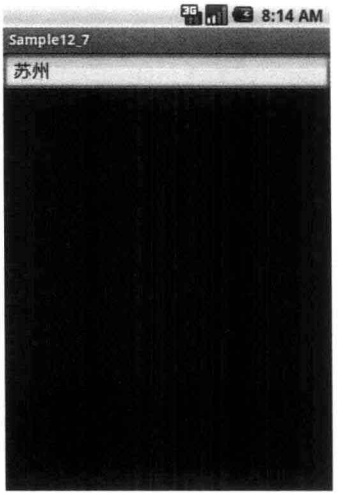


图 12-20　查询结果界面

提示：本程序的运行效果为图 12-18、图 12-19、图 12-20，其中图 12-18 为本软件的主界面，图 12-19 为本软件的查询界面，图 12-20 为本软件查询结果界面。

【实现过程】

在本程序的开发过程中，用了 AutoCompleteTextView 动态匹配内容控件，首先创建适配器，

增加城市名称内容到适配器，通过用户查询以列表形式显示出相应的城市名称，单击某一城市名称，触动监控事件，在 AutoCompleteTextView 控件中动态地显示出内容。

【代码解析】

首先介绍的是本软件的布局文件 main.xml，代码如下。

代码位置：见随书光盘中源代码/第 12 章/Sample12_7/res/layout 目录下的 main.xml。

```
<?xml version="1.0" encoding="utf-8"?> <!--版本号  编码方式-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
                android:orientation="vertical"
                android:layout_width="fill_parent"
                android:layout_height="fill_parent"
    >                                    <!--LinearLayout 的属性设置-->
    <AutoCompleteTextView
                android:id="@+id/AutoCompleteTextView01"
                android:layout_width="fill_parent"
                android:singleLine="true"
                android:layout_height="40dip"
                android:completionThreshold="1"
                android:hint="输入城市名称"
                android:text=""
    ></AutoCompleteTextView>          <!-- AutoCompleteTextView 的属性设置-->
</LinearLayout>
```

提示：上述为本软件主界面布局 Xml 配置文件，其中 AutoCompleteTextView 为动态匹配内容控件。

上述 main.xml 代码已经介绍了该类的主界面的开发，接下来介绍本软件的主控制类 Sample12_7_Activity 的开发，代码如下。

代码位置：见本书随书光盘中源代码/第 12 章/Sample12_7src/com/bn/ex12g 下的 Sample12_7_Activity.java。

```
package com.bn.ex12h;                                          //包的声明
import android.app.Activity;                                   //相关类的引入
import android.os.Bundle;                                      //相关类的引入
import android.widget.AutoCompleteTextView;                    //相关类的引入
public class Sample12_7_Activity extends Activity {
    public void onCreate(Bundle savedInstanceState) {          //重写的方法
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);                             //切换主界面
    String s1[]=new String[]{"北京","天津","廊坊","保定","成都","贵阳","遵义",
                                                               //为其中准备初始数据
            "上海","重庆","昆明","沈阳","苏州","杭州",
            "天津","唐山","吉林","湖南","娄底","湖北",
            "武汉","山东","山西","广东","广西","常州",
            "内蒙古","京华","秦皇岛","沧州","承德","石家庄",
            "衡水","德州","青岛","济南","长沙"} ;
    String s2[]=new String[]{"bj","tj","lf","bd","cd","gy","zy",
                                                               //简称数据数组
            "sh","cq","km","sy","sz","hz",
            "tj","ts","jl","hn","ld","hb",
            "wh","sd","sx","gd","gx","cz",
            "nmg","jh","qhd","cz","cd","sjz",
            "hs","dz","qd","jn","cs"};
    CityAdapter<String> cAdapter = new CityAdapter<String>(this,
```

```
    22                                        android.R.layout.simple_dropdown_item_
1line,
    23                                        s1,s2);              //设置提示框中的内容
    24        AutoCompleteTextView autoView=(AutoCompleteTextView)findViewById(
    25                                       R.id.AutoCompleteTextView01);
                                                                   //设置提示信息的输入框
    26        autoView.setAdapter(cAdapter);                       //添加适配器
    27        autoView.setThreshold(1);                            //设置临界值
    28        autoView.setDropDownHeight(300) ;                    //设置其
    29    } }
```

提示：上述代码中为主控制类首先设置主界面布局，然后配置适配器内容。

上述代码介绍的是本软件的主控制类 Sample12_7_Activity 的开发，接下来介绍的是适配器类 CityAdapter 的框架的开发，代码如下。

代码位置：见本书随书光盘中源代码/第 12 章/Sample12_7src/com/bn/ex12g 下的 CityAdapter。

```
    1    package com.bn.ex12h;                                   //包的声明
    2    import java.util.ArrayList;                             //相关类的引入
    3    ......//此处省略了部分类的引入代码，请读者自行查看随书光盘源代码
    4    import java.util.Arrays;                                //相关类的引入
    5    public class CityAdapter<T> extends BaseAdapter
    6    implements Filterable{                                  //创建类
    7        ......//此处省略了部分变量的声明，请读者自行查看随书光盘源代码
    8        public CityAdapter(Context context, int textViewResourceId,  //构造器
    9        T[] objects,T[] objects2) {
    10            init(context, textViewResourceId, 0, Arrays.asList(objects),
                                                                   //调用 init 方法
    11            Arrays.asList(objects2));                       //调用 init 方法
    12       }
    13       private void init(Context context, int resource,     //创建 init 方法
    14       int textViewResourceId, List<T> objects ,List<T> objects2) {
                                                                   //声明 init 方法
    15           mContext = context;                             //为 mContext 赋值
    16           mInflater=(LayoutInflater)context.getSystemService(Context.LAYOUT_
INFLATER_SERVICE);
    17           mResource = mDropDownResource = resource;       //为 mResource 赋值
    18           mObjects = objects;                             //为 mObjects 赋值
    19           mObjects2 = objects2;                           //为 mObjects2 赋值
    20           mFieldId = textViewResourceId;                  //为 mFieldld 赋值
    21       }
    22       public int getCount() {                             //声明方法
    23           return mObjects.size();                         //返回城市列表的大小
    24       }
    25       public T getItem(int position){                     //声明方法
    26           return mObjects.get(position);                  //返回城市列表
    27       }
    28       public int getPosition(T item){                     //声明方法
    29           return mObjects.indexOf(item);                 //列表中指定字符串值索引
    30       }
    31       public long getItemId(int position){                //声明方法
    32           return position;                               //将 int 型整数以长整型返回
    33       }
    34       public View getView(int position,
    35           View convertView, ViewGroup parent){            //创建 View
```

```
36          return createViewFromResource(position, convertView, parent, mResource);
37      }
38      private View createViewFromResource(int position,
39      View convertView, ViewGroup parent,int resource){ //私有的创建 View 的方法
40          View view;                                          //局部变量
41          TextView text;                                      //局部变量
42          if (convertView == null) {                          //如果当前为空
43              view = mInflater.inflate(resource, parent, false);
44          }
45          else {                                              //如果不为空
46               view = convertView;                          //view 值为 convertView
47          }
48          try {
49              if (mFieldId == 0) {                            //如果当前域为空
50                  text = (TextView) view;
51              }
52              else{                                           //当前区域不为空
53                  text = (TextView) view.findViewById(mFieldId); //为 text 赋值
54              }
55          }catch (ClassCastException e) {                     //异常处理
56               throw new IllegalStateException (              //抛出异常
57               "ArrayAdapter requires the resource ID to be a TextView", e);
                                                                //异常对应信息
58          }
59          text.setText(getItem(position).toString());//设置 TextView 显示的值
60          return view;
61          }
62      public Filter getFilter(){                              //得到过滤器
63          if (mFilter == null){                               //如果为空
64              mFilter = new ArrayFilter();                    //创建对象
65          }
66          return mFilter;                                     //返回创建的对象
67      }
68      ......//此处省略了 ArrayFilter 内部类，将在下面的步骤给出
69  }
```

其中：

- 第 8～12 行表示该类的构造器，主要是调用 init 方法。
- 第 13～21 行表示创建的 init 方法，其主要作用是初始化对应的成员变量。
- 第 38～67 行表示创建私有的方法，主要是创建 View 并返回。

上述代码介绍 CityAdapter 的框架，接下来介绍 ArrayFilter 内部类，将下列代码插入到 CityAdapter<T>框架的第 68 行。

代码位置：见本书随书光盘中源代码/第 12 章/Sample12_7src/com/bn/ex12g 下的 CityAdapter。

```
1   private class ArrayFilter extends Filter {              //创建私有类
2       protected FilterResults performFiltering(CharSequence prefix){
                                                            //执行过滤
3           FilterResults results = new FilterResults();//创建 FilterResults 对象
4           if (mOriginalValues == null) {                  //如果为空
5               synchronized (mLock) {
6               mOriginalValues = new ArrayList<T>(mObjects);
                                                            //创建 ArrayList<T>的对象
7           } }
8           if (prefix == null || prefix.length() == 0){    //判断是否符合条件
9               synchronized (mLock) {
10              ArrayList<T> list = new ArrayList<T>(mOriginalValues);
```

```
                                                    //创建 ArrayList<T>的对象
    11              results.values = list;
    12              results.count = list.size();            //集合的大小
    13          } }
    14          else {
    15              String prefixString = prefix.toString().toLowerCase();
                                                    //转换成小写
    16              final ArrayList<T> values = mOriginalValues;    //得到 value 对象
    17              final int count = values.size();                //得到长度
    18              final ArrayList<T> newValues = new ArrayList<T>(count);
                                                  //创建 ArrayList<T>的对象
    19              for (int i = 0; i < count; i++) {           //循环遍历
    20                  final T value = values.get(i);          //得到的值赋值给 value
    21                  final String valueText = value.toString().toLowerCase();
                                                    //value 转换为字符串
    22                  final T value2 = mObjects2.get(i);
    23                  final String valueText2 = value2.toString().toLowerCase();
                                                    //values2 转换为字符串
    24                  if(valueText2.startsWith(prefixString)){
                                                    //查找拼音
    25                      newValues.add(value);       //value 添加到 newValues 中
    26                  }
    27                  else if(valueText.startsWith(prefixString)) {   //查找汉字
    28                      newValues.add(value);       //value 添加到 newValues 中
    29                  }
    30                  else{                                   //添加汉字关联
    31                      final String[] words = valueText.split(" ");
                                                  //得到字符串数组
    32                      final int wordCount = words.length;     //得到数组长度
    33                          for (int k = 0; k < wordCount; k++){    //循环遍历
    34                              if (words[k].startsWith(prefixString)) {
                                                  //判断是否符合条件
    35                                  newValues.add(value); //value 添加到 newValues 中
    36                              break;                  //退出
    37                      } }
    38                      final String[] words2 = valueText2.split(" ");
                                                  //添加拼音关联汉字
    39                      final int wordCount2 = words2.length;   //得到数组长度
    40                      for (int k = 0; k < wordCount2; k++){   //循环遍历
    41                          if (words2[k].startsWith(prefixString)){
                                                  //判断是否符合条件
    42                              newValues.add(value); //value 添加到 newValues 中
    43                          break;                  //退出
    44                      } }
    45                  }
    46              }
    47              results.values = newValues;         //为 values 赋值
    48              results.count = newValues.size();   //为 count 赋值
    49          }
    50          return results;                         //返回 result
    51      }
    52      @SuppressWarnings("unchecked")
    53          protected void publishResults(CharSequence constraint, FilterResults
results) {
    54          mObjects = (List<T>) results.values;    //局部变量声明
    55          if (results.count > 0) {                //count 大于零
    56              notifyDataSetChanged();
    57          } else {                                //count 不大于零
```

```
58                notifyDataSetInvalidated();              //调用方法
59   } } }
```

其中：

- 第 2～51 行表示开发执行过滤的方法。
- 第 53～59 行表示创建 publishResults 方法，主要判断值是否符合条件。

实例 8　对你的图片进行简单编辑

图片编辑，这项本应该在电脑端完成的工作，在手机端同样可以完成。本节将介绍基于 Android 平台的图片剪裁特效的开发。

【实例描述】

运行本程序进入图片剪裁界面，可以看到随着椭圆的增大或者减小，可以剪裁出对应的图片。本实例的运行效果为如图 12-21、图 12-22、图 12-23 所示。

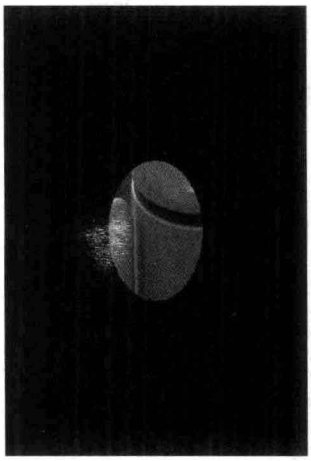


图 12-21　开始界面

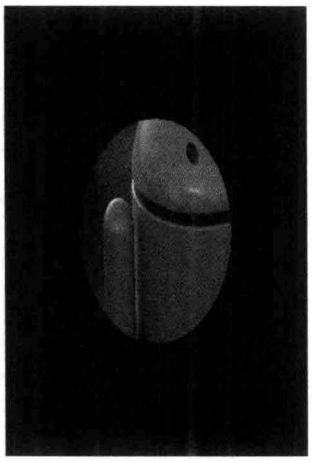


图 12-22　椭圆适中

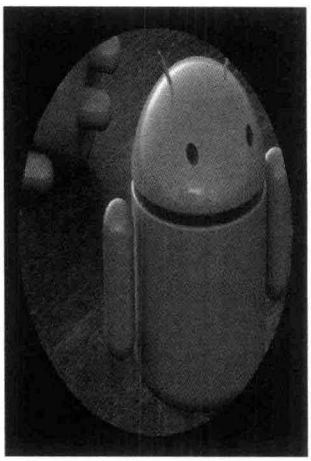


图 12-23　椭圆最大时剪裁图片的界面

提示：本程序的运行效果为图 12-21、图 12-22、图 12-23，其中图 12-21 表示程序开始时的界面，图 12-22 表示椭圆正好适中的界面，图 12-23 表示椭圆最大时剪裁图片的界面。

【实现过程】

在本程序的开发过程中，首先需要创建继承 SurfaceView 并实现 SurfaceHolder.Callback 的类。在该类中绘制需要的图片，调用 pt.addOval 方法，并绘制图片。同时需要创建线程类定时刷帧。

【代码解析】

在本部分会详细地介绍本程序具体的实现过程，首先介绍本程序的主控制类，代码如下。

代码位置：见本书随书光盘中源代码/第 12 章/Sample12_8/src/com/bn/ex12h 目录下的 Sample12_8_Activity。

```
1   package com.bn.ex12h;                                  //声明包
```

```
import android.app.Activity;                                    //导入相关类
......//该处省略了部分类的导入代码，读者可自行查阅随书光盘源代码
import android.view.WindowManager;                              //导入相关类
public class Sample12_8_Activity extends Activity{  //创建继承 Activity 的类
    @Override
    public void onCreate(Bundle savedInstanceState){ //需要重写的方法
        super.onCreate(savedInstanceState);                     //调用父类
        requestWindowFeature(Window.FEATURE_NO_TITLE);          //去除标题
        getWindow().setFlags                                    //设置全屏
          (
              WindowManager.LayoutParams.FLAG_FULLSCREEN,       //设置属性
              WindowManager.LayoutParams.FLAG_FULLSCREEN        //设置属性
          );
        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
        MySurfaceView mView=new MySurfaceView(this);  //创建 MySurfaceView 对象
        setContentView(mView);                                  //设置主界面
}   }
```

提示：该类主要是程序的主控制类，程序开始时首先执行该类中的 onCreate 方法，并设置本程序为全屏，同时强制为横屏。得到 MySurfaceView 的对象，并设定本程序的主界面。

上述代码介绍了本程序的主控制类，接下来介绍显示图片剪裁的界面，代码如下。

代码位置：见本书随书光盘中源代码/第 12 章/Sample12_8/src/com/bn/ex12h 目录下的 MySurfaceView。

```
package com.bn.ex12h;                                           //声明包
import static com.bn.ex12h.Constant.*;                          //导入相关类
......//该处省略了部分类的导入代码，读者可自行查阅随书光盘源代码
import android.view.SurfaceView;                                //导入相关类
public class MySurfaceView extends SurfaceView implements SurfaceHolder.
Callback{
    Sample12_8_Activity activity;                               //声明成员变量
    Paint paint;                                                //声明成员变量 Paint
    Bitmap tempmap;                                             //声明成员变量 Bitmap
    static int xySize=10;                                       //声明静态成员变量
    public MySurfaceView(Sample12_8_Activity activity){ //构造器
        super(activity);                                        //调用父类
        this.activity=activity;                                 //为成员变量赋值
        tempmap=BitmapFactory.decodeResource(activity.getResources(),
R.drawable.adr);
        this.getHolder().addCallback(this);                     //设置回调接口
        paint=new Paint();                                      //创建 paint 对象
        paint.setAntiAlias(true);                               //打开抗锯齿
    }
    @Override
    public void onDraw(Canvas canvas){
        Path pt=new Path();                                     //创建 Path 对象
        pt.addOval(new RectF(MAP_CENTER_XOFFSET-xySize, //绘制椭圆
                             MAP_CENTER_YOFFSET-xySize*1.5f,    //距离上侧距离
                             MAP_CENTER_XOFFSET+xySize,     //距离右侧距离
                             MAP_CENTER_YOFFSET+xySize*1.5f),   //距离下侧距离
                        Path.Direction.CCW);                    //设置绘制方式
        canvas.drawARGB(255, 0, 0, 0);                          //设定颜色
        canvas.clipPath(pt);                                    //调用 clipPath 方法
```

```
28              canvas.drawBitmap(tempmap, MAP_XOFFSET, MAP_YOFFSET, paint);
29          }
30          @Override                                                    //需要重写的方法
31          public void surfaceChanged(SurfaceHolder holder, int format, int width,int
height){ }
32          @Override
33          public void surfaceCreated(SurfaceHolder holder){           //需要重写的方法
34              ViewForDrawThread.flag=true;                             //设置 flag 为 true
35              (new ViewForDrawThread(this)).start();                   //开启线程
36              repaint();                                               //调用 repaint 方法
37          }
38          @Override
39          public void surfaceDestroyed(SurfaceHolder holder){ //销毁时调用
40              ViewForDrawThread.flag=false;                            //设定 flag 为 false
41          }
42          public void repaint(){                                       //声明 repaint 方法
43              SurfaceHolder mHolder=this.getHolder();                  //得到刷帧的控制器
44              Canvas canvas=mHolder.lockCanvas();                      //得到画笔
45              try{
46                  synchronized(mHolder){                               //是否锁定
47                      onDraw(canvas);                                  //调用 onDraw 方法
48                  }
49              }catch(Exception e){                                     //捕获异常
50                  e.printStackTrace();                                 //打印堆栈信息
51              }finally{
52                  if(canvas!=null){                                    //判断 canvas 是否为空
53                      mHolder.unlockCanvasAndPost(canvas);             //释放画笔
54                  }
55  } } }
```

其中：

- 第 10～17 行表示的是该类的构造器，主要是初为成员变量赋值、初始化图片、设置回调接口、创建画笔与打开抗锯齿。
- 第 19～29 行表示的是得到 Path 对象，绘制椭圆与图片。
- 第 33～37 行表示的方法在开始绘制主界面时执行，主要是打开刷帧线程。
- 第 39～41 行表示的方法在程序结束后调用，主要是关闭刷帧线程。
- 第 42～55 行表示的是主界面的刷帧方法。首先获得刷帧控制器，并获得画笔，最后判断 canvas 是否为空，并释放画笔。

上述代码已经介绍了本程序主界面的实现，接下来介绍的是本程序的后台控制线程类，代码如下。

代码位置：见本书随书光盘中源代码/第 12 章/Sample12_8/src/com/bn/ex12h 目录下的 ViewForDrawThread。

```
1   package com.bn.ex12h;                                                //声明包
2   import static com.bn.ex12h.MySurfaceView.*;                          //导入相关类
3   import static com.bn.ex12h.Constant.*;                               //导入相关类
4   public class ViewForDrawThread extends Thread{                       //继承 Thread 的类
5       MySurfaceView mv;                                                //成员变量
6       static boolean flag;                                             //boolean 值
7       static int STEPS=5;                                              //静态成员变量
8       public ViewForDrawThread(MySurfaceView mv){                      //构造器
9           this.mv=mv;                                                  //成员变量赋值
10      }
11      @Override
```

```
12      public void run(){                          //重写的方法
13          while(flag){                            //为 true 时执行
14              xySize=xySize+STEPS;                //改变 xySize
15              if(xySize<=10){                     //判断 xySize 是否等于小于 10
16                  STEPS=5;                        //STEPS 为 5
17              }if(xySize>=150){                   //判断 xySize 是否等于大于 150
18                  STEPS=-5;                       //STEPS 为-5
19              }
20              mv.repaint();                       //调用方法
21              try{
22                  Thread.sleep(SLEEP_TIME);       //休眠 30ms
23              }catch(Exception e){                //捕获异常
24                  e.printStackTrace();            //打印堆栈信息
25  } } } }
```

其中：

- 第 5～7 行表示该类的成员变量的声明。
- 第 8～10 行表示该类的构造器，初始化成员变量 mv。
- 第 12～25 行表示重写的 run()方法，主要是更改 xySize 的值，并判断其是否超出范围。调用刷帧方法，同时调用 Sleep 方法休息 30ms。

实例 9　左右拖拉你的界面

许多游戏中的主菜单为左右拖拉菜单，这极大地增加了游戏的趣味性与观赏性。在本节将要介绍的是如何基于 Android 平台开发左右拖拉菜单。

【实例描述】

运行本程序进入主界面，在主界面可以左右拖动菜单。但是向左拖动时不可以小于该菜单左侧的最小位置，向右拖动不可以超过该菜单右侧的最大位置。

本实例的运行效果为如图 12-24、图 12-25、图 12-26、图 12-27 所示。

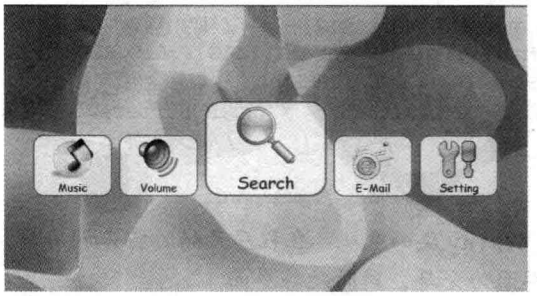


图 12-24　开始界面

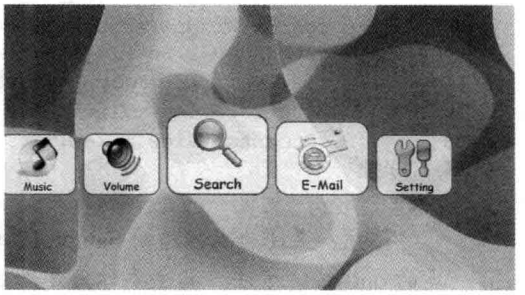


图 12-25　向右拖动中的一个界面

提示：本程序的运行效果为图 12-24、图 12-25，其中图 12-24 表示的是开始进入本程序的界面，图 12-25 表示的是向右拖动菜单时，其中的一个界面。

提示：本程序其余的运行效果为图 12-26、图 12-27，其中图 12-26 表示的是向右拖动中的另一个界面，图 12-27 表示的是拖动完成时的界面。

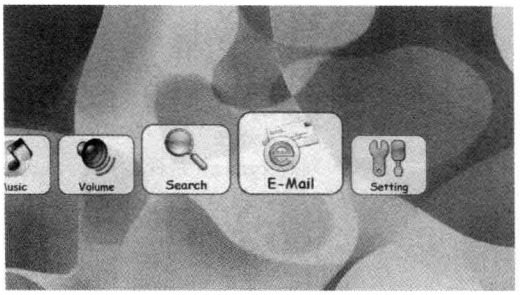


图 12-26　向右拖动中的另一个界面

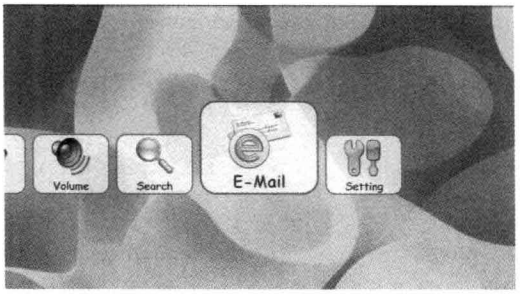


图 12-27　拖动完成界面

【实现过程】

在本程序的开发中，需要创建继承 SurfaceView 的类，在该类中需要重写 onTouchEvent 方法，同时需要写菜单绘制时用到的 onDraw 方法，并创建刷帧方法。同时需要创建一个线程类 ViewDrawThread，在拖动菜单时显示动画的效果。

【代码解析】

在本部分会详细介绍本程序具体的实现过程，首先介绍的是本程序 Sample12_9_Activity 类的开发，其为程序开始时调用的类，代码如下。

代码位置：见本书随书光盘中源代码/第 12 章/Sample12_9/src/com/bn/ex12i 目录下的 Sample12_9_Activity。

```
1   package com.bn.ex12i;                                          //声明包
2   import android.app.Activity;                                   //导入相关类
3   ......//该处省略了部分类的导入代码，读者可自行查阅随书光盘源代码
4   import android.view.WindowManager;                             //导入相关包
5   public class Sample12_9_Activity extends Activity { //创建继承 Activity 的类
6      @Override
7      public void onCreate(Bundle savedInstanceState){ //继承 Activity 需要重写的方法
8         super.onCreate(savedInstanceState);                      //调用父类
9         requestWindowFeature(Window.FEATURE_NO_TITLE);           //取出标题
10        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
11              WindowManager.LayoutParams.FLAG_FULLSCREEN); //设置全屏
12        MySurfaceView mView=new MySurfaceView(this);  //得到 MySurfaceView 对象
13        setContentView(mView);                                   //进入主界面
14  }}
```

提示：上述为本程序的 Sample12_9_Activity 类的代码，在程序开始时首先调用该类的 onCreate 方法。其主要作用是设置界面为全屏，同时跳转界面。

上述代码介绍的是本程序开始时调用的类，接下来将要介绍的是本程序的主界面搭建的 MySurfaceView 类的框架，理解该框架有助于读者更好地学习本程序，代码如下。

代码位置：见本书随书光盘中源代码/第 12 章/Sample12_9/src/com/bn/ex12i 目录下的 MySurfaceView。

```
1   package com.bn.ex12i;                                          //声明包
2   import static com.bn.ex12i.Constant.*;                         //导入相关类
3   ......// 该处省略了部分类的导入代码，读者可自行查阅随书光盘源代码
4   import android.view.SurfaceView;                               //导入相关类
```

```
5   public class MySurfaceView extends SurfaceView implements SurfaceHolder.Callback{
6       ......//该处省略了该类的成员变量的声明，读者可自行查阅随书光盘中源代码
7       public MySurfaceView(Sample12_9_Activity activity){ //构造器
8           super(activity);                                    //调用父类
9           this.activity=activity;                             //为 activity 赋值
10          this.getHolder().addCallback(this);                 //设置回调接口
11          paint=new Paint();                                  //创建 paint 对象
12          paint.setAntiAlias(true);                           //打开抗锯齿
13          initBitmap(activity.getResources());                //调用 initBitmap 方法
14          init();                                             //调用 init 方法
15      }
16      /*该处省略了该类监听触控事件的方法，将在后面给出*/
17      /*该处省略了该类的绘制图片的方法，将在后面给出*/
18      @Override                                               //重写的方法
19      public void surfaceChanged(SurfaceHolder holder, int format, int width,int
height){ }
20      @Override
21      public void surfaceCreated(SurfaceHolder holder){    //重写的方法
22          repaint();                                          //调用 repaint 方法
23      }
24      @Override
25      public void surfaceDestroyed(SurfaceHolder holder){ }    //重写的方法
26      public void repaint(){                                  //声明方法
27          SurfaceHolder mHolder=this.getHolder();
28          Canvas canvas=mHolder.lockCanvas();                 //锁定并获得画笔
29          try{
30              synchronized(mHolder){                          //判断是否锁定
31                  onDraw(canvas);                             //绘制
32              }
33          }catch(Exception e){                                //捕获异常
34              e.printStackTrace();                            //打印堆栈信息
35          }finally{
36              if(canvas!=null){                               //判断 canvas 是否为空
37                  mHolder.unlockCanvasAndPost(canvas);        //解除锁定
38              }
39      }}
40      public void initBitmap(Resources r){                    //得到图片资源的方法
41          ......//该处省略了得到该类需要的位图的代码，读者可自行查阅随书光盘源代码
42      }
43      public void init(){
44          currentWidth=bigWidth;                              //当前菜单宽度
45          currentHeight=bigHeight;                            //当前菜单高度
46          currentX=selectX;                                   //当前菜单 X 位置
47          currentY=selectY;                                   //当前菜单 Y 位置
48          rightWidth=smallWidth;                              //右侧的宽度
49          leftWidth=smallWidth;                               //左侧的宽度
50          leftHeight=smallHeight;                             //左侧的高度
51          rightHeight=smallHeight;                            //右侧的高度
52          tempxLeft=currentX-(span+leftWidth);                //左侧的 X
53          tempyLeft=currentY+(currentHeight-leftHeight);      //左侧的 Y 坐标
54          tempxRight=currentX+(span+currentWidth);            //右侧的 X
55          tempyRight=currentY+(currentHeight-rightHeight);     //右侧的 Y 坐标
56      }
57      public void drawBitmap
58      (
59          Canvas c,                                           //画布引用
60          float x,float y                                     //绘制位置 x、y 坐标
```

```
61            float xRatio,float yRatio,                    //X、Y 方向的缩放比例
62            Bitmap bm                                     //待绘制的图片
63        ){                                                //绘制图片
64            Matrix m1=new Matrix();                       //设置缩放矩阵
65            m1.setScale(xRatio, yRatio);                  //设置具体参数
66            Matrix m2=new Matrix();                       //设置平移矩阵
67            m2.setTranslate(x, y);
68            Matrix mz=new Matrix();                       //计算总矩阵
69            mz.setConcat(m2, m1);                         //设置具体参数
70            c.drawBitmap(bm, mz, paint);                  //绘制当前的菜单项
71        }
72    }
```

其中：

- 第 7～15 行表示该类的构造器，主要是初始化该类的成员变量与位图资源。
- 第 21～23 行表示进入该界面时调用的方法。
- 第 26～39 行表示刷帧的方法，主要是得到 SurfaceHolder 对象，同时锁定画笔。在使用后要将锁定的画笔解除锁定。
- 第 43～56 行表示初始化该类的成员变量的方法。
- 第 57～71 行表示创建在拖动时图片需要缩放的方法，主要是对 Matrix 的应用。

（1）上述代码已经介绍了 MySurfaceView 类的具体框架，接下来介绍的是该类监听触控事件的 onTouchEvent 方法，将下列代码插入到 MySurfaceView 框架的第 16 行。。

代码位置：见本书随书光盘中源代码/第 12 章/Sample12_9/src/com/bn/ex12i 目录下的 MySurfaceView。

```
1   @Override
2   public boolean onTouchEvent(MotionEvent e){          //方法的重写
3       if(anmiState!=0){                                //若在动画播放中则触控无效
4           return true;                                 //返回值为 true
5       }
6       float x = e.getX();                              //获取触控点的 X 坐标
7       float y = e.getY();                              //获取触控点的 Y 坐标
8       switch (e.getAction()){                          //判断触控事件
9           case MotionEvent.ACTION_DOWN:                //为按下事件
10            mPreviousX=x;                              //记录触控笔 X 位置
11            mPreviousY=y;                              //记录触控笔 Y 位置
12          break;
13          case MotionEvent.ACTION_UP:                  //为抬起事件
14           float dx=x- mPreviousX;                     //计算 X 位移
15           if(dx<-slideSpan){                          //X 位移小于阈值则向左滑动
16            if(currentIndex<menu.length-1){            //当前菜单项不为最后一个
17                int afterCurrentIndex=currentIndex+1; //滑动完成后的菜单项编号
18                anmiState=2;                           //向左走
19                 new ViewDrawThread(this,afterCurrentIndex).start();
                                                         //启动线程并更新状态
20                }
21           } else if(dx>slideSpan){                    //X 位移大于阈值则向右滑动
22              if(currentIndex>0){                      //当前菜单项不是第一个
23                int afterCurrentIndex=currentIndex-1; //滑动完成后的菜单项编号
24                 anmiState=1;                          //向右走
25                 new ViewDrawThread(this,afterCurrentIndex).start();
                                                         //启动线程并更新状态
26          } }
27           break;                                      //退出
```

```
        }
        return true;                                    //返回值为 true
    }
```

提示：该类主要为菜单项添加监听，首先需要判断是否是播放状态，播放状体则反之 true。在不为播放状态时需要判断发生的是单击事件还是抬起事件，若为抬起事件，则需要判断 X 的位移是否大于阀值，大于阀值则移动，否则不做任何的处理。

（2）上述代码已经介绍了该类监听触控事件的 onTouchEvent 方法，接下来介绍的是绘制该类菜单项图片的 onDraw 方法，将下列代码插入到 MySurfaceView 框架的第 17 行。。

代码位置：见本书随书光盘中源代码/第 12 章/Sample12_9/src/com/bn/ex12i 目录下的 MySurfaceView。

```
@Override
public void onDraw(Canvas canvas){
    canvas.drawBitmap(bj, 0, 0, paint);                 //绘制背景
    float ratioX=currentWidth/initial_Width;            //计算 X 缩放比
    float ratioY=currentHeight/initial_Height;          //计算 Y 缩放比
    drawBitmap(canvas,currentX,currentY,ratioX,ratioY,menu[currentIndex]);
//绘制当前菜单项
    if(currentIndex>0){                                 //当前菜单项不是第一项
        ratioX=leftWidth/initial_Width;                 //计算 X 缩放比
        ratioY=leftHeight/initial_Height;               //计算 Y 缩放比
        drawBitmap(canvas,tempxLeft, tempyLeft,ratioX,ratioY,menu[current
Index-1]);
    }
    if(currentIndex<menu.length-1){                     //当前菜单项不是最后一项
        ratioX=rightWidth/initial_Width;                //计算 X 缩放比
        ratioY=rightHeight/initial_Height;              //计算 Y 缩放比
        drawBitmap(canvas,tempxRight,tempyRight,ratioX,ratioY,menu[current
Index+1]);
    }
    for(int i=currentIndex-2;i>=0;i--){                 //向左绘制其他未选中菜单
        float tempx=tempxLeft-(span+smallWidth)*(currentIndex-1-i);
                                                        //计算 X 值
        if(tempx<-smallWidth){                          //超出范围
            break;                                      //退出
        }
        int tempy=selectY+(bigHeight-smallHeight);      //计算 Y 值
        ratioX=smallWidth/initial_Width;                //计算缩放比
        ratioY=smallHeight/initial_Height;
        drawBitmap(canvas,tempx,tempy,ratioX,ratioY,menu[i]);
                                                        //绘制当前菜单项
    }
    for(int i=currentIndex+2;i<menu.length;i++){        //向右绘制其他未选中菜单
        float tempx=tempxRight+rightWidth+span+(span+smallWidth)*(i-
(currentIndex+1)-1);
        if(tempx>screenWidth){                          //超出范围
            break;                                      //退出
        }
        int tempy=selectY+(bigHeight-smallHeight);      //计算 Y 值
        ratioX=smallWidth/initial_Width;                //计算 X 缩放比
        ratioY=smallHeight/initial_Height;              //计算 Y 缩放比
        drawBitmap(canvas,tempx,tempy,ratioX,ratioY,menu[i]); //绘制当前菜单项
}   }
```

提示：上述介绍的方法主要在绘制菜单项时调用。绘制时首先需要判断是否为第一项或者最后一项，第一项前无菜单项，最后一项后无菜单项。在绘制当前菜单项的左侧或者右侧的菜单项时，需要判断是否可以再绘制，如果可以则绘制，否则退出。

上述代码已经介绍了该类的界面的绘制的方法，接下来将要介绍的是实现菜单项拖动时动画效果的线程类，代码如下。

代码位置：见本书随书光盘中源代码/第 12 章/Sample12_9/src/com/bn/ex12i 目录下的 MySurfaceView。

```
1   package com.bn.ex12i;                                          //声明包
2   import static com.bn.ex12i.Constant.*;                         //静态导入
3   public class ViewDrawThread extends Thread{                    //创建继承 Thread 的类
4       MySurfaceView mv;                                          //成员变量
5       int afterCurrentIndex;                                     //int 类型成员变量
6       static boolean flag;                                       //boolean 成员变量
7       public ViewDrawThread(MySurfaceView mv,int afterCurrentIndex){ //构造器
8           this.mv=mv;
9           this.afterCurrentIndex=afterCurrentIndex; //初始化 afterCurrentIndex
10      }
11      @Override
12      public void run(){                                         //重写的方法
13        for(int i=0;i<=totalSteps;i++){                          //循环遍历
14          mv.changePercent=percentStep*i;                        //百分比
15          mv.init();                                             //初始化各个位置的值
16          if(mv.anmiState==1){                                   //向右动画
17              mv.currentX=mv.currentX+(bigWidth+span)*mv.changePercent;
                                                                   //当前菜单项的 X 的值
18              mv.currentY=mv.currentY+(bigHeight-smallHeight)*mv.
changePercent;
19              mv.currentWidth=(int)(smallWidth+
20                      (bigWidth-smallWidth)*(1-mv.changePercent));
                                                                   //当前菜单项的宽度
21              mv.currentHeight=(int)(smallHeight+
22                      (bigHeight-smallHeight)*(1-mv.changePercent));
                                                                   //当前菜单项的高度
23              mv.leftWidth=(int)(smallWidth+(bigWidth-smallWidth)*mv.
changePercent);//左侧的菜单宽度
24              mv.leftHeight=(int)(smallHeight+(bigHeight-smallHeight)*mv.
changePercent);
25          }else if(mv.anmiState==2){                             //向左移动
26              mv.currentX=mv.currentX-(smallWidth+span)*mv.changePercent;
                                                                   //当前菜单项的 X 的值
27              mv.currentY=mv.currentY+(bigHeight-smallHeight)*mv.
changePercent;
28              mv.currentWidth=(int)(smallWidth+
29                      (bigWidth-smallWidth)*(1-mv.changePercent));
                                                                   //当前的宽度
30              mv.currentHeight=(int)(smallHeight+
31                      (bigHeight-smallHeight)*(1-mv.changePercent));
                                                                   //当前的高度
32              mv.rightWidth=(int)(smallWidth+(bigWidth-smallWidth)*mv.
changePercent);//右侧菜单宽度
33              mv.rightHeight=(int)(smallHeight+(bigHeight-smallHeight)*mv.
changePercent);
34          }
```

```
35            mv.tempxLeft=mv.currentX-(span+mv.leftWidth);
                                                          //左侧菜单项的 X 的值
36            mv.tempyLeft=mv.currentY+(mv.currentHeight-mv.leftHeight);
                                                          //左侧菜单项的 Y 的值
37            mv.tempxRight=mv.currentX+(span+mv.currentWidth);
                                                          //右侧菜单项的 X 的值
38            mv.tempyRight=mv.currentY+(mv.currentHeight-mv.rightHeight);
                                                          //右侧菜单项的 Y 的值
39            mv.repaint();
40            try{
41                Thread.sleep(timeSpan);                 //休息一定时间
42            }catch(Exception e){                        //捕获异常
43                e.printStackTrace();                    //打印堆栈信息
44            }
45        }
46        mv.anmiState=0;                                 //完成后设置为无动画
47        mv.currentIndex=afterCurrentIndex;              //更新菜单编号
48        mv.init();                                      //初始化各个位置值
49        mv.repaint();                                   //刷帧方法
50    } }
```

其中：

- 第 4～6 行表示对该线程类的成员变量的声明。
- 第 7～10 行表示该线程类的构造器，主要是初始化成员变量。
- 第 12～50 行表示继承 Thread 类需要重写的方法，在每一次的拖动中首先需要初始化各个位置的值，之后要判断是向右的动画，还是向左的动画，根据判断设置各个菜单项的位置。最后需要设置为无动画、更新菜单编号以及初始化各个位置的值，同时需要调用刷帧方法。

实例 10　灵活的桌面小工具

Android 平台下存在着一类非常实用的组件——Widget，其可以将应用程序的主要功能直接放置在桌面上，便于用户查看。在本节将要对如何开发桌面 Widget 进行介绍。

【实例描述】

运行本程序进入主界面，在文本框中输入心情，单击“确定”按钮写入心情。长按桌面，弹出 Add to Home screen 对话框，选中 Widgets 弹出 Choose widget 对话框，在该对话框可以选择程序，单击其设置为要显示在桌面的 Widget 组件。

本实例的运行效果为如图 12-28、图 12-29、图 12-30 所示。

提示：本程序的运行效果为图 12-28、图 12-29、图 12-30，其中图 12-28 表示的是程序开始时的界面，图 12-29 表示的是长按桌面弹出对话框并选中 Widgets 跳转到的界面，图 12-30 行表示的是选中时间心情软件设置其为桌面 Widget 的界面。

【实现过程】

在本程序的开发过程中，主要用到 AppWidgetProvider 的继承，并重写 onDisabled、onEnabled 与 onUpdate 等方法，保证时时更新 Widget。同时创建继承 Service 的类，在该类中需要开启一

个新的线程，在该线程中每个 500ms 发送一个 Intent，保证时间的更新。其功能及关系图如图 12-31 所示。

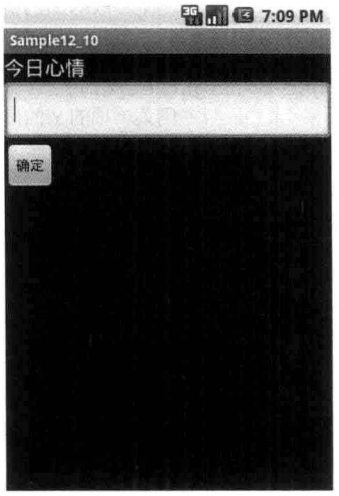


图 12-28　开始界面

图 12-29　Choose Widget 界面

图 12-30　设置为 Widget

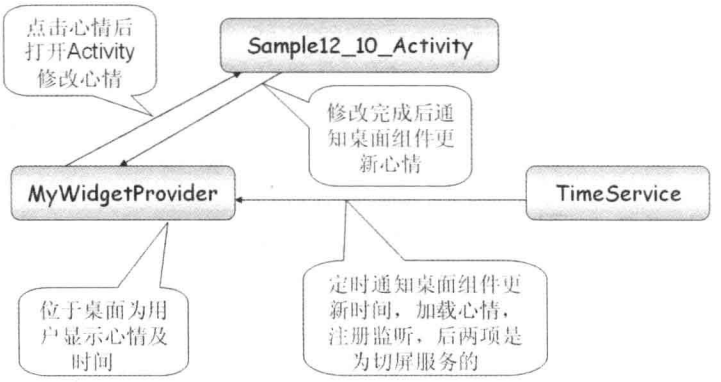


图 12-31　功能及关系

提示：在图 12-31 中可以看出各个类的功能及关系。首先单击心情 Widget 后可以打开 Activity 修改心情，修改完后需要通知桌面组件更新心情，并且在后台的 TimeService 再实时地更新时间，加载心情。

【代码解析】

在本部分会详细介绍本程序的实现过程，首先介绍的是本程序中 main.xml 文件的设置，代码如下。

代码位置：见随书光盘中源代码/第 12 章/Sample12_10/ res/layout 目录下的 main.xml。

```
1   <?xml version="1.0" encoding="utf-8"?>                    <!--版本号和编码方式-->
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3                 android:orientation="vertical"
4                 android:layout_width="fill_parent"
```

```
                android:layout_height="fill_parent"
    >                                              <!--LinearLayout 的属性设置-->
      <TextView
                android:text="今日心情"
                android:id="@+id/TextView01"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textColor="#FFFFFF"
                android:textSize="20dip"
                android:clickable="true"
    >                                              <!-- TextView 的属性设置-->
    </TextView>
    <EditText
                android:id="@+id/EditText01"
                android:singleLine="false"
                android:layout_width="fill_parent"
                android:layout_height="60dip"
    >                                              <!-- EditText 的属性设置-->
    </EditText>
    <Button
                android:text="确定"
                android:id="@+id/Button01"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content">  <!--Button 的大小-->
    </Button>                                      <!-- Button 的属性设置-->
</LinearLayout>
```

提示：上述代码为本程序的主界面的搭建，首先设置布局为 LinearLayout，设置其排列方式为竖直排列。并在该布局中添加 TextView、EditText 以及 Button 控件，然后设置其具体属性。

上述代码已经介绍了本程序的主界面，接下来介绍的是 appwidgetprovder.xml 的设置，代码如下。

代码位置：见随书光盘中源代码/第 12 章/Sample12_10/ res/layout 目录下的 main.xml。

```
<?xml version="1.0" encoding="UTF-8"?>                <!--版本号和编码方式-->
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"
            android:minWidth="280dip"
            android:minHeight="60dip"
            android:initialLayout="@layout/wmain"> <!--布局方式-->
</appwidget-provider>
```

提示：上述代码介绍的是设置为桌面 Widget 后显示的格式。

上述 xml 代码已经介绍完本程序中所有界面的搭建，接下来将要介绍的是本程序的主控制类 Sample12_10_Activity 的开发，代码如下。

代码位置：见本书随书光盘中源代码/第 12 章/Sample12_10/src/com/bn/ex12j 目录下的 Sample12_10_Activity。

```
package com.bn.ex12j;                                  //声明包
import android.app.Activity;                           //导入相关类
......// 该处省略了部分类的导入代码，读者可自行查阅随书光盘源代码
import android.widget.Toast;                           //导入相关类
public class Sample12_10_Activity extends Activity{ //创建继承 Activity 的类
    EditText et;                                       //成员变量
   @Override
```

```
8       public void onCreate(Bundle savedInstanceState){        //需要重写的方法
9           super.onCreate(savedInstanceState);                 //调用父类
10          setContentView(R.layout.main);
11          et=(EditText)this.findViewById(R.id.EditText01);    //初始化 et
12          Button b=(Button)this.findViewById(R.id.Button01);//得到按钮的对象
13          b.setOnClickListener(                               //添加监听
14              new OnClickListener(){                          //匿名内部类
15                      @Override
16                      public void onClick(View v){            //重写的方法
17                          String msg=et.getText().toString();     //生成字符串
18                          if(msg.trim().length()==0){    //若输入的心情为空则提示
19                              Toast.makeText(                 //弹出 Toast
20                                      Sample12_10_Activity.this,
21                                      "心情不能为空!!! ",
22                                      Toast.LENGTH_SHORT
23                              ).show();                       //显示 Toast
24                              return;                         //返回
25                          }else if(msg.length()>12){          //若输入超过长度则提示
26                              Toast.makeText(                 //弹出 Toast
27                                      Sample12_10_Activity.this,
28                                      "心情不能大于 12 个字!!! ",
29                                      Toast.LENGTH_SHORT
30                              ).show();                       //显示
31                              return;                         //返回
32                          }
33                      Intent intent = new Intent("wyf.action.update_xq");
34                      intent.putExtra("xxq", msg);            //存入数据
35                      Sample12_10_Activity.this.sendBroadcast(intent);
36                      Sample12_10_Activity.this.finish();  //结束
37  } } ); } }
```

提示：该类为本程序的主控制类。跳转界面，同时对按钮添加监听器，判断输入的长度是否符合要求。同时创建 Intent 对象，写入数据并发送。

上述代码介绍了本程序主控制类的实现，接下来介绍的是显示桌面 Widget 的类的开发，代码如下。

代码位置：见本书随书光盘中源代码/第 12 章/Sample12_10/src/com/bn/ex12j 目录下的 MyWidgetProvide。

```
1   package com.bn.ex12j;                                       //声明包
2   import android.app.PendingIntent;                           //导入相关类
3   ......//该处省略了部分类的导入代码，读者可自行查阅随书光盘源代码
4   import android.widget.RemoteViews;                          //导入相关类
5   public class MyWidgetProvider  extends AppWidgetProvider{ //创建继承 Activity 的类
6       RemoteViews rv;                                         //成员变量
7       public MyWidgetProvider(){                              //构造器
8           Log.d("MyWidgetProvider","============");          //打印信息
9       }
10      @Override
11      public void onDisabled(Context context) {               //若为最后一个实例
12          //删除时停止后台定时更新 Widget 时间的 Service
13          context.stopService(new Intent(context,TimeService.class)) //关闭后台
14      }
15      @Override
16      public void onEnabled (Context context){            //若为第一个实例则打开服务
```

```
17          //启动后台定时更新时间的 Service
18          context.startService(new Intent(context,TimeService.class));
                                                          //开启新的线程
19      }
20      @Override
21    public void onUpdate(Context context, AppWidgetManager appWidgetManager,
22                         int[] appWidgetIds){                    //重写的方法
23          rv = new RemoteViews(context.getPackageName(), R.layout.wmain);
24        Intent intent = new Intent(context,Sample12_10_Activity.class);
                                                    //创建 Intent 对象
25        PendingIntent pendingIntent=PendingIntent.getActivity(
                                                    //创建 PendingIntent 对象
26              context,                            //上下文
27              0,
28              intent,                             //返回值
29              PendingIntent.FLAG_UPDATE_CURRENT
30        );
31        rv.setOnClickPendingIntent(R.id.TextView01, pendingIntent);//添加监听
32        SharedPreferences sp=context.getSharedPreferences("xqsj",  //创建对象
33                                    Context.MODE_PRIVATE);
34        String xqStr=sp.getString(                  //创建对象
35              "xq",   //键值
36              null    //默认值
37        );
38        if(xqStr!=null){                            //若上次心情存在则更新心情
39          rv.setTextViewText(R.id.TextView01, xqStr);//调用 setText 方法
40        }
41        appWidgetManager.updateAppWidget(appWidgetIds, rv);    //调用方法
42      }
43      @Override                                   //onReceiver 为接收广播更新
44    public void onReceive(Context context, Intent intent){    //需要重写的方法
45          super.onReceive(context, intent);                  //调用父类
46        if (rv == null){                                     //rv 是否为空
47          rv = new RemoteViews(context.getPackageName(), R.layout.wmain);
48        }
49        if (intent.getAction().equals("wyf.action.update_xq")){ //收到的更新
50          rv.setTextViewText(R.id.TextView01, intent.getStringExtra("xxq"));
51           SharedPreferences sp=context.getSharedPreferences("xqsj",
                                                    //创建 SharedPreferences 对象
52                                Context.MODE_PRIVATE);
53          SharedPreferences.Editor editor=sp.edit();      //生产新的字符串
54          editor.putString("xq",intent.getStringExtra("xxq"));
                                                    //加入到 edit 中
55          editor.commit();                        //提交
56        }else  if (intent.getAction().equals("wyf.action.time_upadte")) {
                                                    //收到的是更新时间的 Action
57          rv.setTextViewText(R.id.TextView02,intent.getStringExtra("time"));
58        }else  if (intent.getAction().equals("wyf.action.load_xq")){
                                                    //判断横屏还是竖屏
59          Intent intentTemp = new Intent(context,Sample12_10_Activity.class);
60          PendingIntent pendingIntent=PendingIntent.getActivity(
                                                    //创建新的 PendingIntent 对象
61                  context,                        //上下文
62                  0,
63                  intentTemp,                     //intent 对象
64                  PendingIntent.FLAG_UPDATE_CURRENT
65          );
66          rv.setOnClickPendingIntent(R.id.TextView01, pendingIntent);//添加监听
```

```
67            SharedPreferences sp=context.getSharedPreferences("xqsj", Context.
MODE_PRIVATE);
68            String xqStr=sp.getString(                          //得到字符串
69                    "xq",                                       //键值
70                    null                                        //默认值
71            );
72            if(xqStr!=null){                                  //若上次心情存在则更新心情
73                rv.setTextViewText(R.id.TextView01, xqStr);//设置显示的字符串
74            }
75        }
76        AppWidgetManager appWidgetManger = AppWidgetManager.getInstance(context);
77        int[] appIds = appWidgetManger.getAppWidgetIds(    //得到数组
78          new ComponentName(
79              context,                                        //上下文
80              MyWidgetProvider.class                          //类名
81        ) );
82        appWidgetManger.updateAppWidget(appIds, rv);        //调用方法
83  } }
```

其中：

- 第 11～14 行表示重写的方法，主要作用是删除或定时更新的后台服务。
- 第 16～19 行表示重写的方法，主要作用是在当第一个实例打开时需要开启新的线程。
- 第 21～42 行表示重写的更新数据的方法，首先需要创建 Intent 与 PendingIntent 对象，然后对 Intent 对象添加监听器。其主要作用是实时的更新桌面 Widget 显示的内容。
- 第 44～83 行表示重写接收广播更新的方法，主要是将时间更新，心情更新调用方法，以键值对的形式添加进 SharedPreferences 中。并对横竖屏切换进行判断。

上述代码已经介绍了显示桌面 Widget 类的开发，接下来介绍的是更新时间类的开发，代码如下。

代码位置：见本书随书光盘中源代码/第 12 章/Sample12_10/src/com/bn/ex12j 目录下的 TimeService。

```
1   package com.bn.ex12j;                                   //声明包
2   import java.util.Date;                                  //导入相关类
3   ......// 该处省略了部分类的导入代码，读者可自行查阅随书光盘源代码
4   import android.os.IBinder;                              //导入相关类
5   public class TimeService extends Service {              //创建继承 Service 的类
6       boolean flag=true;                                  //线程循环标志
7       Thread task;                                        //定时刷新时间的任务线程
8       @Override
9       public IBinder onBind(Intent arg0){                 //重写的绑定的方法
10          return null;                                    //返回为空
11      }
12      @Override
13      public void onCreate(){                             //重写的创建的方法
14          super.onCreate();                               //调用父类
15          task=new Thread(){                              //开启新的线程
16              public void run(){                          //重写的方法
17                  while(flag){                            //定时发送 Intent 更新时间
18                      Intent intent = new Intent("wyf.action.time_upadte");
19                      Date d=new Date();                  //创建时间对象
20                      StringBuilder sb=new StringBuilder();
                                                            //创建 StringBuilder 对象
21                      ......//该处省略了部分代码，读者可自行查阅随书光盘中源代码
22                      intent.putExtra("time", sb.toString());
                                                            //在 intent 中添加数据
```

```
23                          TimeService.this.sendBroadcast(intent);
24                          intent = new Intent("wyf.action.load_xq");
                                                          //创建 intent 对象
25                          TimeService.this.sendBroadcast(intent); //发送 intent
26                          try{
27                              Thread.sleep(500);          //休息 500ms
28                          }catch (InterruptedException e){   //捕获异常
29                              e.printStackTrace();        //打印堆栈信息
30      }}}};}
31      @Override
32      public void onStart(Intent intent, int id){        //重写的方法
33          task.start();                                  //开启线程
34      }
35      @Override
36      public void onDestroy(){                           //重写的方法
37          flag=false;                                    //flag 为 false
38      }
39  }
```

其中：

- 第 9～11 行表示重写的 onBind 方法，其返回值为空。
- 第 13～30 行表示重写的 onCreate 方法，首先需要调用父类，然后开启一个新的线程，得到系统的时间，并定时的发送 Intent。
- 第 32～34 行表示重写的 onStart 方法，其主要作用是开启线程。
- 第 36～38 行表示重写的 onDestory 方法，其主要作用是将线程中的循环标志为设置为 false。

实例 11　JDBC 客户端的开发

由于在 Android 平台下的许多商业软件都需要用到 JDBC 编程技术，因此本节介绍的是如何在 Android 平台下实现 JDBC 编程。

【实例描述】

运行本程序进入主界面，在主界面单击“查询信息”按钮即可以查询学生的基本信息。本实例的运行效果如图 12-31、图 12-32 所示。

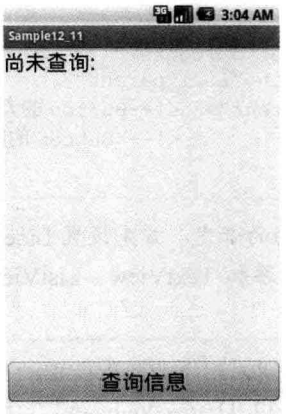


图 12-31　开始界面

图 12-32　查询结果

提示：本程序的运行效果为图 12-31、图 12-32，其中图 12-31 表示的是程序开始时的主界面，图 12-32 表示的是查询所有学生信息的结果界面。

【实现过程】

在本程序的开发过程中，用到的是 JDBC 在 Android 平台下实现编程的技术。首先需要 Connection 创建连接，然后获得 statement 对象，最后获得查询数据的结果集。在主控制类中需要对 ListView 添加适配器，并对按钮添加监听器。使单击按钮查询数据库中所有的结果。

在本程序中需要设置访问权限，其权限代码为<uses-permission android:name="android.permission.INTERNET" />。

【代码解析】

在本部分会详细介绍本程序的实现过程，首先介绍的是本程序中 main.xml 文件的设置，代码如下。

代码位置：见随书光盘中源代码/第 12 章/Sample12_10/ res/layout 目录下的 main.xml。

```
<?xml version="1.0" encoding="utf-8"?>          <!--版本号和编码方式-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:orientation="vertical"
              android:background="#FFFCCC"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
    >                                           <!--LinearLayout 的属性设置-->
    <TextView android:text="尚未查询:"
                      android:id="@+id/TextView01"
                      android:textSize="25dip"
                      android:textColor="#000000"
                      android:layout_width="fill_parent"
        android:layout_height="wrap_content">  <!--TextView 的大小-->
    </TextView>                                 <!-- TextView 的属性设置-->
    <ListView android:id="@+id/ListView01"
                      android:layout_width="fill_parent"
        android:layout_height="330dip">         <!--字体颜色-->
    </ListView>                                 <!-- ListView 的属性设置-->
    <Button android:text="查询信息"
                      android:id="@+id/Button01"
                      android:textSize="25dip"
                      android:gravity="bottom|center"
                      android:layout_width="fill_parent"
        android:layout_height="wrap_content">  <!--Button 的大小-->
    </Button>                                   <!-- Button 的属性设置-->
</LinearLayout>
```

提示：上述 main.xml 代码表示的是对主界面的开发。首先设置 LinearLayout 的属性，并设置其摆放顺序为竖直摆放，然后再其中添加 TextView、ListView、Button 控件，并设置每个控件的具体信息。

上述代码已经介绍了本程序的主界面，接下来将介绍本程序的数据库端的开发，代码如下。

代码位置：见本书随书光盘中源代码/第 12 章/Sample12_11/src/com/bn/ex12k 目录下的 DBUtil。

```
package com.bn.ex12k;                           //声明包
```

```
2   import java.sql.Connection;                                    //导入相关类
3   ......//该处省略了部分类的导入代码，读者可自行查阅随书光盘源代码
4   import java.util.ArrayList;                                    //导入相关类
5   public class DBUtil{
6      public static Connection getConnection(){                   //获得链接的方法
7           Connection con=null;                                   //引用为空
8           try{
9               Class.forName("org.gjt.mm.mysql.Driver");          //加载目标数据库驱动
10              con=DriverManager.getConnection("jdbc:mysql://192.168.9.103:
3306/test?" +
11                                  "useUnicode=true&characterEncoding=UTF-8",
"root","initial");
12          }catch(Exception e){                                   //捕获异常
13              e.printStackTrace();                               //打印堆栈信息
14          }
15          return con;                                            //返回链接
16      }
17      public static ArrayList<String[]> selectAllStudents(){   //查询所有数据
18          ArrayList<String[]> alr=new ArrayList<String[]>();
                                                  //得到ArrayList<String[]>对象
19          try{
20              Connection con=getConnection();                    //得到链接
21              Statement st=con.createStatement();                //创建 statement 对象
22              String sql="select * from student;";               //查询语句
23             ResultSet rs=st.executeQuery(sql);                  //得到结果集
24             while(rs.next()){                                   //只向下一个
25              String[] row=new String[4];                        //声明数组
26              for(int i=1;i<=4;i++){                             //for 循环
27                  row[i-1]=rs.getString(i);                      //数据存入数组中
28              }
29              alr.add(row);                                      //添加进集合中
30             }
31             rs.close();                                         //关闭结果集
32             st.close();
33             con.close();                                        //关闭连接
34          }catch(Exception e){                                   //捕获异常
35              e.printStackTrace();                               //打印堆栈信息
36          }
37          return alr;                                            //返回集合对象 alr
38  } }
```

其中：

- 第 6～16 行表示获得链接的方法，首先需要加载目标数据库的驱动类，在通过 DriverManager.getConnection 方法获得数据库的连接。
- 第 17～38 行表示得到所有学生的信息，首先需要获得连接，然后获得 statement 对象，最后获得结果集，循环遍历该结果集，将统一学生的数据以数组的形式存入集合中。

提示：在本程序中使用的数据库为 MySql，在使用之前首先需要加载对应的 jar 包。

上述程序介绍了本程序的数据库端的开发，接下来将介绍本程序的主控制类 Sample12_11_Activity 的开发，代码如下。

代码位置：见本书随书光盘中源代码/第 12 章/Sample12_11/src/com/bn/ex12k 目录下的 Sample12_11_Activity。

```
1   package com.bn.ex12k;                                          //声明包
```

```
import java.util.ArrayList;                                    //导入相关类
......//该处省略了部分类的导入代码，读者可自行查阅随书光盘源代码
import android.widget.Toast;                                   //导入相关类
public class Sample12_11_Activity extends Activity{ //创建继承 Activity 的类
   @Override
   public void onCreate(Bundle savedInstanceState){
                                                   //继承 Activity 需要重写的方法
      super.onCreate(savedInstanceState);                      //调用父类
      setContentView(R.layout.main);                           //跳转到主界面
      Button button=(Button)findViewById(R.id.Button01);          //获得引用
      button.setOnClickListener(                               //添加监听
        new OnClickListener(){                                 //匿名内部类
                @Override
                public void onClick(View v){                   //重写的方法
                    TextView tv=(TextView)findViewById(R.id.TextView01);
                    ListView lv=(ListView)findViewById(R.id.ListView01);
                    tv.setText("查询所得结果如下:"); //设置显示文字
                    setadapter(lv);                            //添加适配器
    }});}
   public void setadapter(ListView lv){                        //声明的方法
    BaseAdapter badapter=new BaseAdapter(){                    //适配器
        ArrayList<String[]> alist=DBUtil.selectAllStudents();    //集合框架
        @Override
        public int getCount(){
            return alist.size();                               //长度为集合框架的长度
        }
        @Override
        public Object getItem(int arg0){                       //重写的方法
            return null;                                       //返回为空
        }
        @Override
        public long getItemId(int arg0){                       //重写的方法
            return 0;                                          //返回为零
        }
        @Override
        public View getView(int arg0, View arg1, ViewGroup arg2){
                                                               //重写的方法
            LinearLayout ll=new LinearLayout(Sample12_11_Activity.this);
            ll.setOrientation(LinearLayout.HORIZONTAL);     //设置排列方式
            ll.setPadding(5, 5, 5, 5);                         //四周留白
            TextView tv=new TextView(Sample12_11_Activity.this);
                                                               //新建 TextView
            String[] str=alist.get(arg0);                      //得到数据存入数组中
            tv.setPadding(5, 5, 5, 5);                         //四周留白
            tv.setTextColor(Color.BLACK);                      //字体颜色
            tv.setGravity(Gravity.LEFT);                       //对齐位置
            tv.setText("姓名:"+str[0]);                        //添加显示文字
            System.out.println(str[0]);
            ll.addView(tv);                                    //加入到 LinearLayout 中
            tv=new TextView(Sample12_11_Activity.this);     //新建 TextView
            tv.setPadding(5, 5, 5, 5);                         //四周留白
            tv.setTextColor(Color.BLACK);                      //字体颜色
            tv.setGravity(Gravity.RIGHT);                      //对齐位置
            tv.setText("姓名:"+str[1]);                        //添加显示文字
            System.out.println(str[1]);
            ll.addView(tv);                                    //加入到 LinearLayout 中
            tv=new TextView(Sample12_11_Activity.this);     //新建 TextView
```

```
56                tv.setPadding(5, 5, 5, 5);                          //四周留白
57                tv.setTextColor(Color.BLACK);                       //字体颜色
58                tv.setGravity(Gravity.LEFT);                        //对齐位置
59                tv.setText("年龄:"+str[2]);                          //添加显示文字
60                System.out.println(str[2]);
61                ll.addView(tv);                                  //加入到 LinearLayout 中
62                tv=new TextView(Sample12_11_Activity.this);         //新建 TextView
63                tv.setPadding(5, 5, 5, 5);                          //四周留白
64                tv.setTextColor(Color.BLACK);                       //字体颜色
65                tv.setGravity(Gravity.RIGHT);                       //对齐位置
66                tv.setText("班级:"+str[3]);                          //添加显示文字
67                System.out.println(str[3]);
68                ll.addView(tv);                                  //加入到 LinearLayout 中
69                return ll;                                       //返会 LinearLayout
70        }};
71          lv.setAdapter(badapter);                               //添加适配器
72    }}
```

其中：

- 第 7～19 行表示重写的 onCreate 方法，在程序运行时首先调用该方法。在该方法中获得“查询信息”按钮的引用并为其添加监听器，单击该按钮不仅改变 TextView 显示的文字，而且需要为 ListView 添加适配器。
- 第 24～26 行表示设定该 ListViewde 长度为 ArrayList 集合的长度。
- 第 36～70 行表示设定 ListView 的显示，首先设置其布局为 LinearLayout，其排列方式为水平排列，设置 TextView 的左右留白、颜色为黑色、对其方式同时设置其显示的文字。最后需要添加到 LinearLayout 布局中。
- 第 71 行表示为该 ListView 添加适配器。

实例 12　新浪微博客户端的开发

在本小节中介绍的是应用新浪微博的官方 SDK 进行简单的 Android 手机客户端的开发，实现 Android 手机端个人新浪微博的登录，发文字微博和查看个人关注微博的功能。通过本小节的学习，读者可以学会新浪微博 SDK 的简单应用，并进一步的进行开发。

【实例描述】

本款软件是基于新浪微博官方 Java SDK 的二次开发，自己单独开发一个简单的 Android 手机端新浪微博。打开软件，首先是一个 3 秒的动画，显示新浪 logo，之后自动跳转到用户登录页面。在该界面可以实现用户登录的功能。登录进入微博界面，可以发表文字微博，以及查看个人关注等功能。

本实例的运行效果图如图 12-33、图 12-34 以及图 12-35 所示。

提示：在本软件运行时，首先显示一个 3 秒的欢迎动画，动画由浅变深，如图 12-33 所示，之后自动跳转到登录界面，如图 12-34 所示，登录之后跳转到主界面显示最新个人关注，如图 12-35 所示。

图 12-33　欢迎动画

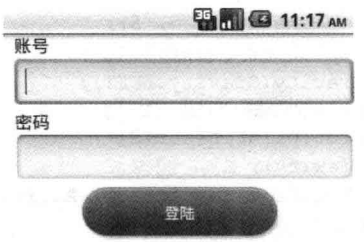


图 12-34　登录界面

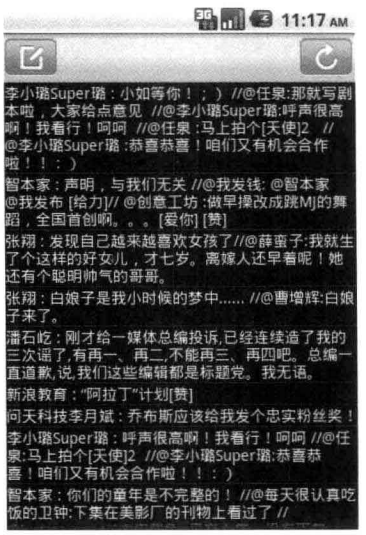


图 12-35　主界面

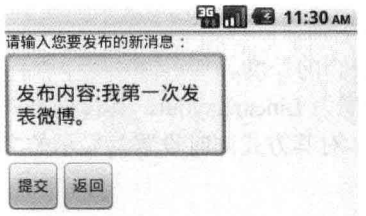


图 12-36　发表微博 1

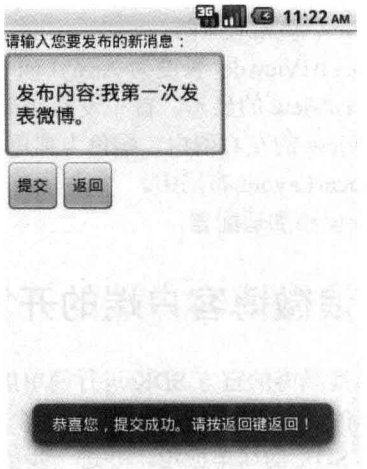


图 12-37　发表微博 2

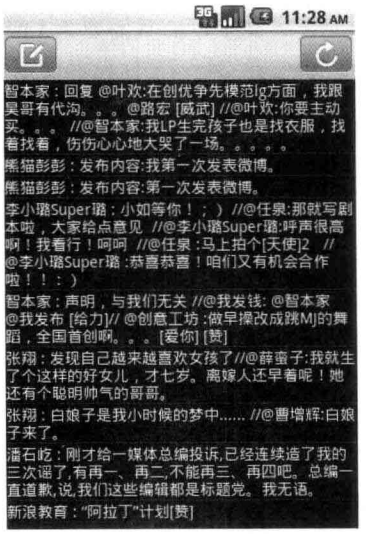


图 12-38　返回主界面

提示：单击主界面左上角的图标，则进入发表文字微博的界面，如图 12-36 所示，输入要发表的文字后单击提交按钮，显示 Toask 提示提交成功，如图 12-37 所示，单击返回按钮后返回主界面，并且自动刷新个人最新关注，可以看到刚刚发表的微博显示在其中，如图 12-38 所示。

【实现过程】

随着时代的进步，科技的发展，人们开始追求更加快捷方便的交流方式，于是微博油然而生。人们可以在任何时间，任何地点通过手机或者电脑发表微博，与朋友一起分享自己的快乐和成功。新浪微博作为国内最早诞生的微博，其粉丝在不到 2 年的时间里，就发展到了 1 亿名

注册用户，可谓是发展迅速，人们的生活也越来越离不开新浪微博。

本小节就是基于新浪微博SDK的二次开发，在开发中用到了新浪微博官方提供的Java版SDK，读者可自行在新浪微博的开发平台下载，网址为：http://60.28.113.74:8000/weibo4j-2010-12-27.zip。

在进行本软件的开发时还必须首先申请一个新浪微博的账号，读者可自行注册，网址为：http://weibo.com/?c=spr_web_sq_hao123_weibo_t001。申请完毕后登录新浪微博开发平台，去申请新建一个应用，网址为：http://open.t.sina.com.cn/，具体操作流程在此不再赘述，新建应用后请记下此应用的 App Key 和 App Secret，在开发中使用。

本软件中使用到的新浪微博提供的 API 为如表 12-1 所列。

表 12-1　使用到的 API

API 格式	使用说明
http://api.t.sina.com.cn/statuses/update.(json|xml)	发表一条新的文字微博
http://api.t.sina.com.cn/statuses/friends_timeline.(json|xml)	获取当前登录用户及其所关注用户的最新微博消息

概要介绍：

在本项目的开发中，主要是运用新浪微博的官方 SDK 进行开发，是基于互联网的开发，所以首先必须仔细阅读官方 API 文档，熟悉 SDK 的使用，利用官方提供的接口，进行单独的开发。

提示：新浪提供的 Java 版 SDK 只可直接用于 Android 2.2 及以上平台开发，如需支持兼容 Android 2.2 以下平台，读者朋友可自行修改 SDK 中相关代码。

在本程序中需要设置访问权限，其权限代码为<uses-permission android:name="android.permission.INTERNET" />。

在开发中会用到创建新浪微博应用时生成的 App Key 和 App Secret，将其替换为 Sample12_12/src/weibo4j 目录下的 Weibo.java 文件中的常量值。其中 CONSUMER_KEY=App Key，CONSUMER_SECRET=App Secret，不进行设置可能会导致无法正常使用新浪微博开发 API。

本软件开发过程中，需要导入 4 个 jar 包才可正常使用，jar 包分别为：commons-codec.jar，commons-httpclient-3.1.jar，commons-logging-1.1.jar 和 junit-4.1.jar。所有 jar 包在下载的官方 SDK 中 weibo4j\lib 目录下，只需拷贝出来，然后用 Eclipse 导入项目即可（如图 12-39 所示）。

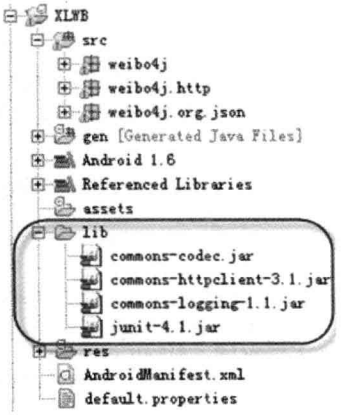


图 12-39　新浪微博客户端项目目录结构图

【代码解析】

首先向读者介绍的是本项目的 AndroidManifest.xml 文件的配置，代码如下。

代码位置：见随书光盘中源代码/第 12 章/Sample12_12 目录下的 AndroidManifest.xml。

```
<?xml version="1.0" encoding="utf-8"?>          <!--版本号和编码方式-->
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package="com.bn.ex121"
          android:versionCode="1"
    android:versionName="1.0">
   <application android:icon="@drawable/icon" android:label="@string/app_name"
    android:theme="@style/notitle">             <!--设置界面的格式-->
      <activity android:name="Logo">            <!--声明 Logo 为主 Activity-->
        <intent-filter>                         <!--添加 intent-filter-->
          <action android:name="android.intent.action.MAIN" />
          <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
      </activity>
     <activity android:name="MainActivity"></activity><!--添加MainActivity-->
   </application>
<uses-permission android:name="android.permission.INTERNET"></uses-permission>
                                                <!--声明权限 -->
</manifest>
```

提示：上述的 xml 文件是对整个项目的配置信息，其中实现了界面的全屏显示，添加了要显示的 activity，还声明了需要用到的 Internet 权限。

上面介绍的是项目的配置文件，要实现界面的全屏显示，还需要在 res/values 文件夹中添加 styles.xml 文件，代码如下。

代码位置：见随书光盘中源代码/第 12 章/Sample12_12/ res/values 目录下的 styles.xml。

```
<?xml version="1.0" encoding="utf-8"?>          <!--版本号和编码方式-->
<resources>
    <style name="notitle">                      <!--格式名称-->
        <item name="android:windowNoTitle">true</item>      <!--格式内容-->
    </style>
</resources>
```

提示：上述的 xml 文件是对软件所有界面的设置，在 AndroidManifest.xml 文件中调用，来实现所有界面都没有标题栏的效果。

介绍完基本配置文件后，接下来介绍的是欢迎界面的 Logo.java 文件，该 Activity 的代码如下。

代码位置：见本书随书光盘中源代码/第 12 章/Sample12_12/src/com/bn/ex121 目录下的 Logo。

```
package com.bn.ex121;                   //包名
import android.app.Activity; //该处省略了部分类的导入代码，读者可自行查阅随书光盘源代码
import android.widget.ImageView;
public class Logo extends Activity{     //继承 Activity
    @Override
    protected void onCreate(Bundle savedInstanceState) {//重写 onCreate 方法
        super.onCreate(savedInstanceState);
        this.setContentView(R.layout.log);              //设置 Layout
        ImageView iv=(ImageView)this.findViewById(R.id.logo_bg);
                                        //获得 ImageView
        AlphaAnimation aa=new AlphaAnimation(0.1f,1.0f);
                                        //创建 AlphaAnimation 对象
```

```
11          aa.setDuration(3000);                          //设置动画时间
12          iv.startAnimation(aa);                         //开启动画
13          aa.setAnimationListener(new AnimationListener() //添加动画监听器
14          {
15              @Override
16              public void onAnimationEnd(Animation arg0) {//动画结束后调用的方法
17                  Intent it=new Intent(Logo.this,MainActivity.class);
                                                           //创建一个 Intent
18                  startActivity(it);                     //启动 Intent
19                  finish();                              //结束当前 Activity
20              }
21              @Override
22              public void onAnimationRepeat(Animation animation) {//空实现
23              }
24              @Override
25              public void onAnimationStart(Animation animation) { //空实现
26              }});
27      }}
```

其中：

- 第 9～12 行表示获得要显示图片的 ImageView，然后创建一个 AlphaAnimation 类型的对象 aa，并且设置动画的 alpha 值为 0.1f～1.0f 渐变，设置动画时间为 3 秒，然后给 ImageView 开启动画。
- 第 13～27 行表示为 AlphaAnimation 类型的对象 aa 添加动画监听器，并且在 onAnimationEnd 方法中添加代码，在动画结束后，发送 Intent 切换到 MainActivity，并且关闭当前的 Activity 防止浪费系统资源。

上述代码实现了欢迎界面的动画显示，最后是主界面的功能实现，文件名为 MainActivity.java 的 Acivity，代码如下。

代码位置：见本书随书光盘中源代码/第 12 章/Sample12_12/src/com/bn/ex12l 目录下的 MainActivity。

```
1   package com.bn.ex12l;                                  //包名
2   import java.util.List;
3   import android.widget.Toast; //该处省略了部分类的导入代码，读者可自行查阅随书光盘源代码
4   public class MainActivity extends Activity{            //继承 Activity
5       public static Weibo weibo=new Weibo();             //创建 Weibo 对象
6       public static User currUser;                       //创建 User 对象
7       EditText etUser;
8       EditText etPass;                                   //创建 EditText 对象
9       public ListView list;                              //创建 ListView 对象
10      public static List<Status> status;                 //创建 List 对象
11      EditText etMsg;                                    //创建 EditText 对象
12      String message;                                    //创建字符串对象
13      @Override
14      public void onCreate(Bundle savedInstanceState) {  //重写 onCreate 方法
15          super.onCreate(savedInstanceState);
16          gotoLogin();                                   //调用 gotoLogin 方法
17      }
18      public void gotoLogin()                            //gotoLogin 方法
19      {   setContentView(R.layout.login);
20          etUser=( EditText)this.findViewById(R.id.user);      //获得 EditText
21         etPass=(EditText)this.findViewById(R.id.password);    //获得 EditText
22         Button btlogin=(Button)this.findViewById(R.id.loginButton);
                                                           //获得 Button
23         btlogin.setOnClickListener(new OnClickListener() //为 Button 添加监听器
```

```
24          {       @Override
25                  public void onClick(View v) {                    //重写的 onClick 方法
26                weibo.setUserId(etUser.getText().toString().trim());//设置用户 id
27                weibo.setPassword(etPass.getText().toString().trim());
                                                                     //设置用户密码
28                try{
29                      currUser=weibo.verifyCredentials();//获得当前登录的用户对象
30                } catch (WeiboException e) {                         //捕获异常
31                      e.printStackTrace();                           //打印信息
32                }
33                gotoMain();                                          //调用 gotoMain 方法
34                  }});
35          }
36          public void gotoMain()                                   //gotoMain 方法
37          {
38              setContentView(R.layout.home);
39              list=(ListView)this.findViewById(R.id.ListView01);  //获得 List
40              Button bNew=(Button)this.findViewById(R.id.title_bt_left);
41              bNew.setOnClickListener                              //为 Button 添加监听器
42              (    new OnClickListener()
43                       {    @Override
44                            public void onClick(View v) {
45                                gotoNew();                           //调用 gotoNew 方法
46                            }});
47              try{
48                  status=weibo.getFriendsTimeline(); //获得当前登录用户的微博主页信息
49              } catch (WeiboException e) {                         //捕获异常
50                  e.printStackTrace();                             //打印信息
51              }
52              BaseAdapter ba=new BaseAdapter()                     //创建适配器
53              {           @Override
54                  public int getCount() {                          //重写的 getCount 方法
55                      return status.size();                        //适配器选项的个数
56                  }
57                  @Override
58                  public Object getItem(int arg0) {                //重写的 getItem 方法
59                      return null;                                 //返回 null
60                  }
61                  @Override
62                  public long getItemId(int arg0) {       //重写的 getItemId 方法
63                      return 0;                                    //返回 0
64                  }
65                  @Override
66                  public View getView(int arg0, View arg1, ViewGroup arg2) {
67                      TextView tv=new TextView(MainActivity.this);
                                                             //创建一个 TextView 对象
68                      tv.setTextColor(Color.WHITE);        //设置文字的颜色
69                      tv.setText(status.get(arg0).getUser().getName()+" : "
                                                             //添加要显示的文字
70                              +status.get(arg0).getText());
71                      return tv;                           //返回 TextView
72                  }
73              };
74              list.setAdapter(ba);                         //添加适配器
75          }
76          public void gotoNew()                            //gotoNew 方法
77          {   setContentView(R.layout.message);
78              etMsg=(EditText)this.findViewById(R.id.EditText01); //获得 EditText
79              Button bCommit=(Button)this.findViewById(R.id.Button01);
```

```
                                                    //获得提交 Button
80          Button bBack=(Button)this.findViewById(R.id.Button02);
                                                    //获得返回 Button
81          bCommit.setOnClickListener              //为提交 Button 添加监听器
82          (   new OnClickListener()
83                  {   @Override
84                      public void onClick(View v) {
85                          message=etMsg.getText().toString().trim();
                                                    //获取输入的微博信息
86                          try{
87                              weibo.updateStatus(message); //调用方法发表微博
88                              Toast.makeText(MainActivity.this,
                                                    //显示 Toast
89                                  "恭喜您,提交成功。请按返回键返回!",3000).show();
90                          } catch (WeiboException e) {        //捕获异常
91                              e.printStackTrace();            //打印异常
92                          }}});
93          bBack.setOnClickListener                        //为返回 Buttont 添加监听器
94          (   new OnClickListener()                           //匿名内部类
95                  {   @Override
96                      public void onClick(View v) {  //重写的 onClick 方法
97                          gotoMain();                //调用 gotoMain 方法
98                      }
99           });}}
```

其中：

- 第 5～12 行表示该类的成员变量，创建了 Weibo 对象和当前 User 对象，还有必须的控件和 List 对象。
- 第 18～35 行表示 gotoLogin 方法，设置显示的 ContentView 为 login.xml，并且获得填写用户名和密码的 EditText，给 Weibo 对象设置 UserId 和 Password，得到当前用户对象。
- 第 47～51 行表示获取当前登录用户及其所关注用户的最新微博消息的方法，把获得的 Status 对象存到 List 中。
- 第 52～73 行表示给 List 添加的适配器，该适配器继承自 BaseAdapter，将获得的 Status 对象进行分析，然后以 TextView 的形式显示出来。
- 第 85-89 行表示获得用户输入的微博信息，然后用 updateStatus 方法，将此条文字微博发送到新浪微博的服务器，发送成功，然后显示 Toast 提示按“返回”按钮返回主界面查看最新微博。

小结

在本章中介绍的是手机特效的开发，主要是对屏幕切换动画、多点触控、手机动态壁纸、自动检测技术以及桌面 Widget 组件等手机效果的开发。同过本章的学习，读者应该可以学会使用本章中所讲到的任何一种特效，并且逐渐学会实例中的思想。

第 13 章　休闲游戏——Q 版疯狂大炮

休闲类游戏是一类非常流行的游戏，其画面为活泼可爱的卡通式，简单又不失可玩性，是玩家闲暇之余不错的选择。本章通过介绍一款休闲游戏——Q 版疯狂大炮在 Android 平台上的设计与实现，使读者掌握此类游戏的开发全过程。

实例 1　游戏背景及功能介绍

【实例描述】

本节将要对 Q 版疯狂大炮进行简单介绍，通过本节的学习，读者可以对 Q 版疯狂大炮游戏有初步的认识，并了解本章开发实例的具体功能及效果，为之后游戏的开发做好准备。

休闲类游戏的画面一般比较简单，没有复杂的游戏背景。虽然玩法相对简单，但要求玩家对游戏中英雄的控制能力比较高，因而此类游戏的可玩性比较强。

Q 版疯狂大炮便是此类休闲游戏的一种，玩家通过控制大炮的位置，发射炮弹的角度及力度来发射炮弹，攻击天上的目标。游戏中融入了很多卡通式素材，使玩家在游戏的同时感受童年的快乐。

实例 2　游戏实际预览效果

【实例描述】

上一小节介绍了 Q 版疯狂大炮游戏的背景，本节将对本游戏的玩法进行简单介绍，使读者了解游戏的控制方式，为后面的学习打好基础，其具体的操作方法如下。

（1）安装游戏后，打开游戏，进入游戏的欢迎动画界面，如图 13-1、图 13-2 所示。

图 13-1　欢迎动画界面 1

图 13-2　欢迎动画界面 2

（2）欢迎动画界面播放完毕后，游戏自动进入主菜单界面，如图 13-3 所示。

（3）单击主菜单中“设置音效”按钮，进入设置音效界面，如图 13-4 所示。

图 13-3　主菜单界面

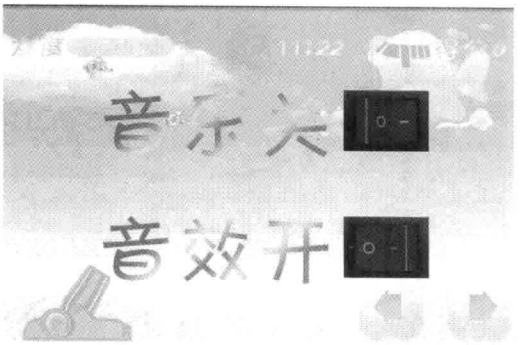


图 13-4　设置音效界面

（4）按下“返回”按钮，回到游戏主菜单界面。单击主菜单中“积分榜”按钮，进入积分榜界面，如图 13-5 所示。

（5）再次按下“返回”按钮，回到游戏主菜单界面。单击主菜单中“开始游戏”按钮，进入游戏开始界面，如图 13-6 所示。

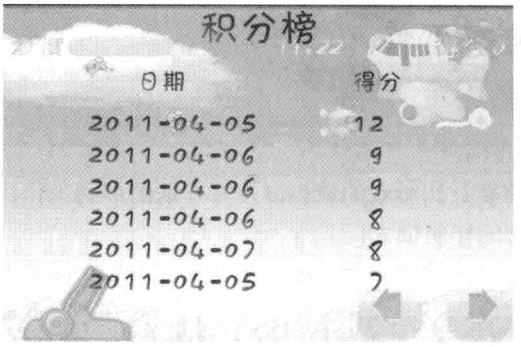


图 13-5　积分榜界面

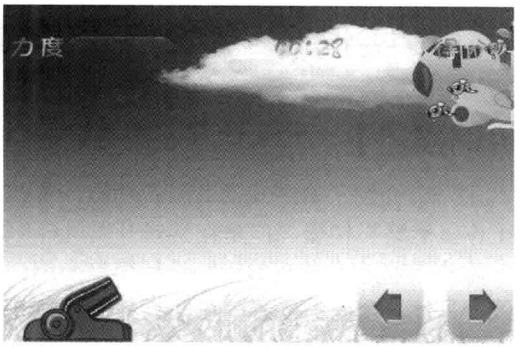


图 13-6　游戏开始界面

（6）在游戏界面中，玩家可以控制单击屏幕的位置和时间来控制大炮发射炮弹的方向和力度，通过屏幕右下角的虚拟按钮来控制大炮的位置，如图 13-7、图 13-8 所示。

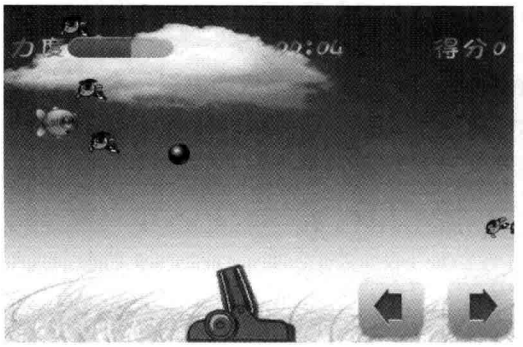


图 13-7　游戏进行中界面 1

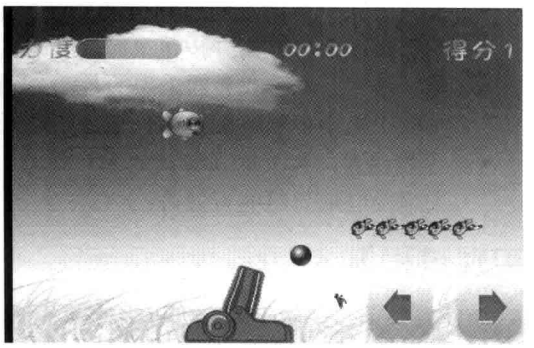


图 13-8　游戏进行中界面 2

（7）游戏界面中有时间倒计时，当时间为 0 时，游戏结束。如果玩家当前的得分大于积分

榜中最高分，进入“打破记录”的界面，如图 13-9 所示。

（8）如果游戏结束后，玩家未能打破记录，便进入“请再接再厉”的界面，如图 13-10 所示。

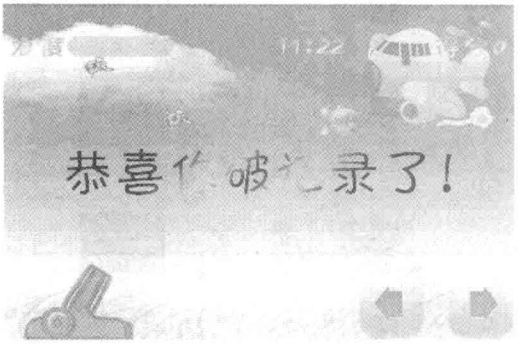


图 13-9　打破记录的界面

图 13-10　请再接再厉的界面

提示：了解游戏的规则对游戏的开发是非常重要的，建议读者在学习本游戏开发的详细步骤之前，先运行一下光盘中的该项目。

实例 3　游戏策划及准备工作

【实例描述】

上一节介绍了本游戏的背景及功能，本节将要介绍游戏的策划以及开发前的准备工作，这些工作虽然略显枯燥，但是在游戏开发过程中起的作用很大。

本小节将对本游戏的策划进行简单介绍。在真实的游戏开发中，对游戏的策划还需要更加细致、具体、全面的准备，在此只列出其中一部分。

- 游戏情节：本游戏属于休闲类游戏，情景设置非常简单，主要有英雄大炮、射击目标、飞行物以及滚屏背景等。
- 运行的目标平台：本游戏的运行目标平台为 Android 2.1、Android 2.2 或者是更高的 Android 版本。
- 操作方式：本游戏通过触摸屏幕进行操作，玩家可触控屏幕完成英雄大炮发射炮弹的动作，触控“方向”按钮来完成英雄大炮左右移动的动作。
- 呈现技术：本游戏界面采用的是 2D 贴图技术，背景采用的是横向滚屏技术，而且游戏画面中采用多层贴图技术，增加了游戏画面的层次感。
- 音效设置：为了增强游戏的用户体验，本游戏添加了背景音乐、目标爆炸的音效以及大炮发射炮弹时的音效等。

【实现过程】

游戏开发前需要做一些必不可少的准备工作，包括游戏中用到的图片、声音素材等。本实例所用到的部分图片资源，如表 13-1 所列。

表 13-1　图片清单

图片名	大小 (kB)	像素 (w×h)	用途	图片名	大小 (kB)	像素(w×h)	用　途
background.png	310	1472×320	滚屏背景	win.png	15	364×44	胜利界面
bg.gif	60	480×320	界面背景	ufo.png	8	64×53	飞碟
bullet.png	4	20×20	炮弹	riqi.png	5	40×20	日期文字
downbar.png	4	100×20	底层力度条	number0-9.png	4	12×20	数字 0-9
upbar.png	4	100×20	上层力度条	mainbtn0.png	5	184×42	主菜单按钮按下图片
explode0-5.png	3	30×30	爆炸图组	mainbtn1.png	5	184×42	开始游戏按钮
fish.png	6	40×31	飞鱼	mainbtn2.png	5	184×42	设置音效按钮
lose.png	17	264×48	文字	mainbtn3.png	5	184×42	积分榜按钮
n0-n9.png	4	13×20	数字	lose.png	17	265×48	失败界面
off.png	15.92	80×58	开关	icon.png	6	48×48	游戏图标
on.png	15.92	80×58	开关	gang.png	3	20×20	日期分隔符
paotai.png	8	104×36	炮台	highscore.png	8	108×33	文字积分榜
paotong.png	7	60×33	炮筒	breadmark.png	3	13×20	日间分隔符
plane.png	12	128×128	飞机	defen.png	5	50×20	彩色文字得分
superman.png	12	64×88	超人”	defen1.png	5	40×20	灰色文字得分
Target0-2.png	5	29×20	目标	bga.png	20	480×320	欢迎背景 1
yinyuekai.png	13	240×64	音乐开文字	bgb.png	20	480×320	欢迎背景 2

说明：上表列出了本实例中用到的部分位图资源详细信息，这些位图资源都存储在 res/drawable-mdpi 文件夹下。

实例 4　游戏的架构

【实例描述】

本节将对 Q 版疯狂大炮游戏的整体架构进行介绍，让读者了解该休闲游戏的整体框架，这样读者在学习后面的内容时将更加容易理解，达到事半功倍的效果。

本小节先对本游戏的框架进行介绍，使读者对游戏的架构有一个整体的把握，以便于后面对每个类的具体的学习，游戏类框架如图 13-11 所示。

说明：游戏中的类分为公共类、辅助界面相关类、游戏界面相关类、情景相关类和自定义控件及工具类，各个模块之相互关联、相互配合便组成整个项目。其中各个类功能的简介将在下一小节中给出。

【实现过程】

1. 公共类

- Activity 的实现类 GameActivity：该类继承并扩展了 Activity 类，是整个游戏程序的入口。

- 触控监听线程 KeyThread：该类为触控监听线程，每隔一定时间读取屏幕的触控状态值，并根据当前屏幕的触控状态值做出对应的操作。
- 常量类 Constant：该类封装了游戏中用到的全部常量，便于游戏的维护和管理。

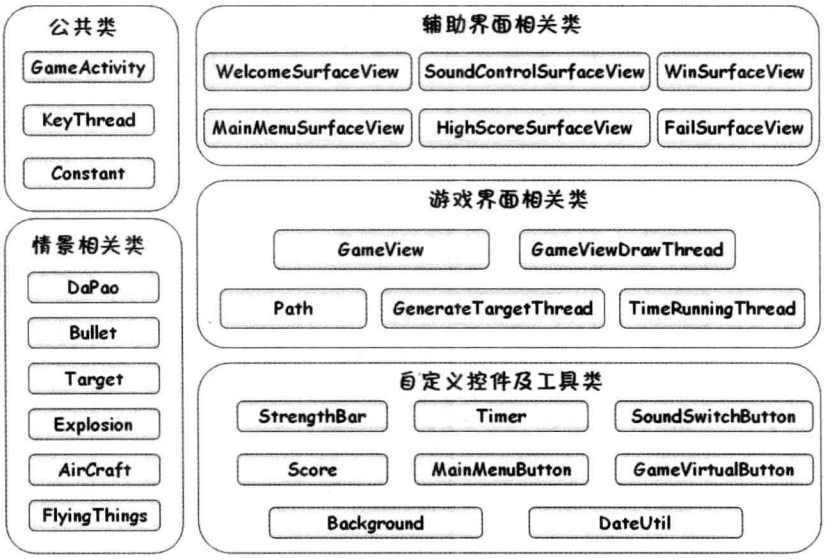


图 13-11　Q 版疯狂大炮游戏类框架图

2. 辅助界面相关类

- 欢迎界面 WellcomeSurfaceView：该类实现游戏的欢迎界面，主要负责欢迎界面的绘制工作。
- 主菜单界面 MainMenuSurfaceView：该类为游戏主菜单界面类，主要负责主菜单中虚拟按钮绘制及监听事件等工作。
- 音效控制界面 SoundControlSurfaceView：该类为游戏中音效控制界面类，其主要负责游戏中背景音乐和音效的打开或关闭。
- 积分榜界面 HighScoreSurfaceView：该类为游戏的积分榜界面，其主要负责从数据库中读取分数和时间信息，并显示在界面中。
- 胜利界面 WinSurfaceView：该类为玩家打破记录后进入的胜利界面，显示一定时间后，会自动跳转到主菜单界面。
- 失败界面 FailSurfaceView：该类为玩家没有打破记录进入的失败界面，同 WinSurfaceView 一样，也会自动跳转到主菜单界面。

3. 游戏界面相关类

- 游戏主界面类 GameView：该类为游戏的主界面类，主要负责游戏中各个对象的创建、游戏中线程控制以及事件的监听等工作。
- 主绘制线程类 GameViewDrawThread：该类为游戏界面的主绘制线程类，主要负责游戏中各个对象的不断绘制。
- 目标的路径类 Path：该类为射击目标的路径类，负责设计目标在游戏界面中的运动轨迹。
- 产生目标的线程类 GenerateTargetThread：该类为定时产生成组射击目标的线程类，主要负责按一定规律在界面中生成射击目标。

- 倒计时线程类 TimeRunningThread：该类为游戏中倒计时线程类，主要负责更改游戏界面中游戏的时间。

4．情景相关类

- 大炮类 DaPao：该类为英雄大炮的封装类，玩家可以控制其位置，发射炮弹的力度及方向。
- 炮弹类 Bullet：该类为炮弹封装类，大炮类发射的炮弹就是该类的对象。
- 目标类 Target：该类为大炮射击的目标的封装类，当此类对象被炮弹击中时，会在击中的地方绘制爆炸效果。
- 爆炸效果类 Explosion：该类为爆炸效果封装类，负责在指定位置绘制爆炸效果的某一帧。
- 飞行器类 AirCraft：该类为游戏界面高空中的飞行器类，不能被炮弹击中，只是为了增加游戏界面的层次感。
- 各飞行物类 FlyingThings：该类为游戏界面中各个飞行器对象的封装类，将各个对象封装起来，便于对象的管理。

5．自定义控件及工具类

- 力度条 StrengthBar：该类为控制炮弹发射速度的力度条封装类，当按下游戏界面中适当的位置后，力度条开始增加，抬起或力度条已满时，力度条停止增加。
- 计时器 Timer：该类为游戏中显示游戏倒计时的封装类，负责将游戏进行时间用相应的图片显示在游戏界面中。
- 声音开关按钮 SoundSwitchButton：该类为游戏音效控制界面中的声音开关按钮封装类，负责根据开关标志来绘制相应开关的图片。
- 得分类 Score：该类封装了游戏中玩家的得分，以及将得分转换成相应的数字图片的方法。
- 主菜单按钮 MainMenuButton：该类为主菜单界面中按钮的封装类，主菜单的三个按钮都是该类的对象。
- 方向控制按钮 GameVirtualButton：该类为大炮横向移动的控制按钮，按下或抬起可以很方便地控制大炮的移动，而且可以根据按下按钮时间设置大炮的移动速度。
- 滚屏背景 Background：该类为游戏滚屏背景类，主要负责不断更改绘制图片的坐标，以达到滚屏的效果。
- 获取系统时间的工具类 DateUtil：该类为封装好的得到系统时间的工具类。调用其相应的静态工具方法即可得到系统时间。

实例 5　游戏的主类代码框架

【实例描述】

从本节开始将正式介绍游戏代码的开发。为了使代码的结构更加清晰，也为了使读者了解游戏的整个开发过程，本小节将尽量按原开发顺序对各个类一一进行讲解。

【实现过程】

首先介绍的是 Activity 的实现。Activity 的作用是对各个界面进行管理、切换以及对线程发送来的请求做出响应，其代码框架如下。

【代码解析】

代码位置：见光盘源代码/第 15 章/FKDP/src/com/bn/fkdp 目录下的 GameActivity.java。

```
package com.bn.fkdp;                                                    //声明包
import android.app.Activity;                                            //引入相关类
……//此处省略了部分类的引入代码，读者可自行查看随书光盘的源代码
import android.widget.Toast;                                            //引入相关类
class WhatMessage{                                          //标识所有 SurfaceView 的常量类
    public static final int GOTO_WELLCOME_VIEW=0;                       //欢迎界面
    public static final int GOTO_MAIN_MENU_VIEW=1;                      //主菜单界面
    public static final int GOTO_GAME_VIEW=2;                           //游戏界面
    public static final int GOTO_SOUND_CONTORL_VIEW=3;                  //声音设置界面
    public static final int GOTO_WIN_VIEW=4;                            //胜利界面
    public static final int GOTO_FAIL_VIEW=5;                           //失败界面
    public static final int GOTO_HIGH_SCORE_VIEW=6;                     //积分榜界面
    public static final int OVER_GAME=7;                                //游戏结束
}
public class GameActivity extends Activity {
    int currentView;                                                    //标识当前的界面
    WellcomeSurfaceView wellcomeView;                                   //显示欢迎动画界面
    MainMenuSurfaceView mainMenuView;                                   //主菜单界面
    GameView gameView;                                                  //游戏界面
    SoundControlSurfaceView soundControlView;                           //音效控制界面
    HighScoreSurfaceView highScoreView;                                 //积分榜界面
    WinSurfaceView winView;                                             //胜利界面
    FailSurfaceView failView;                                           //失败界面
    private boolean backGroundMusicOn=true;                             //音乐是否开启的标志
    private boolean soundOn=true;                                       //音效是否开启的标志
    int currScore;                                                      //游戏结束后的得分
    int highestScore;                                                   //积分榜中最高分
    SQLiteDatabase sld;                                        //SQLiteDatabase 数据库
    Handler myHandler = new Handler(){
    ……//此处省略了对此成员变量的初始化，将在后面章节中给出};
    @Override
    public void onCreate(Bundle savedInstanceState) {                   //重写 onCreate 方法
       super.onCreate(savedInstanceState);
          requestWindowFeature(Window.FEATURE_NO_TITLE);                //设置为全屏
          getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN ,
                    WindowManager.LayoutParams.FLAG_FULLSCREEN);
          this.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_
LANDSCAPE);//设置为横屏
          gotoWellcomeView();                                           //去欢迎界面
    }
    @Override
    public boolean onKeyDown(int keyCode, KeyEvent e){                  //重写 onKeyDown 方法
    ……//此处省略了此方法的具体实现，将在后面章节中给出}
    public void sendMessage(int what){                                  //向发送信息的方法
     Message msg1 = myHandler.obtainMessage(what);                      //获得信息对象
     myHandler.sendMessage(msg1);                                       //发送信息
    }
    public boolean isBackGroundMusicOn() {                  //判断背景音乐是否开启的方法
         return backGroundMusicOn;
     }
     public void setBackGroundMusicOn(boolean backGroundMusicOn) {//开启背景音
乐的方法
         this.backGroundMusicOn = backGroundMusicOn;
```

```
52      }
53      public boolean isSoundOn() {                          //判断音效是否开启的方法
54          return soundOn;
55      }
56      public void setSoundOn(boolean soundOn) {             //关闭音效的方法
57          this.soundOn = soundOn;
58      }
59     //以下方法都省略了方法体，其具体实现将在后面章节中给出
60     private void gotoWellcomeView(){}                      //去欢迎界面的方法
61     private void gotoMainMenuView(){}                      //去主菜单界面的方法
62     private void gotoGameView(){}                          //去游戏界面的方法
63     private void gotoSoundControlView(){}                  //去设置声音界面的方法
64     private void gotoWinView(){}                           //去胜利界面的方法
65     private void gotoFailView(){}                          //去失败界面的方法
66     private void gotoHighScoreView(){}                     //去积分榜界面的方法
67     public void openOrCreateDatabase(){}                   //打开或创建数据库
68     public void closeDatabase() {}                         //关闭数据库的方法
69     public void insert(int score,String date){}            //插入记录的方法
70     public String query(int posFrom,int length){}         //查询的方法
71     public int getRowCount(){}                        //得到数据库中记录条数的方法
72     private void goToOverView(){}   //将得分和时间插入数据库，并跳转到相应的结束界面
73 }
```

其中：

- 代码第 5～14 行定义了标识所有界面的常量类，以常量代表数字的目的是增加代码的可维护性。
- 代码第 16～29 行为 GameActiviy 类中成员变量的声明，这些成员大部分是整个游戏开发过程中都会用到的全局变量。
- 代码第 47～58 行实现了对私有成员变量 backGroundMusicOn 和 soundOn 的封装。

说明：以上代码只是 GameActiviy 类的代码框架，其中大部分方法和成员变量都是空实现，不过读者不用担心，这些方法在下一小节将一一实现。

实例 6　主类中部分成员变量及方法的实现

【实现过程】

在前面小节中介绍了 GameActivity 类的代码框架，本小节将会对其中主要成员变量及方法的开发进行详细介绍，其步骤如下。

【代码解析】

（1）成员变量 myHandler 的实现。myHandler 主要负责处理其他类发送来的信息，其代码如下。将下列代码插入到 GameActivity 类的第 30 行。

代码位置：见光盘源代码/第 13 章/FKDP/src/com/bn/fkdp 目录下的 GameActivity.java。

```
1   Handler myHandler = new Handler(){          //处理各个 SurfaceView 发送的消息
2         public void handleMessage(Message msg) {      //处理消息的方法
3           switch(msg.what)                            //消息内容
4           {
5           case WhatMessage.GOTO_MAIN_MENU_VIEW:
```

```
6                gotoMainMenuView();                          //转到主菜单界面
7                break;
8            case WhatMessage.GOTO_GAME_VIEW:
9                gotoGameView();                              //转到游戏界面
10               break;
11           case WhatMessage.GOTO_SOUND_CONTORL_VIEW:
12               gotoSoundControlView();                      //转到音效设置界面
13               break;
14           case WhatMessage.GOTO_WIN_VIEW:
15               gotoWinView();                               //转到胜利界面
16               break;
17           case WhatMessage.GOTO_FAIL_VIEW:
18               gotoFailView();                              //转到失败界面
19               break;
20           case WhatMessage.GOTO_HIGH_SCORE_VIEW:
21               gotoHighScoreView();                         //转到积分榜
22               break;
23           case WhatMessage.GOTO_WELLCOME_VIEW:
24               gotoWellcomeView();                          //转到欢迎界面
25               break;
26           case WhatMessage.OVER_GAME:
27               goToOverView();                              //游戏结束选择方法
28               break;
29       }}};
```

其中：

- myHandler 继承了 Handler 类的匿名内部类的对象，主要负责发送信息，当收到信息后根据不同的信息调用其他成员方法。
- msg.what 是整型的，因此 WhatMessage 类中的常量是整型的，而不应是枚举类型。

（2）onKeyDown 方法的实现。onKeyDown 方法主要负责监听在游戏中按下返回键转到相应界面。其代码如下。将下列代码插入到 GameActivity 类的第 42 行。

代码位置：见光盘源代码/第 13 章/FKDP/src/com/bn/fkdp 目录下的 GameActivity.java。

```
1    public boolean onKeyDown(int keyCode, KeyEvent e){ //重写 onKeyDown 方法
2        switch(currentView){//当前界面
3        case WhatMessage.GOTO_WELLCOME_VIEW:              //欢迎界面
4            break;
5        case WhatMessage.GOTO_MAIN_MENU_VIEW:             //主菜单界面
6            System.exit(0);                               //退出游戏
7            break;
8        case WhatMessage.GOTO_GAME_VIEW:                  //游戏界面
9        case WhatMessage.GOTO_SOUND_CONTORL_VIEW:         //音效控制界面
10       case WhatMessage.GOTO_WIN_VIEW:                   //胜利界面
11       case WhatMessage.GOTO_FAIL_VIEW:                  //失败界面
12       case WhatMessage.GOTO_HIGH_SCORE_VIEW:            //积分榜界面
13           gotoMainMenuView();                           //去主菜单界面
14       }
15           return true;
16   }
```

其中：

- 每个 SurfaceView 中也有 onKeyDown 方法，但在跳转界面时只能用 activity 中的此方法。
- 成员变量 currentView 的作用是标记当前界面是哪个 SurfaveView，当在相应界面中按下返回键时，此方法会根据 currentView 的值判断具体转到哪一个界面。

（3）gotoMainMenuView 、gotoWellcomeView 等方法的实现，gotoMainMenuView 方法的

代码如下。将下列代码插入到 GameActivity 类的第 61 行。

代码位置：见随书光盘源代码/第 13 章/FKDP/src/com/bn/fkdp 目录下的 GameActivity.java。

```
1   private void gotoMainMenuView(){                          //去主菜单界面的方法
2       if(mainMenuView==null){
3           mainMenuView = new MainMenuSurfaceView(this);     //创建主菜单界面对象
4       }
5       this.setContentView(mainMenuView);                    //将界面设置为主菜单界面
6       currentView=WhatMessage.GOTO_MAIN_MENU_VIEW;          //标记主菜单界面为当前界面
7     }
```

提示：这些方法都是私有的，在其他类中想要实现跳转功能必须通过 Handler 发送请求消息来实现。其他方法和此方法非常类似，由于本书篇幅有限，这里不再赘述，读者可自行查阅随书光盘中源代码。

（4）openOrCreateDatabase 方法的实现，其代码如下。将下列代码插入到 GameActivity 类的第 67 行。

代码位置：见光盘源代码/第 13 章/FKDP/src/com/bn/fkdp 目录下的 GameActivity.java。

```
1   public void openOrCreateDatabase(){                       //打开或创建数据库的方法
2       try{
3           sld=SQLiteDatabase.openDatabase    (              //打开数据库
4                   "/data/data/bn.qiulei/mydb",              //数据库所在路径
5                   null,                                     //CursorFactory
6                   SQLiteDatabase.OPEN_READWRITE|SQLiteDatabase.CREATE_IF_
NECESSARY
7           );
8           String sql="create table if not exists highScore" + //创建表的 sql 语句
9                   "( " +
10                  "score integer," +
11                  "date varchar(20)" +
12                  ");";
13          sld.execSQL(sql);                                 //执行 sql 语句
14      }catch(Exception e){
15          Toast.makeText(this,"数据库错误: "+e.toString(),Toast.LENGTH_SHORT).
show();//显示异常
16      }}
```

其中：

- 代码第 3～7 行为创建 SQLite 数据库方法，第三个参数的含意为读/写、若不存在则创建数据库。
- 代码第 8～13 行为开发创建表的 sql 语句并执行的代码，表中只有分数和时间两个列。

（5）insert 方法的实现。该方法是对数据库中创建的 highScore 表中插入一条记录的方法，其代码如下。将下列代码插入到 GameActivity 类的第 69 行。

代码位置：见光盘源代码/第 13 章/FKDP/src/com/bn/fkdp 目录下的 GameActivity.java。

```
1   public void insert(int score,String date){                //插入记录的方法
2       try{
3           String sql="insert into highScore values("+score+",'"+date+"');";
                                                              //插入一条记录语句
4           sld.execSQL(sql);                                 //执行 sql 语句
5           sld.close();                                      //关闭数据库
6       }catch(Exception e){
7               Toast.makeText(this, "数据库错误: "+e.toString(), Toast.LENGTH_
SHORT).show();;
```

```
8            }
9        }
```

提示：该方法和打开数据库的方法类似，也是执行一条 sql 语句即可实现插入一条数据。由于关闭数据库的方法比此方法还要简单，这里不再赘述，读者可自行查阅随书光盘中源代码。

（6）query 方法的开发。该方法主要负责将数据库中所有的记录以特定格式的字符串的形式返回。其代码如下。将下列代码插入到 GameActivity 类的第 70 行。

代码位置：见光盘源代码/第 13 章/FKDP/src/com/bn/fkdp 目录下的 GameActivity.java。

```
1    public String query(int posFrom,int length){          //查询的方法
2        StringBuilder sb=new StringBuilder();              //要返回的结果
3        Cursor cur=null;                                   //声明游标引用
4        openOrCreateDatabase();                            //打开数据库
5          String sql="select score,date from highScore order by score desc;";
                                                            //查询 sql 语句
6          cur=sld.rawQuery(sql, null);                     //执行 sql 语句
7        try{
8            cur.moveToPosition(posFrom);                 //将游标移动到指定的开始位置
9            int count=0;                                   //当前查询记录条数
10           while(cur.moveToNext()&&count<length){         //循环查询结果临时表
11               int score=cur.getInt(0);                   //获得分数
12               String date=cur.getString(1);              //获得日期
13               sb.append(score);                          //追加分数
14               sb.append("/");                            //将日期和时间用"/"分隔开
15               sb.append(date);                           //追加日期
16               sb.append("/");                            //将记录用"/"分隔开
17               count++;                                   //当前查询记录数加 1
18           }
19       }catch(Exception e){
20               Toast.makeText(this, "数据库错误："+e.toString(), Toast.LENGTH_
SHORT).show();
21           }finally{
22               cur.close();                               //关闭游标
23               closeDatabase();                           //关闭数据库
24           }
25           return sb.toString();                          //转换成字符，并返回
26       }
```

其中：

- 代码第 4～5 行为打开数据库并执行查询语句的代码，注意查询结果是按得分降序排列的，这样才能保证积分榜中最前面的记录数据是最高分。
- 代码第 8～18 行将临时表中必要的数据信息取出并加入到 StringBuilder 类的对象 sb 中，其中参数 posFrom 和 length 分别代表要开始查询的位置及要查询记录的条数。
- 代码第 21～24 行为关闭已经打开的资源的代码。注意已经打开的资源一定要关闭，所以这两行代码放到了 finally 语句中。

（7）getRowCount 方法的开发。该方法的作用是得到数据库中记录的总行数，其代码如下。将下列代码插入到 GameActivity 类的第 71 行。

代码位置：见光盘源代码/第 13 章/FKDP/src/com/bn/fkdp 目录下的 GameActivity.java。

```
1    public int getRowCount(){                              //得到数据库中记录条数的方法
2        int result=0;                                      //声明返回结果
```

```
3        Cursor cur=null;//游标引用
4        openOrCreateDatabase();                                    //打开数据库
5        try{
6            String sql="select count(score) from highScore;";    //查询 sql 语句
7             cur=sld.rawQuery(sql, null);                          //执行 sql 语句
8            if(cur.moveToNext())                                   //游标移动到下一行
9                result=cur.getInt(0);                              //得到记录的总行数
10           }
11       }
12       catch(Exception e){
13               Toast.makeText(this, "数据库错误: "+e.toString(), Toast.LENGTH_
SHORT).show();
14           }finally{
15               cur.close();                                       //关闭游标
16               closeDatabase();                                   //关闭数据库
17           }
18       return result;
19     }
```

其中：

- 对于数据库中数据的查询应尽量用一句 sql 语句完成。如果先把数据取出来，用 Java 代码实现选最大值，执行速度会慢很多。
- 代码第 8 行中用到 cur.moveToNext()方法的原因是，.rawQuery 方法返回的 Cursor 类的对象指向的是结果集中第一条记录的前一行。

（8）goToOverView 方法的开发。该方法的作用是，当游戏结束时将玩家游戏记录插入到数据库中，并根据玩家的得分跳转到相应结束界面，其代码如下。将下列代码插入到 GameActivity 类的第 72 行。

代码位置：见光盘源代码/第 13 章/FKDP/src/com/bn/fkdp 目录下的 GameActivity.java。

```
1   private void goToOverView(){        //将得分和时间插入数据库，并跳转到相应的结束界面
2        Cursor cur=null;                                  //游标引用
3        openOrCreateDatabase();                           //打开或创建数据库
4        try{                                              //从数据库中选出最高分
5            String sql="select max(score) from highScore;";      //sql 语句
6            cur=sld.rawQuery(sql, null);                          //执行 sql 语句
7            if(cur.moveToNext()){                         //如果结果集不为空，移到下一行
8                this.highestScore=cur.getInt(0);          //获得最高分
9            }
10           insert(currScore,DateUtil.getCurrentDate());
                                                           //获得当前分数和日期并插入数据库
11       }catch(Exception e){
12               Toast.makeText(this, "数据库错误: "+e.toString(), Toast.LENGTH_
SHORT).show();
13           }finally{
14               cur.close();                              //关闭游标
15               closeDatabase();                          //关闭数据库
16           }
17           if(currScore>highestScore){                   //如果当前得分大于积分榜中最高分
18           this.gotoWinView();                           //进入胜利的界面
19           }
20           else{                                    //如果当前得分不大于积分榜中最高分
21               this.gotoFailView();                      //进入失败的界面
22       }}
```

提示：方法体中用到了 DateUtil 类中静态方法 getCurrentDate 来获取当前日期，读者不必担心，在这里只要知道其返回值为满足一定格式的日期字符串即可，DateUtil 类会在后面的章节中做详细介绍。

实例 7　按键响应线程类的实现

【实现过程】

KeyThread 类主要负责不断读取 GameView 类中的整型变量 keyState 各个位的值，根据各个位值的不同做出不同的反应，其代码如下。

【代码解析】

代码位置：见光盘源代码/第 13 章/FKDP/src/com/bn/fkdp 目录下的 KeyThread.java。

```
1   package com.bn.fkdp;                                          //声明包
2   public class KeyThread extends Thread {private boolean flag=true;
3       GameView gameView;                                        //gameView 的引用
4       private int sleepSpan=100;                                //线程睡眠时间
5       private float changeSpeedTime=40f;                        //改变速度的时间点
6       public KeyThread(GameView gameView){                      //构造器
7           this.gameView=gameView;
8       }
9       public void run(){                                        //重写 run 方法
10          while(flag){
11              if(!((gameView.keyState&0x10)==0)){     //第 5 位是 1，可以改变力度条
12                  gameView.strengthBar.addWidth(3.5f);
13              }
14              if(!((gameView.keyState&0x20)==0)){//第 6 位是 1，可以改变按下按钮时间
15                  gameView.btnPressTime+=3.5f;
16              }
17              if(!((gameView.keyState&0x1)==0)){//和 00001 按位或，判断第 1 位是否
为 1，标志向左移
18                  if(gameView.btnPressTime<changeSpeedTime){ //如果按键时间不足
19                      gameView.dapao.moveLeftSlowly();    //将大炮慢速左移
20                  }
21                  else{                                       //如果按键超过规定时间
22                      gameView.dapao.moveLeftFast();      //将大炮快速左移
23                  }
24              }
25              else if(!((gameView.keyState&0x2)==0)){//和 00010 按位或，判断第 2
位是否为 1，标志右移
26                  if(gameView.btnPressTime<changeSpeedTime) //如果按键时间不足
27                      gameView.dapao.moveRightSlowly();  //将大炮慢速右移
28                  }
29                  else{                                       //如果按键超过规定时间
30                      gameView.dapao.moveRightFast();     //将大炮快速右移
31                  }
32              }
33              try{
34              Thread.sleep(sleepSpan);                        //睡眠指定毫秒数
35          }
```

```
36              catch(Exception e){
37                  e.printStackTrace();                          //打印堆栈信息
38              }
39             }
40       }
41       public void setFlag(boolean flag) {                      //设置循环标志位的方法
42           this.flag = flag;
43       }
44   }
```

其中：

- 代码第 11～16 行为 keyState 的第 5 位或第 6 位为 1 时增加对力度或时间的操作。其中力度条是单独开发的一个类，而时间是 GameView 类的一个成员变量，所以对两者增加的方法有所不同。
- 代码第 17～32 行为 keyState 的第 1 位或第 2 位为 1 时更改大炮位置的操作。当按下的时间短时大炮移动速度慢，但较精细；当按下的时间超过某个值后，大炮会快速移动，但精度很粗。

实例 8　游戏常量类的设计与实现

【实现过程】

Constant 类的作用是将代码中用到的常量全部集中起来，封装成一个类。这样便于游戏开发时的调试和管理，这里也将碰撞检测的方法封装进了此常量类中，其代码如下。

【代码解析】

代码位置：见光盘源代码/第 13 章/FKDP/src/com/bn/fkdp 目录下的 Constant.java。

```
1   package com.bn.fkdp;                                           //声明包
2   public class Constant {
3       public static final int SCREEN_WIDTH=480;                  //屏幕宽度
4       public static final int SCREEN_HEIGHT=320;                 //屏幕高度
5                                                                  //关于力度条的常量
6       public static final float BAR_X=60;                        //力度条位置
7       public static final float BAR_Y=30;
8       public static final float BAR_MAX_WIDTH=100;               //力度条最大宽度
9       public static final float BAR_HEIGHT=20;                   //力度条高度
10      public static final float TEXT_WIDTH=50;                   //文字图的宽度
11      public static final float TEXT_HEIGHT=20;                  //文字图的高度
12                                                                 //关于炮弹的常量
13      public static final int BULLET_SIZE=20;                    //炮弹尺寸
14                                                                 //关于目标的常量
15      public static final int TARGET_WIDTH=29;                   //目标宽度
16      public static final int TARGET_HEIGHT=20;                  //目标长度
17      public static final int TARGET_MAX_NUM = 10;               //目标的最大数量
18                                                                 //关于大炮的常量
19      public static final int PAOTONG_WIDTH=60;                  //炮筒宽度
20      public static final int PAOTONG_HEIGHT=33;                 //炮筒高度
21      public static final int PAOTAI_WIDTH=104;                  //炮台宽度
22      public static final int PAOTAI_HEIGHT=36;                  //炮台高度
23                                                                 //关于背景的常量
24      public static final int BACKGROUND_WIDTH=1472;             //背景图片的宽度
```

```
25      public static final int BACKGROUND_HEIGHT=SCREEN_HEIGHT;//背景图片的高度
26                                                          //关于虚拟按钮的常量
27      public static final int BUTTON_WIDTH=64;            //按钮宽度
28      public static final int BUTTON_HEIGHT=65;           //按钮高度
29      public static boolean IsTwoRectCross                //判断两个矩形是否交叉的方法
30      (
31              float xLeftTop1,float yLeftTop1,float length1,float width1,
                                                            //左上点 x,y 坐标，长，宽
32              float xLeftTop2,float yLeftTop2,float length2,float width2
33      )
34      {
35          if
36          (
37                  isPointInRect(xLeftTop1,yLeftTop1,xLeftTop2,yLeftTop2,
length2,width2)||
38                  isPointInRect(xLeftTop1+length1,yLeftTop1,xLeftTop2,
yLeftTop2,length2,width2)||
39                  isPointInRect(xLeftTop1,yLeftTop1+width1,xLeftTop2,
yLeftTop2,length2,width2)||
40              isPointInRect(xLeftTop1+length1,yLeftTop1+width1,xLeftTop2,
yLeftTop2,length2,width2)||
41
42                  isPointInRect(xLeftTop2,yLeftTop2,xLeftTop1,yLeftTop1,
length1,width1)||
43                  isPointInRect(xLeftTop2+length2,yLeftTop2,xLeftTop1,
yLeftTop1,length1,width1)||
44                  isPointInRect(xLeftTop2,yLeftTop2+width2,xLeftTop1,
yLeftTop1,length1,width1)||
45              isPointInRect(xLeftTop2+length2,yLeftTop2+width2,xLeftTop1,
yLeftTop1,length1,width1)
46          )
47          {
48              return true;                              //两个矩形有交叉返回 true
49          }
50          return false;                                 //两个矩形没有交叉返回 false
51      }
52      public static boolean isPointInRect(              //一个点是否在矩形内（包括边界）
53              float pointx,float pointy,                //点的坐标
54              float xLeftTop,float yLeftTop,float length,float width
                                                          //矩形的左上点 x,y 坐标，长，宽
55      )
56      {
57          if(
58                  pointx>=xLeftTop&&pointx<=xLeftTop+length&&
                                                          //判断点是否在矩形区域内
59                  pointy>=yLeftTop&&pointy<=yLeftTop+width
60           )
61           {
62               return true;                             //在矩形区域内返回 true
63           }
64          return false;                                 //不在矩形区域内返回 false
65      }
66  }
```

其中：

- 代码第 3～28 行为游戏中用到常量的定义，适当调节这些变量的值可以很方便地调试游戏。
- 代码第 29～51 行为判断两个矩形是否相交的工具方法，基本思想是判断只要其中一个矩形的四个顶点之一在另一个矩形内，两个矩形就是相交的，反之不相交。

实例 9　欢迎动画界面的设计与实现

【实例描述】

上一节介绍了公共类的实现，本节开始介绍辅助界面相关类的实现。辅助界面相关类是指除游戏界面之外其他辅助界面，包括欢迎动画界面、主菜单界面、音效设置界面、积分榜界面、胜利界面和失败界面，下面先来介绍欢迎动画界面。

【实现过程】

本游戏欢迎动画界面的实现比较简单，主要是靠单独开启的线程来不断地调节画笔的不透明度，并不断地用该画笔绘制图片来实现的，其代码如下。

【代码解析】

代码位置：见光盘源代码/第 13 章/FKDP/src/com/bn/fkdp 目录下的 WelcomView.java。

```
1   package com.bn.fkdp;                                          //声名包
2   import android.graphics.Bitmap;                               //引入相关类
3   ……//此处省略了部分类的引入代码，读者可自行查看随书光盘的源代码
4   import android.view.SurfaceView;                              //引入相关类
5   public class WellcomeSurfaceView extends SurfaceView
6   implements SurfaceHolder.Callback {                           //实现生命周期回调接口
7       GameActivity activity;
8       Paint paint;                                              //画笔
9       int currentAlpha=0;                                       //当前的不透明值
10      int screenWidth=Constant.SCREEN_WIDTH;                    //屏幕宽度
11      int screenHeight=Constant.SCREEN_HEIGHT;                  //屏幕高度
12      int sleepSpan=50;                                         //动画的时延 ms
13      Bitmap[] logos=new Bitmap[2];                             //logo 图片数组
14      Bitmap currentLogo;                                       //当前 logo 图片引用
15      int currentX;                                             //当前位置
16      int currentY;
17      public WellcomeSurfaceView(GameActivity activity) { //构造器
18          super(activity);
19          this.activity = activity;
20          this.getHolder().addCallback(this);     //设置生命周期回调接口的实现者
21          paint = new Paint();                                  //创建画笔
22          paint.setAntiAlias(true);                             //打开抗锯齿
23          logos[0]=BitmapFactory.decodeResource(activity.getResources(),
R.drawable.bga);
24          logos[1]=BitmapFactory.decodeResource(activity.getResources(),
R.drawable.bgb);
25      }
26      public void onDraw(Canvas canvas){                        //重写 onDraw 方法
27          paint.setColor(Color.BLACK);                          //设置画笔颜色
28          paint.setAlpha(255);                                  //设置画笔透明度
29          canvas.drawRect(0, 0, screenWidth, screenHeight, paint);
                                                                  //绘制黑填充矩形清背景
30          if(currentLogo==null)return;
31          paint.setAlpha(currentAlpha);                         //设置画笔透明度
32          canvas.drawBitmap(currentLogo, currentX, currentY, paint);
                                                                  //进行平面贴图
33      }
```

```
34      public void surfaceChanged(SurfaceHolder arg0, int arg1, int arg2, int arg3) {}
35      public void surfaceCreated(SurfaceHolder holder) {  //创建时被调用
36          new Thread(){                                   //创建线程对象
37              public void run(){                          //实现 run 方法
38                  for(Bitmap bm:logos){                   //循环两张欢迎图片
39                      currentLogo=bm;                     //获得当前显示图片
40                      currentX=screenWidth/2-bm.getWidth()/2 //计算图片位置
41                      currentY=screenHeight/2-bm.getHeight()/2;
42                      for(int i=255;i>-10;i=i-10)    {
                                                    //动态更改图片的透明度值并不断重绘
43                          currentAlpha=i;                 //获得当前透明度
44                          if(currentAlpha<0){             //保证透明度大于 0
45                              currentAlpha=0;
46                          }
47                          SurfaceHolder  myholder=WellcomeSurfaceView.this.
getHolder();
48                          Canvas canvas = myholder.lockCanvas(); //获取画布
49                          try{
50                              synchronized(myholder){     //锁定 myholder
51                                  onDraw(canvas);         //绘制
52                              }
53                          }
54                          catch(Exception e){
55                              e.printStackTrace();        //打印异常
56                          }
57                          finally{
58                              if(canvas != null){         //如果画布引用不为空
59                                  myholder.unlockCanvasAndPost(canvas);
                                                            //释放画布
60                              }
61                          }
62                          try{
63                              if(i==255){             //若是新图片，多等待一会
64                                  Thread.sleep(1000);
65                              }
66                              Thread.sleep(sleepSpan);  //使线程睡眠一段时间
67                          }
68                          catch(Exception e){
69                              e.printStackTrace();        //打印异常
70                          }
71                      }
72                  }
73                  activity.sendMessage(WhatMessage.GOTO_MAIN_MENU_VIEW);
                                                            //去主菜单界面
74              }
75          }.start();                                      //启动线程
76      }
77      public void surfaceDestroyed(SurfaceHolder arg0) {} //销毁时被调用
78  }
```

其中：

- 代码第 7～16 行为成员变量的定义，其中 paint 和 currentLogo 要定义为成员变量的原因是，使用和改变两个变量的值不在同一个方法内。
- 代码第 26～33 行为 onDraw 方法的实现。onDraw 方法主要负责用成员变量 paint 来绘制图片。
- 代码第 36～75 行创建并启动了一个线程，负责动态改变成员变量 paint 和 currentLogo 的值，其中第 73 行，当动画播放完后向 activity 发送了一个去主菜单界面的消息。

实例 10　主菜单界面的设计与实现

【实现过程】

上一小节中讲到动画播放完毕后，activity 会将界面转到主菜单界面，本小节就将介绍主菜单界面的开发与实现，其代码如下。

【代码解析】

代码位置：见光盘源代码/第 13 章/FKDP/src/com/bn/fkdp 目录下的 MainMenuSurfaceView.java。

```
1   package com.bn.fkdp;                                          //声明包
2   import android.graphics.Bitmap;                               //引入相关类
3   ……//此处省略了部分类引入的代码，读者可自行查看随书光盘的源代码
4   import android.view.SurfaceView;                              //引入相关类
5   public class MainMenuSurfaceView extends SurfaceView implements SurfaceHolder.
Callback{
6       GameActivity activity;                                    //activity 的引用
7       Paint paint;                                              //画笔引用
8       DrawThread drawThread;                                    //绘制线程引用
9       Bitmap mainBtn0Bitmap;                                    //虚拟按钮图片
10      ……//此处省略了部分成员变量定义的代码，读者可自行查看随书光盘的源代码
11      MainMenuButton startGameBtn;                              //主菜单上虚拟按钮对象引用
12      MainMenuButton soundBtn;
13      MainMenuButton highScoreBtn;
14      Bitmap bgBitmap;                                          //背景图片
15      public MainMenuSurfaceView(GameActivity activity) { //构造器
16          super(activity);
17          this.activity=activity;
18          this.requestFocus();                                  //获得焦点并设置为可触控
19        this.setFocusableInTouchMode(true);
20          getHolder().addCallback(this);                        //注册回调接口
21      }
22      protected void onDraw(Canvas canvas) {                    //重写 onDraw 方法
23          super.onDraw(canvas);                                 //调用父类 onDraw 方法
24          canvas.drawColor(Color.WHITE);
25          canvas.drawBitmap(bgBitmap, 0, 0, paint);             //绘制背景
26          startGameBtn.drawSelf(canvas, paint);                 //绘制虚拟按钮
27          soundBtn.drawSelf(canvas, paint);
28          highScoreBtn.drawSelf(canvas, paint);
29      }
30      public boolean onTouchEvent(MotionEvent event) {//重写 onTouchEvent 方法
31          int x = (int) event.getX();                           //获得触控点坐标
32          int y = (int) event.getY();
33      switch(event.getAction()){                                //获得动作
34      case MotionEvent.ACTION_DOWN:                             //如果是按下动作
35          if(startGameBtn.isActionOnButton(x, y)){              //单击在开始游戏按钮上
36              startGameBtn.switchOn();                          //按下按钮
37              activity.sendMessage(WhatMessage.GOTO_GAME_VIEW);
                                                                  //发送开始游戏的消息
38          }else if(soundBtn.isActionOnButton(x, y)){ //单击在音效设置按钮上
39              soundBtn.switchOn();                              //按下按钮
40              activity.sendMessage(WhatMessage.GOTO_SOUND_CONTORL_VIEW);
41          }else if(highScoreBtn.isActionOnButton(x, y)){ //单击在积分榜按钮上
42              highScoreBtn.switchOn();                          //按下按钮
```

```
43                  activity.sendMessage(WhatMessage.GOTO_HIGH_SCORE_VIEW);
                                                                    //发送积分榜的消息
44              }
45              break;
46          case MotionEvent.ACTION_UP:                             //如果是抬起动作
47              startGameBtn.switchOff();                           //关掉按钮
48              soundBtn.switchOff();                               //关掉按钮
49              highScoreBtn.switchOff();                           //关掉按钮
50              break;
51          }
52              return true;                                        //返回 true
53          }
54          public void surfaceChanged(SurfaceHolder holder, int format, int width,
55                  int height) {}
56          public void surfaceCreated(SurfaceHolder holder){
57          ……//此处省略了加载资源和启动线程的代码，读者可自行查看随书光盘的源代码
58          }
59          public void surfaceDestroyed(SurfaceHolder holder) {
60          ……//此处省略了停止线程的代码，读者可自行查看随书光盘的源代码
61          }
62          public void initBitmap(){
63          ……//此处省略了加载图片的代码，读者可自行查看随书光盘的源代码
64          }
65          void createAllThreads(){                                //创建所有线程的方法
66              drawThread=new DrawThread(this);                    //创建绘制线程
67          }
68          void startAllThreads(){                                 //开启所有线程的方法
69              drawThread.setFlag(true);
70              drawThread.start();
71          }
72          void stopAllThreads(){                                  //停止所有线程的方法
73              drawThread.setFlag(false);
74          }
75          private class DrawThread extends Thread{
76          ……//此处省略了绘制线程类的实现，读者可自行查看随书光盘的源代码
77      }}
```

其中：

- 代码第 22～29 行为 onDraw 方法的实现。由于按钮是已经封装好了类，在创建按钮对象后，直接调用各个按钮对象的 drawSelf 方法就可以实现按钮的绘制工作。
- 代码第 30～45 行为当按下屏幕某点时，调用按钮对象的 isActionOnButton 方法来判断触控事件是否发生在该按钮上。调用 switchOn 方法即可按下按钮，同时进入相应界面。
- 代码第 46～50 行为当触控事件为抬起时，调用按钮对象的 swithOff 方法。

注意：音效控制界面和主菜单界面非常类似，由于本书篇幅所限，这里不再赘述，请读者自行查看随书光盘中的源代码。

实例 11　积分榜界面的代码框架

【实现过程】

积分榜界面主要负责从数据库中取出数据，得到一定格式的字符串，再将其用图片显示在

界面中。HighScoreSurfaceView 的代码框架如下。

【代码解析】

代码位置：见光盘源代码/第 13 章/FKDP/src/com/bn/fkdp 目录下的 HighScoreSurfaceView.java。

```
1   package com.bn.fkdp;                                          //声明包
2   import android.graphics.Bitmap;                               //引入相关类
3   ……//此处省略了部分类的引入代码，读者可自行查看随书光盘的源代码
4   import android.view.SurfaceView;                              //引入相关类
5   public class HighScoreSurfaceView extends SurfaceView implements SurfaceHolder.
Callback{
6       GameActivity activity;                                    //activity 的引用
7       Paint paint;                                              //画笔引用
8       DrawThread drawThread;                                    //绘制线程引用
9       Bitmap bgBitmap;                                          //背景图片
10      Bitmap highScoreBitmap;                                   //文字的图片
11      Bitmap defenBitmap;
12      Bitmap riqiBitmap;
13      Bitmap[] numberBitmaps;                                   //数字图片
14      Bitmap gangBitmap;                                        //符号"-"对应的图片
15      int bmpx;                                                 //文字位置
16      String queryResultStr;                                    //查询数据库的结果
17      String[] splitResultStrs;                                 //将查询结果切分后的数组
18      private int numberWidth;                                  //数字图片的宽度
19      private int posFrom=-1;                                   //查询的开始位置
20      private int length=6;                                     //查询的最大记录个数
21      int downY=0;                                              //按下和抬起的 y 坐标
22      int upY=0;
23      public HighScoreSurfaceView(GameActivity activity) {
24      ……//此处省略了构造方法体中代码，读者可自行查看随书光盘的源代码
25      }
26      protected void onDraw(Canvas canvas) {
27      ……//此处省略了绘制信息的代码，读者可自行查看随书光盘的源代码
28      }
29                                                                //画日期图片的方法
30      public void drawDateBitmap(String numberStr,float endX,float endY,Canvas
canvas,Paint paint){
31      ……//此处省略了绘制信息的代码，将在下一小节中给出
32      }
33      public boolean onTouchEvent(MotionEvent event) {//重写的 onTouchEvent 方法
34      ……//此处省略了处理触控信息的代码，将在下一小节中给出
35      }
36      public void surfaceCreated(SurfaceHolder holder){
                                                          //重写的 surfaceCreated 方法
37      ……//此处省略了部分成员变量初始化的代码，将在下一小节中给出
38      }
39  ……//以下方变量或方法与前面介绍过的界面类类似，这里只给出其框架，读者可自行查看随书光盘的
源代码
40      public void surfaceChanged(SurfaceHolder holder, int format, int width,int
height) {}
41      public void surfaceDestroyed(SurfaceHolder holder) {}
42      public void initBitmap(){}                                //初始化位图的方法
43      void createAllThreads(){}                                 //创建所有线程的方法
44      void startAllThreads(){}                                  //开启所有线程的方法
45      void stopAllThreads(){}                                   //停止所有线程的方法
46      private class DrawThread extends Thread{}                 //绘制线程类
47  }
```

实例 12　积分榜界面中部分方法的实现

【实现过程】

上一小节中介绍了 HighScoreSurfaceView 类的代码框架，本小节将介绍其部分成员方法的具体实现。这里按方法被调用的先后顺序进行介绍。

【代码解析】

（1）surfaceCreated 方法的实现。该方法主要负责成员变量的初始化工作，其代码如下。将下列代码插入到 HighScoreSurfaceView 类的第 36 行。

代码位置：见光盘源代码/第 13 章/FKDP/src/com/bn/fkdp 目录下的 HighScoreSurfaceView.java。

```
1   public void surfaceCreated(SurfaceHolder holder){
2           paint=new Paint();                                          //创建画笔
3           paint.setAntiAlias(true);                                   //打开抗锯齿
4           createAllThreads();                                         //创建所有线程
5           initBitmap();                                               //初始化位图资源
6           numberWidth=numberBitmaps[0].getWidth()+3;                  //得到数字图片的宽度
7           bmpx=(Constant.SCREEN_WIDTH-highScoreBitmap.getWidth())/2;
                                                                        //初始化图片的位置
8           posFrom=-1;                                                 //查询的开始位置置-1
9           queryResultStr=activity.query(posFrom,length);  //得到数据库中的数据
10          splitResultStrs=queryResultStr.split("/", 0); //用"/"切分，且舍掉空串
11          startAllThreads();                                          //开启所有线程
12      }
```

其中：

- 代码第 8 行中虽然成员变量 posFrom 的初值是–1，但是仍要在该方法中将其重新置为–1，否则积分榜中会记录上次翻页情况，这不符合游戏的要求。
- 代码第 9～10 行是从数据库中得到一定格式返回的字符串形式的数据，所以要用"/"将时间字符串分开，放到成员变量 splitResultStrs 中，以供在其他方法中使用。
- 代码第 11 行中线程的开启要放在最后，保证所有初始化工作全部结束后再开启线程工作。

（2）drawDateBitmap 方法的实现。该方法的作用是将成员变量 splitResultStrs 中日期信息用图片的形式显示在积分榜上，其代码如下。将下列代码插入到 HighScoreSurfaceView 类的第 30 行。

代码位置：见光盘源代码/第 13 章/FKDP/src/com/bn/fkdp 目录下的 HighScoreSurfaceView.java。

```
1   public void drawDateBitmap(String numberStr,float endX,float endY,Canvas
canvas,Paint paint){
2           for(int i=0;i<numberStr.length();i++){ //循环字日期符串
3               char c=numberStr.charAt(i);        //从左往右得到每个字符
4               if(c=='-'){                        //如果得到的字符为日期分隔符
5                   canvas.drawBitmap(gangBitmap,endX-numberWidth*(numberStr.
length()-i),
    endY, paint);                                  //绘制图片
6               }
7               else{                              //得到的字符为数字
8                   canvas.drawBitmap(             //用对应的数字图片绘制每个字符
9                       numberBitmaps[c-'0'],      //图片数组
10                      endX-numberWidth*(numberStr.length()-i),     //图片 x 位置
11                      endY,                      //图片 Y 位置
12                      paint                      //画笔
```

```
13                    );
14        }}}
```

其中：

- 代码第 1 行中方法的参数 numberStr 只能是规定好的以'-'分开的日期格式的字符串，该方法在 onDraw 方法里被调用。
- 代码第 8～13 行为用图片绘制数字的方法，该方法是通用的，在本游戏的游戏界面中绘制时间时也用到了此方法。

（3）onTouchEvent 方法的实现。该方法主要负责积分榜中数据的翻页，其代码如下。将下列代码插入到 HighScoreSurfaceView 类的第 33 行。

代码位置：见光盘源代码/第 13 章/FKDP/src/com/bn/fkdp 目录下的 HighScoreSurfaceView.java。

```
1   public boolean onTouchEvent(MotionEvent event) {
2           int y = (int) event.getY();                          //得到事件发生的 y 坐标
3       switch(event.getAction()){
4       case MotionEvent.ACTION_DOWN:                            //如果是按下动作
5           downY=y;                                             //获取动作的 y 坐标
6           break;
7       case MotionEvent.ACTION_UP:                              //如果是抬起动作
8           upY=y;                                               //再次获取动作的 y 坐标
9           if(Math.abs(downY-upY)<20){                          //在域值范围内，不翻页
10              return true;
11          }
12          else if(downY<upY){                                  //向下拖动
13              if(this.posFrom-this.length>=-1){                //如果抹到最前页，不可再抹
14                  this.posFrom-=this.length;
15              }
16          }
17          else{                                                //向上拖动
18
19              if(this.posFrom+this.length<activity.getRowCount()-1){
                                                                 //如果抹到最后页，不可再抹
20                  this.posFrom+=this.length;
21              }
22          }
23          queryResultStr=activity.query(posFrom,length);  //得到数据库中的数据
24              splitResultStrs=queryResultStr.split("/", 0);
                                                                 //用"/"切分，且舍掉空串
25          break;
26      }
27          return true;
28      }
```

其中：

- 代码第 4～11 行判断按下和抬起的位置是否超过了阈值，如果没有超过阈值判断为没有翻页。如果不设定阈值，用手机单击屏幕也可能会翻页，这显然不合要求。
- 代码第 12～25 行为当判断为翻页后再判断是向上翻还是向下翻，注意向上翻到第 1 页或向下翻到最后一页时都不可以再翻。

说明：由于胜利界面和失败界面非常简单，这里不再赘述，请读者自行查看随书光盘中的源代码。

实例 13　游戏界面显示类的代码框架

【实例描述】

上一节中介绍了辅助界面相关类的实现，本节将介绍游戏中最重要的游戏界面相关类的实现。游戏界面相关类是游戏的主体，也是实现起来最复杂的部分，下面对其进行详细介绍。首先介绍的是最后一个界面 GameView 类的设计与实现。

【实现过程】

GameView 类是游戏的核心类，其他类都是为该类服务而做的准备工作。由于 GameView 的逻辑相对复杂，本小节先介绍该类的代码框架，其代码框架如下。

【代码解析】

代码位置：见光盘源代码/第 13 章/FKDP/src/com/bn/fkdp 目录下的 GameView.java。

```
1   package com.bn.fkdp;                                          //声明包
2   import java.util.HashMap;                                     //引入相关类
3   ……//此处省略了部分类的引入代码，读者可自行查看随书光盘的源代码
4   import android.view.SurfaceView;                              //引入相关类
5   public class GameView extends SurfaceView implements SurfaceHolder.Callback{
6       GameActivity activity;                                    //activity 的引用
7       Paint paint;                                              //画笔引用
8       //线程引用
9       GameViewDrawThread drawThread;                            //绘制线程引用
10      KeyThread changeBarThread;                                //改变力度条长度的线程
11      GenerateTargetThread generateTargetThread;                //产生目标的线程
12      TimeRunningThread timeRunningThread;                      //负责计时的线程
13      //位图引用
14      Bitmap bulletBitmap;                                      //炮弹位图
15      Bitmap[] target1Bitmaps;                                  //目标 1 位图
16      Bitmap[] explodeBitmapArray;                              //爆炸位图数组
17      ……//此处省略了部分图片引用声名的代码，读者可自行查看随书光盘的源代码
18      Bitmap breakMarkBitmap;
19      //列表引用
20      List<Bullet> alBullet;                                    //炮弹列表
21      List<Target> alTarget;                                    //目标列表
22      List<Explosion> alExplosion;                              //爆炸列表
23      //对象引用
24      DaPao dapao;                                              //大炮对象引用
25      Background background;                                    //背景对象引用
26      FlyingThings flyingThings;                                //所有飞行物
27      GameVirtualButton virtualButton;                          //虚拟按钮对象引用
28      StrengthBar strengthBar;                                  //力度条对象的引用
29      Score score;                                              //得分对象的引用
30      Timer timer;                                              //计时器对象的引用
31      //状态值
32      int keyState=0;//键盘状态  1-left 2-right 4-null 8-null 16-change bar
32-button press time
33      float angle=(float) (Math.PI/6);                          //炮筒仰角
34      float btnPressTime=0;                                     //按下按钮的时间
```

```
35      //声音相关变量
36      SoundPool soundPool;                          //声音
37      HashMap<Integer, Integer> soundPoolMap;       //声音池
38      MediaPlayer mMediaPlayer;                     //音乐播放器引用
39      protected void onDraw(Canvas canvas) {        //重写的 onDraw 方法
40      ……//此处省略了方法体中的代码，将在下一小节中给出
41      }
42      public boolean onTouchEvent(MotionEvent event) {//重写的 onTouchEvent 方法
43      ……//此处省略了方法体中的代码，将在下一小节中给出
44      }
45      public void surfaceChanged(SurfaceHolder holder, int format, int width,
46              int height) {}
47      public void surfaceCreated(SurfaceHolder holder){
                                                      //重写的 surfaceCreated 方法
48      ……//此处省略了初始化资源的代码，将在下一小节中给出
49      }
50      public void overGame(){                       //结束游戏的方法
51      ……//此处省略了方法体中的代码，将在下一小节中给出
52      }
53      ……//以下方方法非常简单，这里只给出其框架，读者可自行查看随书光盘的源代码
54      public GameView(GameActivity activity) {}   //构造器
55      public void surfaceDestroyed(SurfaceHolder holder) {}
                                                      //重写的 surfaceDestroyed 方法
56      public void initBitmap(){}                    //将图片加载
57      void createAllThreads(){}                     //创建所有线程的方法
58      void startAllThreads(){}                      //开启所有线程的方法
59      void stopAllThreads(){}                       //停止所有线程的方法
60      public void initSounds(){}                    //初始化声音的方法
61      public void playSound(int sound, int loop) {}   //播放声音的方法
62  }
```

其中：

- 代码第 9～12 行声明了游戏中用到的 4 个线程，分别用于控制游戏界面、改变力度条长度、产生炮弹射击目标和游戏倒计时。
- 代码第 20～22 行声明了 3 个 List 列表引用，分别用于管理炮弹、目标和爆炸效果对象。用集合框架管理多个相同对象是非常方便的。
- 代码第 24～30 行声明了游戏界面中用到的自定义的类的对象的引用。
- 代码第 32～34 行定义了游戏中用到的常量和状态值。其中 keyState 的第 1、2、5、6 位为 1 时从低到高分别代表大炮向左移、大炮向右移、改变力度条、移动大炮的按钮按下的时间。第 4、5 位未定义，留待游戏玩法的扩充。

说明：代码框架中的方法均为空实现，部分方法的实现将在下一小节中详细介绍。

实例 14　游戏界面显示类中部分方法的实现

【实现过程】

上一小节中介绍了 GameView 类的代码框架，本小节将介绍其部分成员方法的具体实现。这里依然按方法被调用的先后顺序进行介绍。

【代码解析】

（1）surfaceCreated 方法的实现。该方法主要负责成员变量的初始化工作，其代码如下。将下列代码插入到 HighScoreSurfaceView 类的第 47 行。

代码位置：见光盘源代码/第 13 章/FKDP/src/com/bn/fkdp 目录下的 GameView.java。

```
1   public void surfaceCreated(SurfaceHolder holder){
2           paint=new Paint();                                          //创建画笔
3           paint.setAntiAlias(true);                                   //打开抗锯齿
4           createAllThreads();                                         //创建所有线程
5           initBitmap();                                               //初始化位图资源
6           initSounds();                                               //初始化声音资源
7           mMediaPlayer = MediaPlayer.create(activity, R.raw.background);
                                                                        //初始化背景音乐
8           mMediaPlayer.setLooping(true);                              //设置为循环播放
9           alBullet=new Vector<Bullet>();                              //创建炮弹列表
10          alTarget=new Vector<Target>();                              //创建目标列表
11          alExplosion=new Vector<Explosion>();                        //创建爆炸列表
12          dapao=new DaPao(this,paotongBitmap,paotaiBitmap,bulletBitmap);
                                                                        //创建大炮对象
13          background=new Background(backgroundBitmap);                //创建背景对象
14          flyingThings=new FlyingThings(airCraftBitmaps); //创建飞行物对象
15          strengthBar=new StrengthBar(this,downBarBitmap,upBarBitmap,
liduBitmap); //创建力度条对象
16          virtualButton=new GameVirtualButton(
onLeftBitmap,onRightBitmap,offLeftBitmap,offRightBitmap);   //创建虚拟按钮对象
17          score=new Score(this,defenBitmap,numberBitmaps);    //创建得分对象
18          timer=new Timer(this,breakMarkBitmap,numberBitmaps);//创建计时器对象
19          angle=(float) (Math.PI/6);                                  //炮筒仰角
20          startAllThreads();                                          //开启所有线程
21          if(activity.isBackGroundMusicOn()){                         //开启背景音乐
22              mMediaPlayer.start();
23      }}
```

其中：

- 代码第 2～19 行是各成员变量的初始化，其中自定义的对象创建时需要为各个类分配必要的图片资源，以及 this 引用。
- 代码第 21～22 行为开启背景音乐的代码，控制是否开启音乐的 boolean 型变量，没有写在该类中而是写在 activity 中的原因是在音效控制界面中要能设置该值。

（2）onDraw 方法的实现。该方法主要负责各个对象的绘制工作，其代码如下。将下列代码插入到 HighScoreSurfaceView 类的第 39 行。

代码位置：见光盘源代码/第 13 章/FKDP/src/com/bn/fkdp 目录下的 GameView.java。

```
1   protected void onDraw(Canvas canvas) {
2           super.onDraw(canvas);                                       //调用父类该方法
3           try{
4               background.drawSelf(canvas,paint);                      //画滚动背景
5               flyingThings.drawSelf(canvas,paint);                    //画飞行物
6               List<Bullet> alBullet1=new Vector<Bullet>(alBullet);
                                                                        //复制炮弹列表
7               for(Bullet b:alBullet1){                                //绘制炮弹
8                   b.drawSelf(canvas,paint);
9               }
10              dapao.drawSelf(canvas,paint);                           //绘制大炮
11              List<Target> alTarget1=new Vector<Target>(alTarget);
```

```
                                                                    //复制炮弹列表
12              for(Target t:alTarget1){                            //绘制目标
13                  t.drawSelf(canvas,paint);
14              }
15              List<Explosion> alExplosion1=new Vector<Explosion>
(alExplosion);//复制爆炸效果列表
16              for(Explosion e:alExplosion1){
17                  e.drawSelf(canvas,paint);                       //绘制爆炸效果
18              }
19              strengthBar.drawSelf(canvas,paint);                 //绘制力度条
20              virtualButton.drawSelf(canvas,paint);               //绘制虚拟按钮
21              score.drawSelf(canvas, paint);                      //绘制得分
22              timer.drawSelf(canvas, paint);                      //绘制时间
23          }
24          catch(Exception e){
25              e.printStackTrace();                                //打印异常
26      }}
```

其中：

- 由于各个类都已经封装好，onDraw 方法中的工作非常简单，直接调用各个对象的 drawSelf 方法即可，无须关心每个类具体绘制方法的实现，这充分体现了封装的优点。
- 遍历列表中对象之前先要复制列表的原因是，由于其他类中对列表中对象同时进行删除操作，可能会产生异常。

（3）onTouchEvent 方法的实现。该方法主要负责监听触控事件，并更改相应标志位，其代码如下。将下列代码插入到 HighScoreSurfaceView 类的第 42 行。

代码位置：见光盘源代码/第 13 章/FKDP/src/com/bn/fkdp 目录下的 GameView.java。

```
1   public boolean onTouchEvent(MotionEvent event) {
2           int x = (int) event.getX();                             //获得触控点坐标
3           int y = (int) event.getY();
4       switch(event.getAction()){                                  //得到触控事件类型
5       case MotionEvent.ACTION_DOWN:                               //按下事件
6           if(virtualButton.isActionOnLeftBtn(x, y)){ //如果按下左按钮
7               virtualButton.switchLeftBtnOn();                    //按下左按钮
8               keyState=keyState|0x20;
                                    //和 0010,0000 按位或，第 6 位置 1，标志增加按键时间
9               keyState=keyState|0x1;    //和 00001 按位或，第 1 位置 1，标志向左移
10          }else if(virtualButton.isActionOnRightBtn(x, y))    //如果按下右按钮
11          {
12              virtualButton.switchRightBtnOn();                   //按下右按钮
13              keyState=keyState|0x20;
                                    //和 0010,0000 按位或，第 6 位置 1，标志增加按键时间
14              keyState=keyState|0x2;    //和 00010 按位或，第 2 位置 1，标志向右移
15          }else{                                                  //按下的不是虚拟按钮
16              angle=dapao.calcuateAngle(x, y);                    //计算炮筒仰角
17              strengthBar.setWidth(0);                            //条宽度清 0
18              keyState=keyState|0x10; //和 10000 按位或，第 5 位置 1，开始改变力度条
19          }
20          break;
21      case MotionEvent.ACTION_MOVE:
22          if(!virtualButton.isActionOnBtnArea(x, y)){//如果不在虚拟按钮区域移动
23
24              angle=dapao.calcuateAngle(x, y);                    //计算炮筒仰角
25          }
26          break;
27      case MotionEvent.ACTION_UP:
```

```
28            if(!virtualButton.isActionOnBtnArea(x, y)){//如果不在虚拟按钮区域抬起
29                keyState=keyState&0xFFEF; //和 01111 按位与，第 5 位清 0，停止改变力度条
30                alBullet.add(dapao.sendBullet());        //创建炮弹对象并加入列表
31            }
32            virtualButton.switchLeftBtnOff();            //抬起虚拟按钮
33            virtualButton.switchRightBtnOff();
34            btnPressTime=0;                              //铵键时间清 0
35            keyState=keyState&0xFFDF; //和 1101,1111 按位与，第 6 位清 0，清除右移标志
36            keyState=keyState&0xFFFE; //和 1110 按位与，第 1 位清 0，清除右移标志
37            keyState=keyState&0xFFFD; //和 1101 按位或，第 2 位清 0，清除左移标志
38            break;
39        }
40            return true;                                 //返回 true
41        }
```

其中：

- 代码第 6～15 行为按下的是大炮左右移动按钮时，更改增加按键时间和大炮左右移动的状态位。因为有线程一直在监听 keyState 的状态，当状态位改变后，按键时间和大炮位置就会被改变。
- 代码 16～18 行为按下的点是在游戏界面中除大炮左右移动按钮位置后，根据按下点的位置动态地更改炮筒仰角并更改增加力度条的标志位。
- 代码第 27～37 行为触控事件为抬起时清除各标志位、抬起相应按钮或发射炮弹。其中发射炮弹直接调用大炮发射炮弹的方法，并将返回的炮弹对象加入炮弹列表中即可。

（4）overGame 方法的实现。该方法在适当的位置被调用后会立即停止游戏，其代码如下。将下列代码插入到 HighScoreSurfaceView 类的第 50 行。

代码位置：见光盘源代码/第 13 章/FKDP/src/com/bn/fkdp 目录下的 GameView.java。

```
1    public void overGame(){                                  //结束游戏的方法
2            stopAllThreads();
3          if(mMediaPlayer.isPlaying()){                      //停止背景音乐
4            mMediaPlayer.stop();
5          }
6            activity.currScore=score.getScore();//将总得分赋值给activity中的score
7            activity.sendMessage(WhatMessage.OVER_GAME);     //获得积分榜中最高分
8        }
```

提示： 游戏结束的方法代码虽然简短，但其实并不简单。因为方法都已封装好，这里直接对方法调用，读者可以查看各方法的源代码，理清其中的逻辑。

实例 15　目标路径类的实现

【实现过程】

Path 类中只有一个静态的三维数组，用于存放目标的各条路径信息，其代码如下。

【代码解析】

代码位置：见光盘源代码/第 13 章/FKDP/src/com/bn/fkdp 目录下的 Path.java。

```
1    package com.bn.fkdp;                                     //声明包
2    public class Path {
```

```
    public static final int INFINITY=999999999;      //定义一个无穷大的数
    public static final float[][][] PATHS=//路径数组，第三维每个数分别代表目标的
x,y 和到下一步的时间
    {
        {
            {480,0,40},                              //路径中第 1 个点信息
            {200,300,20},                            //路径中第 2 个点信息
            {0,0,1},                                 //路径中第 3 个点信息
            {INFINITY,INFINITY,1},                   //消失在无穷远处
        },
        {
            {480,320,40},                            //路径中第 1 个点信息
            {200,200,20},                            //路径中第 2 个点信息
            {100,200,20},                            //路径中第 3 个点信息
            {50,0,1},                                //路径中第 4 个点信息
            {INFINITY,INFINITY,1},                   //消失在无穷远处
        },
        {
            {480,200,40},                            //路径中第 1 个点信息
            {300,200,20},                            //路径中第 2 个点信息
            {0,0,1},                                 //路径中第 3 个点信息
            {INFINITY,INFINITY,1},                   //消失在无穷远处
        },
    }};
```

其中：

- 代码第 3 行定义了一个无穷大的数，用于将目标的最后位置设置为无穷远处。
- 代码第 4～25 行定义了一个静态数组，第三维每个数分别代表目标的 x,y 和到下一步的时间，当最后一个点为无穷远时，其前面一个点的第三个数应为 1，表示一步便移到无穷远处。

实例 16　产生目标线程类的实现

【实现过程】

GenerateTargetThread 类负责定时随机产生一组大炮的射击目标，按一定轨迹在游戏界面中移动，供英雄大炮射击，其代码如下。

【代码解析】

代码位置：见光盘源代码/第 13 章/FKDP/src/com/bn/fkdp 目录下的 GenerateTargetThread.java。

```
package com.bn.fkdp;                                 //声明包
public class GenerateTargetThread extends Thread {
    GameView gameView;
    private int sleepSpan=10000;                     //线程睡眠时间
    private boolean flag=true;                       //线程循环标志
    int pathIndex=0;                                 //当前路径数组下标
    private int distancespan=800;                    //成组出现目标的时间间隔
    GenerateTargetThread(GameView gameView){         //构造器
        this.gameView=gameView;
    }
    public void run(){                               //重写 run 方法
        while(flag){                                 //while 循环
```

```
13                if(gameView.alTarget.size()<Constant.TARGET_MAX_NUM){
                                                          //限制目标数量
14                    int type=(int) (Math.random()*3);  //随机产生一个目标类型值
15                    int num=3+(int) (Math.random()*3); //随机产生一个目标数量
16                    for(int i=0;i<num;i++){
17                        gameView.alTarget.add(         //创建目标对象，加入列表
new Target(gameView,gameView.target1Bitmaps,type,Path.PATHS[pathIndex]));
18                        try{
19                        Thread.sleep(distancespan);    //睡眠指定毫秒数
20                      }
21                      catch(Exception e){
22                        e.printStackTrace();           //打印堆栈信息
23                      }
24                    }
25                    pathIndex=(pathIndex+1)%Path.PATHS.length; //加载下一个路径
26                }
27                try{
28                Thread.sleep(sleepSpan);               //睡眠指定毫秒数
29              }
30              catch(Exception e){
31                e.printStackTrace();                   //打印堆栈信息
32              }
33            }
34        }
35        public void setFlag(boolean flag) {            //设置标志位的方法
36            this.flag = flag;
37    }}
```

其中：

- 代码第 3～7 行为成员变量的定义，其中 pathIndex 指 Path 类中三维数组的下标，即所有路径中的哪一条。distanceSpan 指成组的目标之间出现的时间间隔。
- 代码第 13～25 行的含意为随机产生一组目标，其总数、目标的类型是随机的，而路径是按路径数组中循环获取的。

说明：TimeRunningThread 和 GameViewDrawThread 两个线程类非常简单，前者只在线程中调用 GameView 类的 onDraw 方法来绘制游戏界面，后者调用 Timer 类的 subtractTime 方法来倒计时。因此这里不再赘述，请读者自行查看随书光盘中的源代码。

实例 17　英雄大炮类的代码框架

【实例描述】

前面章节中用到了很多读者并不熟悉的情景中的实体类，从本节开始将对这些类一一进行讲解。首先介绍的是由玩家控制的英雄大炮类。

【实现过程】

DaPao 类是由炮台和炮筒两部分组成的，炮筒可跟随炮台移动，炮筒可以转动，而炮台不能转动。代码如下。

【代码解析】

代码位置：见光盘源代码/第 13 章/FKDP/src/com/bn/fkdp 目录下的 DaPao.java。

```
1   package com.bn.fkdp;                                          //声明包
2   import android.graphics.Bitmap;                               //引入相关类
3   ……//此处省略了部分类的引入代码，读者可自行查看随书光盘的源代码
4   import android.graphics.Paint;                                //引入相关类
5   public class DaPao {
6       GameView gameView;
7       float x=Constant.PAOTAI_WIDTH/2;                          //炮筒横放时左上角位置
8       float y=Constant.SCREEN_HEIGHT-Constant.PAOTONG_HEIGHT-15;
9       final int paotongWidth=Constant.PAOTONG_WIDTH;            //炮筒横放时的宽度
10      final int paotongHeight=Constant.PAOTONG_HEIGHT;          //炮筒横放时的高度
11      final float centerRatio=0.3f;                             //中心点占宽度的比例
12      final float center=paotongWidth*centerRatio;              //中心点位置
13      float xspanSlow=3;                                        //x 方向每次慢移的步进
14      float xspanFast=10;                                       //x 方向每次快移的步进
15      float paotaiX;                                            //炮台的位置
16      float paotaiY;
17      final int paotaiWidth=Constant.PAOTAI_WIDTH;              //炮台宽度
18      final int paotaiHeight=Constant.PAOTAI_HEIGHT;            //炮台高度
19      Bitmap paotongBitmap;                                     //炮筒位图
20      Bitmap paotaiBitmap;                                      //炮台位图
21      Bitmap bulletBitmap;                                      //炮弹位图
22      DaPao(GameView gameView,Bitmap paotongBitmap,Bitmap paotaiBitmap,Bitmap
bulletBitmap){
23          this.gameView=gameView;                               //给成员变量赋值
24          this.paotongBitmap=paotongBitmap;
25          this.paotaiBitmap=paotaiBitmap;
26          this.bulletBitmap=bulletBitmap;
27      }
28      public void drawSelf(Canvas canvas,Paint paint){          //绘制大炮的方法
29      ……//此处省略了方法体中的代码，将在下一小节中给出
30      }
31      public Bullet sendBullet(){                               //大炮发射炮弹的方法
32      ……//此处省略了方法体中的代码，将在下一小节中给出
33      }
34      public void moveLeftSlowly(){                             //大炮慢速左移的方法
35          if(paotaiX>=0){
36              x-=xspanSlow;
37          }
38      }
39      public void moveLeftFast(){                               //大炮快速左移的方法
40          if(paotaiX>=0){
41              x-=xspanFast;
42          }
43      }
44      public void moveRightSlowly(){                            //大炮慢速右移的方法
45          if(paotaiX<=Constant.SCREEN_WIDTH-paotaiWidth){
46              x+=xspanSlow;
47          }
48      }
49      public void moveRightFast(){                              //大炮快速右移的方法
50          if(paotaiX<=Constant.SCREEN_WIDTH-paotaiWidth){
51              x+=xspanFast;
52          }
53      }
54      public float calcuateAngle(float pressX,float pressY){    //计算仰角的方法
```

```
55          ……//此处省略了方法体中的代码，将在下一小节中给出
56          }
57  }
```

其中：

- 代码第 6～21 行为大炮成员变量的定义，包括了大炮所有的属性，其中 centerRatio 的含义为炮筒旋转的中心点占整个炮筒宽度的比例。
- 代码第 34～53 行为整个大炮左右移动的四个成员方法。前面章节中已经用到了这四个方法，即按键时间短时调用慢速移动的方法，按键时间长时调用快速移动的方法。

说明：代码框架中的部分方法为空实现，部分方法的实现将在下一小节中详细介绍。

实例 18　英雄大炮类成员方法的实现

【实现过程】

上一小节中介绍了 DaPao 类的代码框架，本小节将介绍其中空实现的成员方法。下面按代码的开发顺序进行介绍。

【代码解析】

（1）drawSelf 方法的实现。该方法分炮筒和炮台两部分来绘制大炮，其代码如下。将下列代码插入到 DaPao 类的第 28 行。

代码位置：见光盘源代码/第 13 章/FKDP/src/com/bn/fkdp 目录下的 DaPao.java。

```
1   public void drawSelf(Canvas canvas,Paint paint){
2           Matrix m1=new Matrix();                              //平移矩阵
3           //调整后的 x,y 坐标
4           float xtemp=
    (float) (x-paotongHeight/2.0f*Math.sin(gameView.angle)+center*(1-Math.cos(game
View.angle)));
5           float ytemp=
    (float) (y+paotongHeight/2.0f*(1-Math.cos(gameView.angle))+center*Math.sin(game
View.angle));
6           m1.setTranslate(xtemp,ytemp);
7           Matrix m2=new Matrix();                              //旋转矩阵
8           m2.setRotate((float) (-Math.toDegrees(gameView.angle)));
9           Matrix mz=new Matrix();
10          mz.setConcat(m1, m2);
11          canvas.drawBitmap(paotongBitmap, mz,paint);          //绘制炮筒
12          paotaiX=x+center-paotaiWidth/2.0f;                   //计算炮台位置
13          paotaiY=y+12;
14          canvas.drawBitmap(paotaiBitmap, paotaiX, paotaiY,paint);//绘制炮台
15      }
```

其中：

- 代码第 2～11 行为绘制炮筒的代码，炮筒的平移和旋转是用矩阵来实现的。在绘制之前要先算好绕旋转中心点旋转 angle 度之后炮筒左上顶点的坐标。
- 代码第 12～14 行为绘制炮台的代码，由于炮台并不旋转，因此炮台的绘制比较简单，只要按照炮台的尺寸，算出其左上顶点的坐标即可。

（2）sendBullet 方法的实现。该方法按当前炮筒的位置计算出炮弹的位置，创建并返回一

个 Bullet 对象，其代码如下。将下列代码插入到 DaPao 类的第 31 行。

代码位置：见光盘源代码/第 13 章/FKDP/src/com/bn/fkdp 目录下的 DaPao.java。

```
1   public Bullet sendBullet(){                          //大炮发射炮弹的方法
2           Bullet result=null;
3           if(gameView.activity.isSoundOn()){           //播放发射炮弹的声音
4               gameView.playSound(1, 0);
5           }
6           float v=gameView.strengthBar.getWidth()*0.12f;
                                                         //将力度条的宽度转换成炮弹初速度
7           float angle=gameView.angle;                  //仰角
8           float r=Constant.BULLET_SIZE/2.0f;           //炮弹半径
9           float w=paotongWidth+r;                      //筒宽
10          float h=paotongHeight;                       //筒高
11          float bx=(float) (x+center+(w-center)*Math.cos(gameView.angle)-r);
                                                         //炮弹的初始位置
12          float by=(float) (y+h/2-(w-center)*Math.sin(gameView.angle)-r);
13          result=new Bullet(                           //创建炮弹对象
14                  gameView,bulletBitmap,bx,by,
(float) (v*Math.cos(angle)),-1*(float) (v*Math.sin(angle)));
15          return result;                               //返回创建的炮弹对象
16      }
```

其中：

- 代码第 3～5 行为发射炮弹时调用 GameView 类中播放发射炮弹声音的方法，其中参数 1 代表播放声音的种类，0 代表不循环播放。
- 代码第 6～14 行为根据力度条的宽度和大炮炮筒的各参数计算出炮弹的初速度、初始位置等变量，并创建相应的炮弹对象。

（3）calcuateAngle 方法的实现。该方法主要负责根据触控位置计算炮筒的仰角，其代码如下。将下列代码插入到 DaPao 类的第 31 行。

代码位置：见光盘源代码/第 13 章/FKDP/src/com/bn/fkdp 目录下的 DaPao.java。

```
1   public float calcuateAngle(float pressX,float pressY){   //计算仰角的方法
2           float result=0;
3           float centerX=x+center;                          //旋转中心点位置
4           float centerY=y+paotongHeight/2;
5           if(pressX>centerX){                              //单击点在炮筒右边
6               result=(float) Math.atan(Math.abs((centerY-pressY)/(pressX-
centerX)));
7           }
8           else if(pressX==centerX){                        //单击点在炮筒正上方
9               result=(float) (Math.PI/2.0f);
10          }
11          else{                                            //单击点在炮筒左边
12              result=(float) (Math.atan(Math.abs((pressX-centerX)/(centerY-
pressY)))+Math.PI/2.0f);
13          }
14          return result;                                   //返回计算出的结果
15      }
```

其中：

- 代码第 1 行中方法的参数 pressX、 pressY 含意为触控点的位置坐标。
- 代码第 3～13 行为计算炮筒仰角的代码，由于炮筒向左右和正上方三个方向都可以转动，故要分为三种情况分别计算其仰角。

实例 19　炮弹的实现

【实现过程】

Bullet 类是大炮发射的炮弹类，该类实现起来比较简单，主要是成员变量的赋值和炮弹的前进以及与目标的碰撞检测，其代码如下。

【代码解析】

代码位置：见光盘源代码/第 13 章/FKDP/src/com/bn/fkdp 目录下的 Bullet.java。

```
1   package com.bn.fkdp;                                          //声明包
2   import java.util.List;                                        //引入相关类
3   ……//此处省略了部分类的引入代码，读者可自行查看随书光盘的源代码
4   import android.graphics.Paint;                                //引入相关类
5   public class Bullet {
6       GameView gameView;
7       private Bitmap bitmap;                                    //位图
8       float x;                                                  //x 方向位移
9       float y;                                                  //y 方向位移
10      float vx;                                                 //x 方向速度
11      float vy;                                                 //y 方向速度
12      private float t=0;                                        //时间
13      private float timeSpan=1;
14      private final float g=1.0f;                               //重力加速度
15      final int size=Constant.BULLET_SIZE;                      //炮弹尺寸
16      public Bullet(GameView gameView,Bitmap bitmap,float x,float y,float
vx,float vy){
17          this.gameView=gameView;                               //成员变量赋值
18          this.bitmap=bitmap;
19          this.x=x;
20          this.y=y;                                             //成员变量赋值
21          this.vx=vx;
22          this.vy=vy;
23      }
24      public void drawSelf(Canvas canvas,Paint paint){     //绘制炮弹的方法
25          go();
26          canvas.drawBitmap(bitmap, x, y,paint);
27      }
28      public void go(){                                         //炮弹前进的方法
29          x+=vx*t;                                              //水平方向匀速直线运动
30          y+=vy*t+0.5f*g*t*t;                                   //竖直方向匀加速直线运动
31          List<Target> alTarget1=new Vector<Target>(gameView.alTarget);//复制
炮弹列表
32          for(Target t:alTarget1){
33              if(Constant.IsTwoRectCross(x, y, size, size, t.x, t.y, t.width,
t.height)){//如果击中目标
34                  t.explode();                                  //目标爆炸
35                  return;
36              }
37          }
38          if(x>=Constant.SCREEN_WIDTH||y>=Constant.SCREEN_HEIGHT){//如果炮弹出
了屏幕
39              this.disapear();                                  //炮弹消失
40              return;
```

```
41            }
42            t+=timeSpan;                                    //时间间隔
43        }
44        public void disapear(){                             //炮弹消失的方法
45            gameView.alBullet.remove(this);                 //在炮弹列表中删除该炮弹
46    }}
```

其中：

- 代码第 6～23 行为成员变量及构造方法的定义，各个炮弹对象的初始位置和初速度的不同正是因为创建对象时传递了不同的参数。
- 代码第 28～43 行为炮弹前进的方法。炮弹的运动轨迹是抛物线，水平方向匀速直线运动，竖直方向是加速度为 g 的匀加速直线运动。当运动到某个位置时判断是否与目标碰撞，如果发生碰撞，调用目标爆炸的方法即可达到射击目标的目的。

实例 20　目标的实现

【实现过程】

Target 类是大炮射击的目标类，其中目标按一定轨迹运动的实现是目标类中最有技术含量也是最有难度的，请读者仔细分析，其代码如下。

【代码解析】

代码位置：见光盘源代码/第 13 章/FKDP/src/com/bn/fkdp 目录下的 Target.java。

```
1   package com.bn.fkdp;                                    //声明包
2   import android.graphics.Bitmap;                         //引入相关类
3   import android.graphics.Canvas;                         //引入相关类
4   import android.graphics.Paint;                          //引入相关类
5   public class Target {
6       GameView gameView;
7       Bitmap bitmap;                                      //用于绘制的图片
8       Bitmap[] bitmaps;                                   //所有目标图片
9       private Bitmap[] explodeBitmaps;                    //爆炸动画位图数组
10      float x;                                            //目标位置
11      float y;
12      final int width=Constant.TARGET_WIDTH;              //目标的宽度
13      final int height=Constant.TARGET_HEIGHT;            //目标的高度
14      float xspan=0;                                      //目标步进
15      float yspan=0;                                      //目标步进
16      int type;                                           //目标的类型
17      int begin;                                          //当前路径起始点下标
18      int destination;                                    //目的地的下标
19      float[][] path;                                     //路径
20      public Target(GameView gameView,Bitmap[] bitmaps,int type,float[][]
path){      //构造器
21          this.gameView=gameView;
22          this.bitmaps=bitmaps;
23          this.explodeBitmaps=gameView.explodeBitmapArray;
24          this.type=type;
25          this.path=path;
26          this.x=path[0][0];                              //获取路径中第一个点的坐标
27          this.y=path[0][1];
28          bitmap=bitmaps[type];                           //根据类型分配不同的位图
```

```
29          begin=0;                                          //当前路径起始点下标设为 0
30          destination=begin+1;                              //目的地的下标比起始点的大 1
31      }
32      public void drawSelf(Canvas canvas,Paint paint){     //绘制目标的方法
33          go();
34          canvas.drawBitmap(bitmap, x, y,paint);
35      }
36      public void go(){
37          caculateCurrentXYSpan();                          //计算当前步进
38          x+=xspan;                                         //移动目标
39          y+=yspan;
40          if(!this.isInScreen()){                           //如果目标出了屏幕
41              this.disappear();                             //目标消失
42          }
43      }
44      public void caculateCurrentXYSpan(){                  //计算当前步进
45          if(x==path[destination][0]&&y==path[destination][1]){//如果到达目的
地，继续去往下一个目的地
46              begin=(begin+1)%path.length;
47              destination=(destination+1)%path.length;
48          }
49          xspan=(path[destination][0]-path[begin][0])/path[begin][2];
                                                    //通过当前路径算出 x,y 方向步进
50          yspan=(path[destination][1]-path[begin][1])/path[begin][2];
51      }
52      public void explode(){                                //目标爆炸的方法
53          if(gameView.activity.isSoundOn()){                //播放目标爆炸的声音
54              gameView.playSound(2, 0);
55          }
56          gameView.alTarget.remove(this);                   //从列表中删除目标
57         gameView.alExplosion.add(new Explosion(gameView,explodeBitmaps,x,y));
                                                              //列表中添加爆炸
58          gameView.score.addScore(1);                       //得分加 1
59      }
60      public void disappear(){                              //目标消失的方法
61          gameView.alTarget.remove(this);
62      }
63      public boolean isInScreen(){                          //判断目标是否在屏幕内
64          if(Constant.IsTwoRectCross(                       //如果目标在屏幕里
65  x, y, width, height, 0, 0, Constant.SCREEN_WIDTH, Constant.SCREEN_HEIGHT)) {
66              return true;
67          }
68          return false;
69  }}
```

其中：

- 代码第 20～31 行为构造器，其中第 26、27 行是从 path 数组中获取路径中第一个点的坐标。begin 和 destination 分别为目标当前路径起点和终点的位置在 path 数组中的下标。
- 代码第 36～43 行为目标前进的方法，在更改目标的位置之前先调用了 caculateCurrentXYSpan 方法计算当前目标 x、y 方向上的步进，之后又判断目标是否出了屏幕。
- 代码第 44～51 行为计算当前目标 x、y 方向上的步进的方法，其思想是用起点和终点的距离差值除以经过两点所用的时间。
- 代码第 52～59 行为目标爆炸的方法，目标爆炸与目标消失的方法不同之处在于爆炸要播放爆炸的声音和爆炸动画。爆炸动画的实现也很简单，只要创建一个爆炸效果对象并加入爆炸效果列表中即可，至于爆炸动画是如何形成的，将在下一小节中介绍。

实例 21　爆炸效果的实现

【实现过程】

Explosion 类的代码相对简单，但其中思想并不容易掌握，如果单独考查代码并不容易明白爆炸的原理，本小节将对其代码进行详细讲解。Explosion 类的代码如下。

【代码解析】

代码位置：见光盘源代码/第 13 章/FKDP/src/com/bn/fkdp 目录下的 Explosion.java。

```
1   package com.bn.fkdp;                                          //声明包
2   import android.graphics.Bitmap;                               //引入相关类
3   import android.graphics.Canvas;                               //引入相关类
4   import android.graphics.Paint;                                //引入相关类
5   public class Explosion {
6       GameView gameView;
7       private Bitmap[] bitmaps;                                 //爆炸位图数组
8       float x;                                                  //x 方向位移
9       float y;                                                  //y 方向位移
10      private int anmiIndex=0;                                  //爆炸动画帧索引
11      public Explosion(GameView gameView,Bitmap[] bitmaps,float x,float y){
12          this.gameView=gameView;                               //成员变量赋值
13          this.bitmaps=bitmaps;
14          this.x=x;
15          this.y=y;
16      }
17      public void drawSelf(Canvas canvas,Paint paint){          //绘制爆炸动画的方法
18          canvas.drawBitmap(bitmaps[anmiIndex], x, y,paint);
19          if(anmiIndex<bitmaps.length-1){                       //如果没有播放完
20              anmiIndex++;
21          }
22          else{                                                 //如果爆炸动画完毕
23              gameView.alExplosion.remove(this);                //从列表中删除此爆炸
24  }}}
```

其中：

- 代码第 6～10 行为成员变量的定义。可以看出，每个爆炸效果对象都有一个位图数组，而在同一时刻只有一个下标为 anmiIndex 的位图被绘制。
- 代码第 17～24 行为绘制爆炸动画的方法。绘制完某一帧后，判断当前绘制的位图的下标是否超过了数组中最后一个位图的下标，若没有超过，将 anmiIndex 自增 1，当再次调用 drawSelf 方法时绘制的就是下一张图；若已经绘制到最后一张图，动画播放完毕，从列表中删除此爆炸。

实例 22　飞行器及其子类的实现

【实现过程】

AirCraft 类及其子类是游戏界面中飞机、飞碟、飞鱼和超人等第二层物体。由于飞行器不会被炮弹击中，只是在游戏界面中绘制，因此其实现非常简单，本小节仅以飞碟类为例进行讲

解，其代码如下。

【代码解析】

代码位置：见光盘源代码/第 13 章/FKDP/src/com/bn/fkdp 目录下的 AirCraft.java。

```
1   package com.bn.fkdp;                                          //声明包
2   import android.graphics.Bitmap;                               //引入相关类
3   ……//此处省略了部分类的引入代码，读者可自行查看随书光盘的源代码
4   import android.graphics.Paint;                                //引入相关类
5   public abstract class AirCraft  {                             //抽象类飞行器
6       float x;                                                  //飞行器的位置
7       float y;
8       float xspan;                                              //飞行器的速度
9       float yspan;
10      Bitmap bitmap;                                            //飞行器的位图
11      abstract void go();                                       //抽象类向前飞行的抽象方法
12      void drawSelf(Canvas canvas,Paint paint){                 //绘制飞行器的方法
13          canvas.drawBitmap(bitmap, x, y, paint);
14      }
15  }
16  class UFO extends AirCraft{                                   //飞碟类
17      UFO(Bitmap bitmap){                                       //构造方法
18          x=2*Constant.SCREEN_WIDTH;                            //成员变量赋值
19          y=0;
20          xspan=9;                                              //成员变量赋值
21          yspan=2;
22          this.bitmap=bitmap;                                   //成员变量赋值
23      }
24      void go() {                                               //实现父类的 go 方法
25          if(x<-2*Constant.SCREEN_WIDTH){                       //当飞碟飞行超过一定距离后
26              x=2*Constant.SCREEN_WIDTH;                        //重新回到原来的位置
27              y=0;
28          }
29          x-=xspan;                                             //改变飞碟的位置
30          y+=yspan;
31      }
32      void drawSelf(Canvas canvas,Paint paint){                 //重写父类的 drawSelf 方法
33          float ratio=(x-40.0f)/440.0f;                       //在屏幕中出现时， ratio 应为 1
34          if(ratio<=0.03){                                      //如果比例太小，则不再绘制
35              return;
36          }
37          Matrix m1=new Matrix();                               //移动矩阵
38          m1.setTranslate(x, y);                                //移动到的位置
39          Matrix m2=new Matrix();                               //缩放矩阵
40          m2.setScale(ratio,ratio);                             //缩放的比例
41          Matrix m3=new Matrix();
42          m3.setConcat(m1, m2);                                 //矩阵相乘
43          canvas.drawBitmap(bitmap, m3,paint);                  //绘制变化后的飞碟位图
44  }}
```

其中：

- 代码第 5～15 行为抽象类 AirCraft，将其定义为抽象类的原因是不用创建该类的对象，其作用只是为了让飞行物实体类继承该类来创建具体的飞行物对象。
- 代码第 16～44 行为继承了抽象类 AirCraft 并实现了抽象方法 go 的 UFO 类。该类重写了父类的 drawSelf 方法，实现了自己特殊的绘制方法，飞碟越飞越小是靠矩阵来实现的。

实例 23　飞行物的实现

【实现过程】

FlyingThings 类的主要作用是将所有飞行物统一地管理起来，这样在 GameView 中只需要创建该类的对象，就实现了所有飞行物的创建，其代码如下。

【代码解析】

代码位置：见光盘源代码/第 13 章/FKDP/src/com/bn/fkdp 目录下的 FlyingThings.java。

```
1   package com.bn.fkdp;                                          //声明包
2   import android.graphics.Bitmap;                               //引入相关类
3   import android.graphics.Canvas;                               //引入相关类
4   import android.graphics.Paint;                                //引入相关类
5   public class FlyingThings {
6       AirCraft[] airCrafts;                                     //管理所有飞行物的数组
7       public FlyingThings(Bitmap[] bitmaps){                    //构造器
8           airCrafts=new AirCraft[]{                             //创建所有飞行器对象
9               new Plane(bitmaps[0]),                            //飞机
10              new UFO(bitmaps[1]),                              //飞碟
11              new Superman(bitmaps[2]),                         //超人
12              new Fish(bitmaps[3])                              //飞鱼
13          };
14      }
15      public void drawSelf(Canvas canvas,Paint paint){          //绘制所有飞行物的方法
16          for(AirCraft ac:airCrafts){                           //遍历所有飞行物的数组
17              ac.go();                                  //调用各飞行物的 go 方法
18              ac.drawSelf(canvas,paint);                //调用各飞行物的 drawSelf 方法
19  }}}
```

其中：

- 代码第 7～14 行为构造器的代码，在构造器参数中传入各飞行器的位图数组，构造器方法体中创建了各个飞行物的对象并存入 airCrafts 数组中。
- 代码第 15～19 行为绘制所有飞行物的方法，该方法的实现非常简单，只要遍历所有飞行物的数组，并调用各个类的 go 方法和 drawSelf 方法即可实现各飞行物的移动和绘制。

实例 24　力度条的实现

【实例描述】

上一节介绍了情景相关类的实现，本节开始介绍最后一部分，即自定义控件及工具类的实现。其中包括虚拟按钮、得分、定时器、背景、力度条以及获取系统日期的工具类，下面对各类进行详细介绍。

【实现过程】

StrengthBar 类的作用是显示炮弹发射时的初速度，其分为两层，底层是蓝色半透明条，用于显示最大力度，上层是几乎不透明的条，用于显示当前力度，其代码如下。

【代码解析】

代码位置：见光盘源代码/第 13 章/FKDP/src/com/bn/fkdp 目录下的 StrengthBar.java。

```
1   package com.bn.fkdp;                                          //声明包
2   import android.graphics.Bitmap;                               //引入相关类
3   import android.graphics.Canvas;                               //引入相关类
4   import android.graphics.Paint;                                //引入相关类
5   public class StrengthBar {
6       GameView gameView;
7       final float space=3;                                      //文字和力度条的距离
8       //关于力度条图的量
9       private Bitmap downBitmap;                                //位图
10      private Bitmap upBitmap;
11      final float x=Constant.BAR_X;                             //力度条的位置
12      final float y=Constant.BAR_Y;
13      final float spanX=1;                                      //力度条的步进
14      final float maxWidth=Constant.BAR_MAX_WIDTH;              //力度条总宽度
15      final float height=Constant.SCREEN_HEIGHT;                //力度条高度
16      private float width;                                      //力度条当前宽度
17      //关于文字图的量
18      private Bitmap textBitmap;
19      final float textWidth=Constant.TEXT_WIDTH;                //文字图的宽度
20      final float textHeight=Constant.TEXT_HEIGHT;              //文字图的高度
21      final float textX=x-space-textWidth;                      //字的位置坐标
22      final float textY=y;
23      public StrengthBar(GameView gameView,Bitmap downBitmap,Bitmap upBitmap,
Bitmap textBitmap){
24          this.gameView=gameView;                               //成员变量赋值
25          this.downBitmap=downBitmap;
26          this.upBitmap=upBitmap;                               //成员变量赋值
27          this.textBitmap=textBitmap;
28      }
29      public void drawSelf(Canvas canvas,Paint paint){          //绘制力度条的方法
30          canvas.drawBitmap(downBitmap, x, y,paint);            //绘制底下的条
31          canvas.save();
32          canvas.clipRect(x, y, x+width, y+height);             //限制绘制的范围
33          canvas.drawBitmap(upBitmap, x, y,paint);              //绘制上面的条
34          canvas.restore();
35          canvas.drawBitmap(textBitmap, textX, textY,paint);    //绘制文字图
36      }
37      public float getWidth() {                                 //得到力度条当前宽度的方法
38          return width;
39      }
40      public void setWidth(float width) {                       //设置力度条宽度的方法
41          this.width = width;
42      }
43      public void addWidth(float width) {                       //增加力度条宽度的方法
44          if(width<=maxWidth)   {
45              this.width+=width;
46  }}}
```

其中：

- 代码第 6～22 行为成员变量的定义，成员变量分为关于力度条图片的变量和关于力度文字的变量，包括位置、尺寸、位图等。
- 代码第 29～36 行为绘制力度条的方法。其中绘制力度条的思想是，将图片在指定的矩形区域内绘制，再动态改变指定区域的大小即可。

实例 25　定时器的实现

【实现过程】

Timer 类的主要功能是游戏倒计时，并将时间以图片的形式显示在游戏界面中。当时间为 0 时游戏结束，其代码如下。

【代码解析】

代码位置：见光盘源代码/第 13 章/FKDP/src/com/bn/fkdp 目录下的 Timer.java。

```
1   package com.bn.fkdp;                                          //声明包
2   import android.graphics.Bitmap;                               //引入相关类
3   import android.graphics.Canvas;                               //引入相关类
4   import android.graphics.Paint;                                //引入相关类
5   public class Timer {
6       GameView gameView;
7       private final int totalSecond=60;                         //总秒数
8       private Bitmap breakMarkBitmap;                           //时间分隔符的位图
9       private Bitmap[] numberBitmaps;                           //数字位图
10      private int leftSecond=totalSecond;
11      int endX=Constant.SCREEN_WIDTH*2/3;                       //数字的右上点坐标
12      int endY=(int) Constant.BAR_Y;
13      int numberWidth;
14      int numberHeight;
15      int breakMarkWidth;
16      int breakMarkHeight;
17      public Timer(GameView gameView,Bitmap breakMarkBitmap,Bitmap[] numberBitmaps){//构造器
18          this.gameView=gameView;                               //成员变量赋值
19          this.breakMarkBitmap=breakMarkBitmap;
20          this.numberBitmaps=numberBitmaps;
21          numberWidth=numberBitmaps[0].getWidth();              //成员变量赋值
22          numberHeight=numberBitmaps[0].getHeight();
23          breakMarkWidth=breakMarkBitmap.getWidth();            //成员变量赋值
24          breakMarkHeight=breakMarkBitmap.getHeight();
25      }
26      public void drawSelf(Canvas canvas,Paint paint){          //绘制时间的方法
27          int second=this.leftSecond%60;                        //得到秒钟
28          int minute=this.leftSecond/60;                        //得到分钟
29          drawNumberBitmap(second,numberBitmaps,endX,endY,canvas, paint);
                                                                  //绘制秒钟
30          int secondLength=(second+"").length()<=1 ? (second+"").length()+1 :
(second+"").length();
31          int breakMarkX=endX-secondLength*numberWidth-breakMarkWidth;
                                                                  //秒钟的位置
32          int breakMarkY=endY;
33          canvas.drawBitmap(breakMarkBitmap, breakMarkX, breakMarkY,paint);
                                                                  //绘制时间分隔符图片
34          int miniteEndX=breakMarkX;                            //计算分钟图片的位置
35          int miniteEndY=endY;
36          drawNumberBitmap(minute,numberBitmaps,miniteEndX,miniteEndY,canvas,
paint);//绘制分钟
37      }
38      public void subtractTime(int second){                     //减少时间的方法
39          if(this.leftSecond>0){
```

```
40                this.leftSecond-=second;
41            }else{                                          //如果时间为 0，结束游戏
42                gameView.overGame();
43            }
44        }
45        public void drawNumberBitmap(
int number,Bitmap[] numberBitmaps,float endX,float endY,Canvas canvas,Paint paint)
46        {                                                   //画数字图片的方法
47            String numberStr=number+"";                     //将数字转化成字符串
48            if(number<10){                                  //保证至少有两位
49                numberStr="0"+numberStr;
50            }
51            for(int i=0;i<numberStr.length();i++){          //循环绘制每个字符
52                char c=numberStr.charAt(i);                 //得到其中一个字符
53                canvas.drawBitmap(
54                        numberBitmaps[c-'0'],               //位图
55                        endX-numberWidth*(numberStr.length()-i),    //位置
56                        endY,
57                        paint                               //画笔
58                );
59    }}}
```

其中：

- 代码第 26～37 行为绘制时间的方法。由于要绘制的时间为总秒数，因此要将其换算成分钟数和秒数，再对分钟和秒钟以及中间的分隔符分别进行绘制。
- 代码第 38～44 行为倒计时的方法，循环调用此方法便可实现时间的倒计时，当时间为 0 时调用 GameView 中游戏结束的方法。

实例 26　得分榜的实现

【实现过程】

Score 类的实现和 Timer 类非常相似，也是将数字转换成图片后在游戏界面中显示，改变得分时只需要在其他地方调用该类的对应方法即可，其代码如下。

【代码解析】

代码位置：见光盘源代码/第 13 章/FKDP/src/com/bn/fkdp 目录下的 Score.java。

```
1   package com.bn.fkdp;                                    //声明包
2   import android.graphics.Bitmap;                         //引入相关类
3   import android.graphics.Canvas;                         //引入相关类
4   import android.graphics.Paint;                          //引入相关类
5   public class Score {
6       GameView gameView;
7       final int space=5;                                  //文字和力度条的距离
8       private Bitmap defenBitmap;                         //得分位图
9       private Bitmap[] numberBitmaps;                     //数字位图
10      private int score=0;                                //得分
11      int endX=Constant.SCREEN_WIDTH;                     //数字的右上点坐标
12      int endY=(int) Constant.BAR_Y;
13      int numberWidth;                                    //数字图片的宽度
14      int numberHeight;                                   //数字图片的高度
15      int textWidth;                                      //文字图片的宽度
```

```
    int textHeight;                                          //文字图片的高度
    public Score(GameView gameView,Bitmap defenBitmap,Bitmap[] numberBitmaps)
{//构造器
        this.gameView=gameView;                              //成员变量的赋值
        this.defenBitmap=defenBitmap;
        this.numberBitmaps=numberBitmaps;
        numberHeight=numberBitmaps[0].getWidth();            //成员变量的赋值
        numberWidth=numberBitmaps[0].getHeight();
        textWidth=defenBitmap.getWidth();                    //成员变量的赋值
        textHeight=defenBitmap.getHeight();
    }
    public void drawSelf(Canvas canvas,Paint paint) {        //绘制得分的方法
        drawNumberBitmap(this.score,numberBitmaps,endX,endY,canvas, paint);
//绘制数字
        String numberStr=this.score+"";                      //将得分转换成字符串
        int textX=endX-numberWidth*numberStr.length()-textWidth-space;
//计算字体图片的位置
        int textY=endY;
        canvas.drawBitmap(defenBitmap, textX, textY,paint); //绘制得分图片
    }
    public void addScore(int score){                         //增加得分的方法
        this.score+=score;
    }
    public int getScore(){                                   //获得得分的方法
        return score;
    }
    public void drawNumberBitmap(
int number,Bitmap[] numberBitmaps,float endX,float endY,Canvas canvas,Paint paint)
    {                                                        //画数字图片的方法
        String numberStr=number+"";                          //将数字转换成字符串
        for(int i=0;i<numberStr.length();i++){               //循环绘制每个字符
            char c=numberStr.charAt(i);                      //得到其中一个字符
            canvas.drawBitmap(
                    numberBitmaps[c-'0'],                    //位图
                    endX-numberWidth*(numberStr.length()-i),    //位置
                    endY,
                    paint                                    //画笔
            );
}}}
```

其中：

- 代码第 26～32 行为绘制得分的方法。绘制得分分为得分文字和数字图片两部分，其中绘制数字的方法在前面章节中已有介绍，请读者自行查看分析。
- 代码第 33～38 行为增加得分与获取得分的方法，增加得分的方法是在目标被击中时被调用的，而获得得分的方法是在将分数写入数据库中时被调用的。

实例 27　滚屏背景的实现

【实现过程】

BackGround 类是游戏背景滚屏技术的实现类，游戏界面中的蓝天、白云和草地是在向左慢慢移动的，使游戏界面动态性增强，其代码如下。

【代码解析】

代码位置：见光盘源代码/第 13 章/FKDP/src/com/bn/fkdp 目录下的 Score.java。

```
1   package com.bn.fkdp;                                          //声明包
2   import android.graphics.Bitmap;                               //引入相关类
3   import android.graphics.Canvas;                               //引入相关类
4   import android.graphics.Paint;                                //引入相关类
5   public class Background {
6       GameView gameView;
7       private Bitmap bitmap;                                    //位图
8       float x=0;                                                //x 方向位移
9       float y=0;                                                //y 方向位移
10      float spanX=1;                                            //x 方向上滚屏速度
11      final int bitmapWidth=Constant.BACKGROUND_WIDTH;          //背景图宽度
12      final int bitmapHeight=Constant.BACKGROUND_HEIGHT;        //背景图高度
13      final int width=Constant.SCREEN_WIDTH;                    //背景界面的宽度
14      final int height=Constant.SCREEN_HEIGHT;                  //背景界面的高度
15      public Background(Bitmap bitmap){                         //构造器
16          this.bitmap=bitmap;
17      }
18      public void drawSelf(Canvas canvas,Paint paint){          //绘制背景的方法
19          go();
20          canvas.drawBitmap(bitmap, x, y,paint);
21      }
22      public void go(){                                         //滚屏的方法
23          if(Math.abs(x)>bitmapWidth-width){                  //如果屏滚到头,从头开始滚
24              x=0;
25          }
26          x-=spanX;                                             //改变坐标位置
27  }}
```

其中：

- 代码第 5～14 行为成员变量的定义。要特别注意的是背景图片 bitmap 的宽度要大于屏幕的宽度，而且背景图片是用软件特殊处理过的，使图片首尾相接，这样才能达到最佳的滚屏效果。
- 代码第 22～26 行为滚屏的方法。当屏幕滚到背景图片最右边时，将坐标 x 值重新置为 0，从屏幕的最左端开始滚屏，当转到最左端时坐标的位置仍要改变。

实例 28　主菜单按钮的实现

【实现过程】

MainMenuButton 类为主菜单中的虚拟按钮控件，该类只负责按钮按下和抬起时图片的切换，按钮负责的动作其实是在该类外适当的位置被执行的。MainMenuButton 类的代码如下。

【代码解析】

代码位置：见光盘源代码/第 13 章/FKDP/src/com/bn/fkdp 目录下的 MainMenuButton.java。

```
1   package com.bn.fkdp;                                          //声明包
2   import android.graphics.Bitmap;                               //引入相关类
3   import android.graphics.Canvas;                               //引入相关类
4   import android.graphics.Paint;                                //引入相关类
```

```
5   public class MainMenuButton {
6       int x;                                                    //按钮位置
7       int y;
8       int width;                                                //按钮宽度
9       int height;                                               //按钮高度
10      Bitmap onBitmap;                                          //铵钮位图
11      Bitmap offBitmap;
12      boolean isOn=false;                                       //按钮是否被按下的标志
13      public MainMenuButton(Bitmap onBitmap,Bitmap offBitmap,int x,int y){
                                                                  //构造器
14          this.onBitmap=onBitmap;                               //成员变量赋值
15          this.offBitmap=offBitmap;
16          this.x=x;
17          this.y=y;                                             //成员变量赋值
18          this.width=offBitmap.getWidth();                      //成员变量赋值
19          this.height=offBitmap.getHeight();
20      }
21      public void drawSelf(Canvas canvas,Paint paint){          //绘制虚拟按钮的方法
22          if(isOn){                                             //如果按钮被按下
23              canvas.drawBitmap(onBitmap, x, y, paint);
24          }else{                                                //按钮没有被按下
25              canvas.drawBitmap(offBitmap, x, y, paint);
26          }
27      }
28      public void switchOn(){                                   //按下按钮的方法
29          isOn=true;
30      }
31      public void switchOff(){                                  //放开按钮的方法
32          isOn=false;
33      }
34      public boolean isActionOnButton(int pressX,int pressY){
                                                                  //判断是否有触控事件的方法
35          return Constant.isPointInRect(pressX, pressY, x, y, width, height);
36  }}
```

其中：

- 代码第 6～12 行为成员变量的定义。按钮的成员变量包括按钮图片、尺寸、位置以及按钮是否被按下的标志位等。
- 代码第 21～27 行为绘制按钮图片的方法，绘制按钮是根据 isOn 标志位，当标志位为按下时，绘制按下的按钮图片，反之则绘制未被按下的图片。
- 代码第 34～36 行为判断按钮上是否有触控事件发生的方法，方法的参数为屏幕触控的位置，如果触控的位置在虚拟按钮的区域内则判断为按下，返回 true，反之返回 false。

说明：SoundSwitchButton 类和 GameVirtualButton 类都是游戏中的虚拟按钮类，这两个类和 MainMenuButton 形式上大同小异，这里不再赘述，请读者自行查看随书光盘中的源代码。

实例 29　获取系统日期的方法

【实现过程】

DateUtil 类中只有一个静态方法 getCurrentDate，其主要负责获取当前系统时间，并以特定

格式的字符串的形式返回，其代码如下。

【代码解析】

代码位置：见光盘源代码/第 13 章/FKDP/src/com/bn/fkdp 目录下的 DateUtil.java。

```
1   package com.bn.fkdp;                                    //声明包
2   import java.util.Date;                                  //引入相关类
3   public class DateUtil{
4       public static String getCurrentDate(){             //得到当前日期的静态方法
5           String dateStr=null;                            //定义返回的结果
6           Date date=new Date();                           //创建日期对象
7           int year=date.getYear()+1900;                   //获得年份
8           int month=date.getMonth()+1;                    //获得月份
9           int dt=date.getDate();                          //获得日期
10          dateStr=year+"-"+((month<10)?"0"+
11                  month:month)+"-"+((dt<10)?"0"+dt:dt);
                                                            //将日期组织成特定格式的字符串
12          return dateStr;                                 //返回字符串
13  }}
```

提示：该方法非常简单，但又很实用，在需要获得系统时间的地方直接调用该方法即可获得特定格式的字符串，之后再将字符串解析成所需的格式并显示到游戏界面中即可。

实例 30　游戏的优化与改进

【实现过程】

到此为止，本游戏的功能开发完毕，但是还有很多地方值得优化与改进。下面提出几点可以改进的地方，有兴趣的读者可以对本游戏进行必要的提升。

- 游戏背景的多样化：到目前为止本游戏只有一个游戏场景，游戏显得比较单调，可以开发更多的场景，在开始界面中可以选择相应的场景，使游戏更加丰富。
- 目标种类的复杂化：本游戏的目标类只有三种，而飞行物等类不能被击中，如果再开发几种大炮的射击目标，各目标有自己的特殊路径，会使游戏的可玩性大大增强。
- 积分榜界面的优化：在游戏积分榜界面中实现了得分的上下翻页，但没有显示当前页的位置。有兴趣的读者可以在积分榜界面的适当位置加入页码或进度条，使其更加人性化。

第 14 章　娱乐游戏——3D 迷宫

在本章节中着重向读者介绍一款 3D 娱乐游戏的开发，本款游戏为 3D 迷宫寻宝。迷宫游戏一直都是娱乐性较强的游戏，在未知的道路中摸索前进，对前方路的新奇与担忧，给玩家带来无限的希冀与真实，因此其可以经久不衰。

实例 1　游戏背景及功能介绍

本节将对 3D 迷宫游戏整体上进行简单的介绍，使读者了解本游戏的开发背景以及对本游戏功能介绍，并了解在 Android 手机中本游戏的玩法。

【实例描述】

迷宫游戏是一款大众的娱乐性智力游戏，其适合于所有人群。既可以帮助小朋友开发智力，也可以带给成年人娱乐与惊奇。同时带给老年人对年轻时的美好回忆。

当然在行走迷宫的途中会有一些障碍物或者宝藏或者陷阱，正所谓惊险与机遇并存，迷宫游戏给玩家带来了新奇的体验。

【实现过程】

本游戏是一款娱乐性智力游戏，操作不是很难。如果玩家的设备支持键盘操控，则在游戏中玩家可以用键盘的上、下、左、右键控制第一人称视角人物的运动方向。当然玩家也可选择虚拟键操控本游戏。在本游戏的开发过程中，开发者充分考虑了游戏的可玩性。

玩家可以选择使用虚拟键操控，这时候屏幕上的虚拟摇杆可控制第一人称视角人物的前进、后退、左转和右转。在这样周到的考虑下，选择自己喜欢的模式，尽情地享受迷宫的惊奇吧！

实例 2　游戏实际预览效果

【实例描述】

本游戏的运行步骤如下。

（1）启动游戏后首先进入的是游戏的主菜单界面，效果如图 14-1 所示。其中包括开始游戏按钮、键盘模式按钮、虚拟键模式按钮和退出游戏按钮。单击“开始游戏”按钮则开启游戏，单击“退出游戏”按钮则退出游戏。

（2）在主界面中可以设置游戏的操控方式为键盘模式，效果如图 14-2 所示。当用户的手机支持键盘操控时，可选择该模式。

（3）在主界面中可以设置游戏的操控方式为虚拟键模式，效果如图 14-3 所示。当用户的手机不支持键盘操控时，可选择该模式，游戏默认操控模式为虚拟键模式。

（4）单击游戏开始按钮，则开启游戏，首先为游戏的加载，如图 14-4 所示，为游戏加载界

面图。

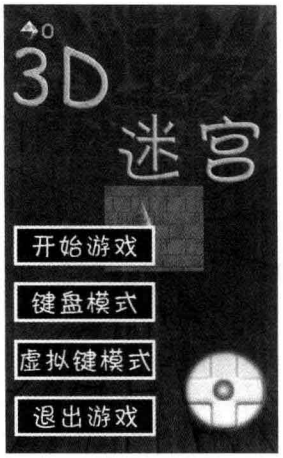
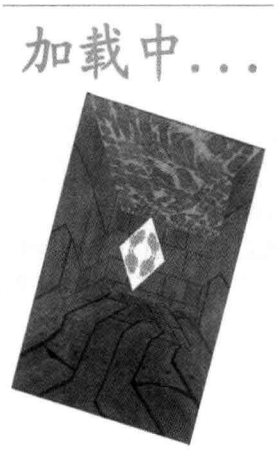
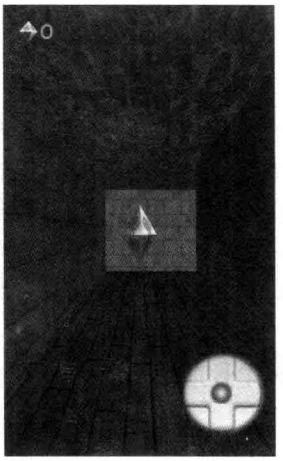


图 14-1　游戏开始主界面　　图 14-2　设置游戏模式为键盘模式图 14-3　设置游戏模式为虚拟键盘模式

（5）游戏加载完成后，正式进入游戏界面，如图 14-5 所示，为进入游戏的初始界面。在游戏界面中，左上侧为玩家当前所获得晶体的数量数值记录。右下侧为虚拟摇杆，用于控制人物的运动。

（6）在游戏中，存放一些漂浮在半空的并且自转运动的晶体，作为迷宫中的宝物，如图 14-6 所示。





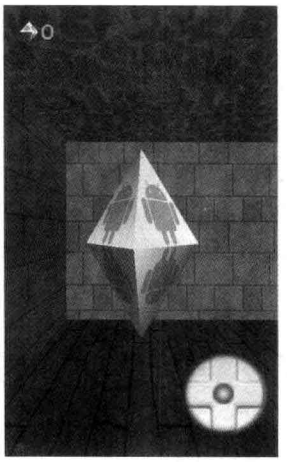


图 14-4　游戏加载界面　　图 14-5　游戏初始进入界面　　图 14-6　游戏中的晶体

（7）玩家可以通过虚拟摇杆来控制人物的运动，如图 14-7 所示，为通过虚拟摇杆的前进操作。

（8）玩家可以通过虚拟摇杆来控制人物的运动，如图 14-8 所示，为通过虚拟摇杆的后退操作。

（9）玩家可以通过虚拟摇杆来控制人物的运动，如图 14-9 所示，为通过虚拟摇杆的左转操作。

（10）玩家可以通过虚拟摇杆来控制人物的运动，如图 14-10 所示，为通过虚拟摇杆的右转操作。

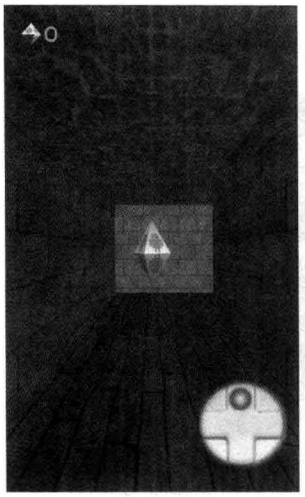


图 14-7　按前进按钮

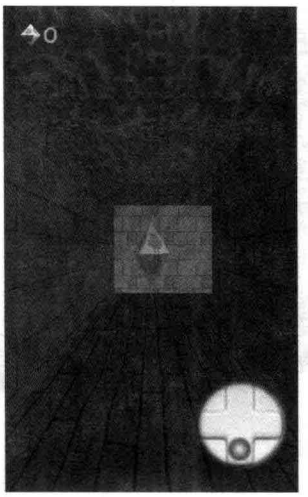


图 14-8　按后退按钮

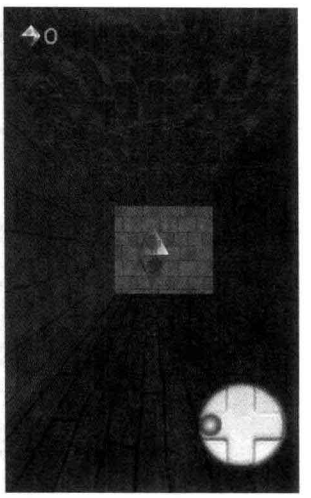


图 14-9　按左转按钮

（11）玩家可以通过触碰的方式来吃掉晶体，当吃掉晶体后，左上侧记录数值加 1，如图 14-11 所示。

（12）在游戏中设计了错综复杂的迷宫路线，但未知的新奇与惊喜隐藏在其中，很可能下一个路线就会通向宝藏，如图 14-12 所示。

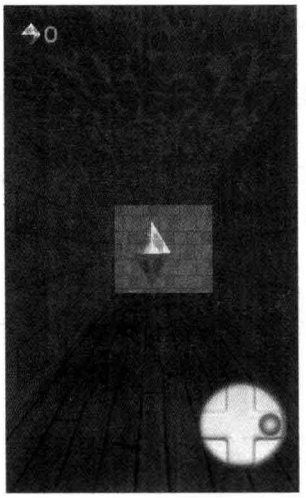


图 14-10　按右转按

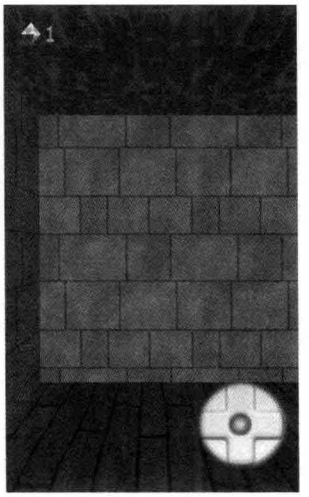


图 14-11　吃晶体计数加 1

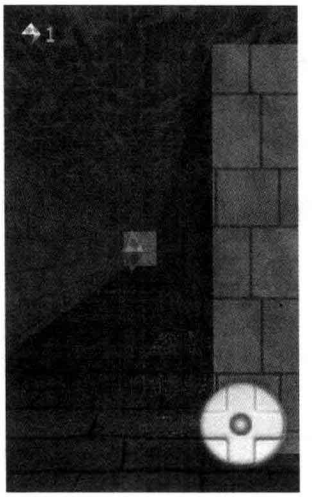


图 14-12　错综复杂的迷宫场景

（13）在本版游戏中仅设置了 6 枚晶体，如图 14-13 所示，为已经获取了 5 枚晶体，面前的为最后一枚晶体——第 6 枚晶体。

（14）当玩家获得全部的晶体时，标志着玩家成功闯关。如图 14-14 所示，为玩家胜利恭喜界面。

提示：本游戏的功能基本介绍完毕。比较遗憾的是本游戏只有一关。读者可在学习完本章内容后自行添加其他关节，以增加游戏难度，提高游戏的可玩性。

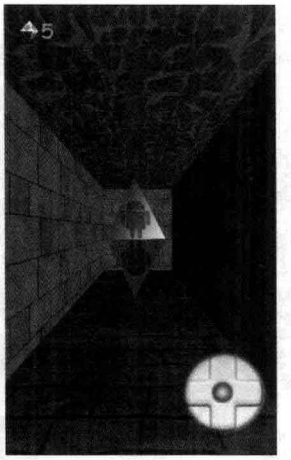


图 14-13　游戏最后一枚晶体

图 14-14　游戏过关界面

实例 3　游戏策划及准备工作

【实例描述】

本节主要介绍 3D 迷宫游戏的策划，以及游戏实现前的准备工作。在商业化游戏开发中，这些步骤还需要更细致、更具体以及更全面。

有了这款游戏，只要带着手机，玩家便可以随时随地进入虚拟的迷宫场景中，享受未知的惊奇，开启宝库的大门，满足玩家对游戏娱乐性的要求。

开发这个游戏的目的是为读者在 Android 平台上游戏开发提供一个指导方案，而不是生产商业化的游戏产品。读者可以以此为范例开发出更好、更具可玩性的 Android 游戏。

【实现过程】

游戏的准备工作通常是根据游戏的策划来制作游戏所用的图片、声音等。在本游戏中用到的图片文件资源如表 14-1 所列。

表 14-1　图片清单

图 片 名	大小 (KB)	像素 (w×h)	用　途	图 片 名	大小 (KB)	像素 (w×h)	用　途
zjm.jpg	46	480×800	主界面	biankuang.png	2.0	30×30	按钮黄色边框
bb.png	4	32×64	棱锥	ceil.bmp	49	128×128	迷宫顶部纹理
control.png	6	64×64	摇杆底图	floor.bmp	49	128×128	迷宫地板纹理
load.png	163	320×480	加载界面	number.png	10	128×32	数字
reddot.png	3	32×32	摇杆上侧显示红点	robot.png	3	128×128	晶体侧面纹理
wall.bmp	49	128×128	迷宫墙壁纹理	win.png	165	320×480	胜利界面图

提示：该图片的资源存放在第 14 章/GL_MASE/res/drawable-mpi 文件夹中。

在表 14-1 中主要介绍的是本游戏用到的图片的资源，接下来将要介绍在本游戏用到的声音文件资源，如表 14-2 所列。

表 14-2　声音清单

声 音 名	大小(KB)	用　途	声 音 名	大小(KB)	用　途
Gameback.mp3	215	背景音乐	gotobject.mp3	5	吃晶体音
win.mid	27	游戏胜利音效			

res 是存放所有非代码资源的文件夹，其下的 drawable 文件夹一般存放图片资源，raw 文件夹一般存放声音资源。由于从 1.6r2 开始 SDK 支持各种尺寸的屏幕，所以在 res 下有多个以 drawable 开头的文件夹，mdpi 为标准图库。本程序中所有的图片资源都存储在 res\drawable-mdpi 文件夹下。

提示：如果程序中需要对图片进行不等比例的拉伸时，可使用 android sdk 安装目录下 tools 文件夹中的 draw9patch 工具对图片进行处理，以达到更好的拉伸效果。

实例 4　游戏的架构

【实例描述】

在正式开发代码之前，首先对本游戏的设计框架进行简要介绍，希望可以帮助读者理解后面章节的内容。如果读者能够仔细阅读本节，则可以整体上了解本游戏，这对之后的开发带来事半功倍的效果。

为了让读者更好地理解后面的代码，下面将对游戏中各个类逐一进行简要说明，而关于这些类的详细代码及介绍将在后面的章节中相继给出。

【实现过程】

1．公共类

主类 MazeActivity.java 类：该类是通过继承和扩展基类 Activity 来实现的，是整个应用程序的入口。在该类中根据收到的 Handler 消息可以将界面切换到不同的界面，并且控制声音播放和场景渲染的暂停和恢复，以及 2D 界面中手机返回键的设置。

常量 Constant.java 类：该类记录程序中需要用到的常量，在该类中运用了大量的二维数组，大大地减少了代码量并优化了代码结构。

提示：程序开发中，常量类至关重要，能够避免开发人员在调试代码和修改代码时发生混乱的问题。

2．2D 界面类及其相关类

- 主菜单界面类 ViewMainMenu.java 类

该类实现了迷宫游戏主界面的渲染，用于设置游戏的操控模式，并且是游戏的开始入口，能根据单击开始游戏菜单向 MazeActivity.java 发送 Handler 消息，进而跳转到游戏加载界面，进而进入游戏。同时在该界面中可以退出游戏。

- ThreadSetView.java 类

该类用于控制设置界面选择进行选择后的动画效果。使玩家在选择操控模式时会有明显的黄色边框显示效果。

- 游戏加载界面是通过 load.xml 来实现的。
- 游戏胜利界面是通过 win.xml 来实现的。

3．3D 游戏界面及其相关类

- 游戏场景界面 MySurfaceView.java 类：该类主要用于游戏场景的渲染，并为触摸的遥感添加监听器，判断发生的事件。
- Ceil.java 类：该类的主要作用是绘制迷宫中的屋顶部分。
- Floor.java 类：该类的主要作用是绘制迷宫中的地板部分。
- Score.java 类：用于计算游戏中当前获取的晶体数量数据，并通过 TextureRect.java 类来得以绘制。
- TextureRect.java 类：该类的主要作用是绘制游戏中当前获取的晶体数量数据。
- TradPair.java 类：该类的主要作用是绘制 3D 迷宫中用到的晶体部件。
- TradPairGroup.java 类：用于根据地图设计，统一绘制迷宫中的所有晶体，并开启线程控制晶体自动旋转。
- Wall.java 类：该类的主要作用是绘制迷宫中的墙壁部分。
- KeyThread.java 类：该类用于实时监测第一人称视角人物在游戏中的运动状态，并且玩家可以通过键盘或是虚拟摇杆来控制人物的运动，改变其运动状态。

实例 5　游戏主类的设计与实现

【实例描述】

从本节开始将进入本游戏的开发，笔者相信动手是学习程序最快的方式，所以希望读者能够跟随笔者的思路亲自动手实现这个游戏，有条件的读者也可以将其移植到 Google Phone 中调试运行。

【实现过程】

应用程序中每个屏幕的显示都通过继承并扩展基类 Activity 类来实现。在 MazeActivity.java 类中完成游戏各个画面，以及游戏音乐的初始化工作等，该类的开发步骤如下。

【代码解析】

（1）游戏界面的搭建，首先是该类框架的介绍，以及音乐的初始化，代码如下。

代码位置：见随书光盘中源代码/第 14 章/GL_MASE/src/wyf/wpf/MazeActivity.java

```
1   package wyf.wpf;                                          //包名
2   import static wyf.wpf.Constant.*;        //包引用，该处有所省略，读者可自行查看源代码
3   ……//该处省略了部分包的导入代码，读者可自行查阅随书光盘中源代码
4   import android.view.Window;
5   public class MazeActivity extends Activity {
6       static final int START_GAME=0;                        //开始游戏的 Message 编号
7       static final int GAME_WIN=1;                          //游戏胜利界面编号
8       static final int GAME_LOAD=2;                         //加载界面编号
```

```
    static final int MAINMENU=3;                    //主界面编号
    private MySurfaceView mGLSurfaceView;
    Handler hd;                                     //消息处理器
    static MediaPlayer mpBack;                      //游戏背景音乐播放器
    static MediaPlayer mpWin;                       //游戏赢音乐播放器
    static SoundPool soundPool;                     //声音池
    static HashMap<Integer, Integer> soundPoolMap;
                                          //声音池中声音 ID 与自定义声音 ID 的 Map
    boolean isWin=false;                            //是否赢标记
    static boolean isXNJP=true;                     //是否是虚拟键盘操控
    static float screenHeight;                      //屏幕高度
    static float screenWidth;                       //屏幕宽度
    int viewFlag=-1;                                //当前界面标志位
    private ViewMainMenu vmm;                       //主菜单界面
   /*该处省略了重写的 onCreate 方法，将在后面给出*/
   protected void onResume(){                       //重写 onResume()方法
      super.onResume();
      threadWork=true;                              //标示软件运行中
      if(mGLSurfaceView!=null){
        if(isWin){
            mpWin.start();                  //如果在游戏胜利界面，则开启胜利音
        }
        else{
            mpBack.start();                 //如果在游戏中，则开启游戏背景音
        }}}
   protected void onPause(){                        //重写 onPause()方法
      super.onPause();
      threadWork=false;                             //标示软件在暂停中
      if(mpBack!=null){
        mpBack.pause();                             //暂停背景音的播放
      }
      if(mpWin!=null){
        mpWin.pause();                              //暂停胜利音的播放
      }}
   public void waitTwoSeconds(){                    //该方法用于实现使程序睡眠 2 秒的功能
        try {
                Thread.sleep(2000);                 //睡眠 2000 毫秒
            } catch (InterruptedException e) {
                e.printStackTrace();                //异常处理
            }}
   public void initSound() {                        //初始化声音的方法
    if(mpBack!=null){
        return;
    }
    mpBack = MediaPlayer.create(this, R.raw.gameback);  //背景音乐
    mpBack.setLooping(true);
    mpWin= MediaPlayer.create(this, R.raw.win);         //赢音乐
    mpWin.setLooping(true);
        soundPool = new SoundPool(4, AudioManager.STREAM_MUSIC, 100);
                                                        //声音池
      soundPoolMap = new HashMap<Integer, Integer>();
      soundPoolMap.put(1, soundPool.load(this, R.raw.gotobject, 1));
                                                        //吃东西音乐
   }
   public void playSound(int sound, int loop) {         //播放声音方法
     AudioManager mgr = (AudioManager)this.getSystemService(Context.AUDIO_
SERVICE);//设备
     float streamVolumeCurrent = mgr.getStreamVolume(AudioManager.STREAM_
```

```
MUSIC);  //当前音量
    63        float streamVolumeMax = mgr.getStreamMaxVolume(AudioManager.STREAM_
MUSIC);//最大音量
    64        float volume = streamVolumeCurrent / streamVolumeMax;
                                                              //音量比例
    65        soundPool.play(soundPoolMap.get(sound), volume, volume, 1, loop, 1f);
                                                              //播放音乐
    66       }
    67       /*该处省略了触摸控制的方法，将在后面给出*/
    68   }}
```

其中：

- 第 6～21 行为成员变量的声明部分，其中包括对界面编号的声明和声音资源对象的声明等。
- 第 22 行为重写的 onCreate()方法部分。该部分完成了界面的显示和跳转信息操作具体内容在下面小节做介绍。
- 第 23～32 行为重写 onResume()场景恢复方法，首先需要判断 mGLSurfaceView 是否为空，如果不为空则判断是否在赢的界面。
- 第 33～41 行为重写 onPause()场景暂停方法，并且需要判断 mpBack 是否为空。
- 第 42～47 行为实现使程序睡眠 2 秒的功能的方法。
- 第 48～59 行为初始化声音的方法。
- 第 60～66 行为播放声音池音效的方法。
- 第 67 行为按键单击事件处理方法，具体内容在下面章节做介绍。

（2）上述代码介绍的是本程序的主控制类的框架的开发，接下来向读者介绍的是 onCreate()方法的具体实现过程，将下列代码插入到主控制类框架的第 22 行。

代码位置：见随书光盘中源代码/第 14 章/GL_MASE/src/wyf/wpf/MazeActivity.java。

```
1       protected void onCreate(Bundle savedInstanceState) {
2          super.onCreate(savedInstanceState);                  //调用父类
3            requestWindowFeature(Window.FEATURE_NO_TITLE);  //全屏
4            getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN ,
5                      WindowManager.LayoutParams.FLAG_FULLSCREEN);
6            setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
                                                              //设置为竖屏模式
7          DisplayMetrics dm=new DisplayMetrics();            //获取屏幕分辨率
8          getWindowManager().getDefaultDisplay().getMetrics(dm);
9          screenHeight=dm.heightPixels;                      //纵向分辨率
10         screenWidth=dm.widthPixels;                        //横向分辨率
11           initSound();                                     //初始化声音
12           hd=new Handler() {                               //初始化消息处理器
13           public void handleMessage(Message msg){          //消息处理器内部类
14                super.handleMessage(msg);
15                switch(msg.what){                           //判断消息类型
16                  case START_GAME:                          //游戏界面
17                       mpBack.start();                      //开启游戏背景音声音
18                       mGLSurfaceView = new MySurfaceView(MazeActivity.this);
19                     mGLSurfaceView.requestFocus();         //获取焦点
20                     mGLSurfaceView.setFocusableInTouchMode(true);
                                                              //设置为可触控
21                     setContentView(mGLSurfaceView);        //跳转到游戏界面
22                     viewFlag=START_GAME;
23                     new KeyThread(mGLSurfaceView).start();//开启键盘监控线程
24                  break;
```

```
25                  case GAME_WIN:                                  //胜利界面
26                      setContentView(R.layout.win);               //切换到赢的界面
27                      viewFlag=GAME_WIN;
28                  break;
29                  case GAME_LOAD:                                 //加载界面
30                      setContentView(R.layout.load);              //显示游戏加载界面
31                      viewFlag=GAME_LOAD;
32                        new Thread(){
33                            public void run(){
34                                waitTwoSeconds();                 //睡眠两秒钟
35                                hd.sendEmptyMessage(START_GAME);
                                                                    //发消息加载游戏
36                            }}.start();
37                  break;
38                  case MAINMENU:                                  //主菜单界面
39                      vmm=new ViewMainMenu(MazeActivity.this); //创建主菜单界面对象
40                      setContentView(vmm);                        //跳转到游戏主菜单界面
41                        ThreadSetView tsv=new ThreadSetView(vmm);
                                                                    //新建动画线程
42                        ThreadSetView.flag=true;
43                       tsv.start();                               //启动线程
44                      vmm.setFocusableInTouchMode(true);          //设置为可触控
45                      vmm.requestFocus();                         //获取焦点
46                      viewFlag=MAINMENU;
47                  break;
48              }}};
49          vmm=new ViewMainMenu(this);                             //创建主菜单界面对象
50            setContentView(vmm);                                  //显示主菜单界面
51            ThreadSetView tsv=new ThreadSetView(vmm);             //动画线程
52           tsv.start();                                           //启动线程
53            viewFlag=MAINMENU;
54      }
```

其中：

- 第 3～5 行为游戏显示的全屏设置。第 6 行为游戏显示的强制竖屏显示。第 7～10 行为获取当前设备的分辨率。
- 第 12～48 行为消息处理器的实现部分。其中通过判断消息的类型来进行不同的处理，根据信息的不同，使界面跳转到相应的界面中。

（3）上述代码介绍的是主控制类重写的 onCreate 方法，接下来介绍的是 MazeActivity 类中对于游戏中按键触控事件的设置，将下列代码插入到主控制类的第 67 行。

代码位置：见随书光盘中源代码/第 14 章/GL_MASE/src/wyf/wpf/MazeActivity.java

```
1   public boolean onKeyDown(int keyCode, KeyEvent e) {
2           if(keyCode==4){
3               if(viewFlag==GAME_LOAD){                    //当界面显示为加载界面时
4                   return false;                           //不起作用
5               }
6               else{
7                   switch(viewFlag){                       //根据界面判断返回键操作
8                       case START_GAME:                    //当前界面为游戏界面
9                           hd.sendEmptyMessage(MAINMENU);//跳转到主菜单界面
10                          mpBack.pause();                 //游戏背景音效暂停
11                      break;
12                      case GAME_WIN:                      //当前界面为游戏胜利界面
13                          mpWin.pause();                  //游戏胜利音效暂停
14                          hd.sendEmptyMessage(MAINMENU);//跳转到主菜单界面
```

```
                    break;
                    case MAINMENU:                          //当前界面为主菜单界面
                        System.exit(0);                     //退出游戏
                    break;                                  //退出
                }}}
        if(mGLSurfaceView==null){                           //判断是否符合条件
            return false;                                   //返回 false
        }
        switch(keyCode){
          case 19:                                          //向上键代表前进
              KeyThread.keyState=1;                         //为 keyState 赋值 1
          break;
          case 20:                                          //向下键代表后退
              KeyThread.keyState=2;                         //为 keyState 赋值 2
          break;
          case 21:                                          //向左代表左转身
              KeyThread.keyState=3;                         //为 keyState 赋值 3
          break;
          case 22:                                          //向右代表右转身
              KeyThread.keyState=4;                         //为 keyState 赋值 4
          break;
        }
      return true;                                          //返回 true
    }
    public boolean onKeyUp(int keyCode, KeyEvent e){        //按键抬起触发事件
        if(keyCode>=19&&keyCode<=22){                       //如果为单击上下左右键
            KeyThread.keyState=0;                           //为 keyState 赋值 0
        }
        return true;                                        //返回 true
    }
```

其中：

- 第 2～19 行为游戏中返回键的设置。其中，第 3～5 行表示的是当前界面为加载界面时的设置，在加载界面中键盘不可控。第 6～19 行表示的是设置其他界面中返回键的触发操作。
- 第 20～38 行为在游戏界面中时，键盘上下左右键的触发事件。当单击不同的键时，会触发相应的运动状态。

实例 6　游戏常量类的设计与实现

【实现过程】

很多开发人员，尤其是初学编程的人常常会忽视在应用程序中编写常量类的作用，把应用程序中要用的常量编写到一个类中，一方面有利于调试程序，另一方面避免了魔法数字带来的不必要麻烦。

【代码解析】

在本节中将要介绍的是游戏的常量类的开发，代码如下。

代码位置：见随书光盘中源代码/第 14 章/GL_MASE/src/wyf/wpf/Constant.java。

```
1   package wyf.wpf;
2   public class Constant {                                 //创建类
```

```
    public static boolean threadWork=false;          //键盘监听线程工作标志位
    public static final float SPEED=0.05f;           //移动速度
    public static final float[][] MOVE_XZ={          //摄像机每个方向的移动位移
        {0,-SPEED},                                  //0-North  Z 轴负向
        {SPEED,0},                                   //1-EAST   X 轴正向
        {0,SPEED},                                   //2-SOUTH  Z 轴正向
        {-SPEED,0},                                  //3-WEST   X 轴负向
    };
    public static final float NEAR=0.45f;            //可视区域最近端
    public static final float ICON_DIS=NEAR;         //图标离视点的距离
    public static final float ICON_WIDTH=0.05f;      //图标尺寸
    public static final float ICON_HEIGHT=0.1f;
    public static final int[][] MAP={                //迷宫地图，0 不可过，1 可过
        {1,1,1,1,1,0,0,0,0,0},
        {0,0,0,0,1,0,0,0,0,0},
        {0,0,0,0,1,0,0,0,0,0},
        {0,0,0,0,1,0,0,0,0,0},
        {0,0,0,0,1,0,0,0,0,0},
        {0,0,0,0,1,1,1,1,0,0},
        {0,0,0,0,0,0,0,1,0,0},
        {0,1,1,1,1,1,1,1,0,0},
        {0,1,0,0,0,0,0,0,0,0},
        {0,1,0,0,0,0,0,0,0,0}
    };
    public static final int[][] MAP_OBJECT={         //表示可遇晶体位置的矩阵
        {0,1,0,0,0,0,0,0,0,0},
        {0,0,0,0,1,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,1,0,0,0,0,0},
        {0,0,0,0,0,0,0,1,0,0},
        {0,0,0,0,0,0,0,0,0,0},
        {0,1,0,0,1,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0}
    };
    public static final int OBJECT_COUNT=6;          //可遇晶体的数量
    public static final float WALL_HEIGHT=2.8f;      //墙的高度
    public static final float UNIT_SIZE=1.4f;        //地面每个格子的大小
    public static final float FLOOR_Y=-1.0f;         //地面的 Y 坐标
    public static final float CELL_Y=FLOOR_Y+WALL_HEIGHT;     //屋顶的 Y 坐标
    public static final int CAMERA_COL=1;            //初始时 Camera 位置
    public static final int CAMERA_ROW=9;
    public static final float HALF_COLL_SIZE=UNIT_SIZE/2-0.2f;  //碰撞体尺寸
}
```

提示：本段代码是把所用到的成员变量总结到了一起，方便读者进行查阅，各个成员变量的含义请参见代码中的注释。

实例 7　游戏主菜单类的设计与实现

【实现过程】

本游戏有许多界面，包括 Logo 界面、声音选择界面、菜单界面、帮助界面、关于界面以

及设置界面等界面，但是所用的方法都类似，它们都继承了 SurfaceView 类。本节将以游戏主菜单为例，介绍游戏主菜单的实现方法，以及与不同菜单之间的切换。

【代码解析】

（1）首先介绍的是 ViewMainMenu.java 类，其是游戏的主菜单界面，并且在该菜单中可以选择进入游戏的其他界面，代码如下。

代码位置：见随书光盘中源代码/第 14 章/GL_MASE/src/wyf/wpf/ViewMainMenu.java

```
1   package wyf.wpf;                                    //包名
2   import android.graphics.Bitmap;                     //包引用，该处有所省略，读者可自行查看源代码
3   ……//该处省略了部分包的导入代码，读者可自行查阅随书光盘中源代码
4   import android.view.SurfaceHolder;
5   public class ViewMainMenu extends SurfaceView implements SurfaceHolder.Callback{
6       MazeActivity activity;Bitmap mainMenu;                  //声明引用
7       Bitmap biankuang;                                       //单击显示边框位图
8       int dianjiFlag=-1;                                      //单击位置标志位
9       Paint paint;                                            //创建画笔
10      public ViewMainMenu(MazeActivity activity) {
11          super(activity);this.activity=activity;
12          this.getHolder().addCallback(this);                 //设置生命周期接口
13          paint=new Paint();                                  //创建画笔
14          paint.setAntiAlias(true);                           //打开抗锯齿
15      mainMenu=BitmapFactory.decodeResource(this.getResources(),
16                  R.drawable.zjm);                            //加载菜单图片
17      biankuang=BitmapFactory.decodeResource(this.getResources(),
18                  R.drawable.biankuang);                      //单击显示边框
19      }
20      public boolean onTouchEvent(MotionEvent e){             //触摸事件回调方法
21          float x=e.getX();float y=e.getY();                  //获取单击坐标
22        float yRatio=y/MazeActivity.screenHeight; float xRatio=x/MazeActivity.
screenWidth;
23          switch(e.getAction()){
24              case MotionEvent.ACTION_DOWN:
25               if(yRatio>0.491f&&yRatio<0.581f&&xRatio>0.031f&&xRatio<0.55f)
{   //按下开始游戏
26                  dianjiFlag=0;ThreadSetView.flag=false;
27                  activity.hd.sendEmptyMessage(MazeActivity.GAME_LOAD);
                                                            //加载游戏
28               }
29               if(yRatio>0.619f&&yRatio<0.708f&&xRatio>0.031f&&xRatio<0.55f)
{   //按下键盘模式
30                  dianjiFlag=1;MazeActivity.isXNJP=false;
31               }
32
if(yRatio>0.746f&&yRatio<0.836f&&xRatio>0.031f&&xRatio<0.55f){   //按下虚拟键模式
33                  dianjiFlag=2;MazeActivity.isXNJP=true;
34               }
35
if(yRatio>0.871f&&yRatio<0.961f&&xRatio>0.031f&&xRatio<0.55f){   //按下退出游戏
36                  dianjiFlag=3;System.exit(0);
37               }
38              break;
39          }
40          return true;
41      }
42      public void onDraw(Canvas canvas) {                     //重写 onDraw()
43          super.onDraw(canvas);canvas.drawBitmap(mainMenu,0,0, paint);
```

```
                                                    //绘制背景图
    44          switch(dianjiFlag){                 //判断单击的菜单选项
    45              case 0:                         //单击的为开始游戏菜单，绘制黄色边框图
    46                  canvas.drawBitmap(biankuang,14.88f,394.4f,
paint);dianjiFlag=-1;
    47              break;
    48              case 1:                         //单击的为键盘模式菜单，绘制黄色边框图
    49                  canvas.drawBitmap(biankuang,14.88f,495.2f,
paint);dianjiFlag=-1;
    50              break;
    51              case 2:                         //单击的为虚拟键模式菜单，绘制黄色边框图
    52                  canvas.drawBitmap(biankuang,14.88f,596.8f,
paint);dianjiFlag=-1;
    53              break;
    54              case 3:                         //单击的为退出游戏菜单，绘制黄色边框图
    55                  canvas.drawBitmap(biankuang,14.88f,696.8f,
paint);dianjiFlag=-1;
    56              break;
    57          }}
    58      public void surfaceChanged(SurfaceHolder holder, int format, int width,int
height) {
    59      }
    60      public void surfaceCreated(SurfaceHolder holder) {
    61          Canvas canvas = holder.lockCanvas();            //获取画布
    62          try{
    63              synchronized(holder){
    64                  onDraw(canvas);                         //绘制
    65              }}
    66          catch(Exception e){
    67              e.printStackTrace();                        //异常处理
    68          }
    69          finally{
    70              if(canvas != null){
    71                  holder.unlockCanvasAndPost(canvas);     //释放画布
    72              }}}
    73      public void surfaceDestroyed(SurfaceHolder holder) {
    74      }}
```

其中：

- 第 6～18 行为声明该类的成员变量，在构造器内获取 activity 引用，设置生命周期接口，创建画笔，打开抗锯齿并初始化加载主菜单图片。
- 第 20～41 行重写 onTouchEvent 方法，设置一个按钮位置，并为按钮指定单击后的事件处理。
- 第 42～74 行为重写 onDraw、surfaceChange、surfaceCreated 方法，在 surfaceCreated 方法中创建画布，在 onDraw 方法中绘制主菜单界面。

（2）接下来向读者介绍在主菜单界面，选择菜单单击后边框显示黄色的实现方法。

代码位置：见随书光盘中源代码/第 14 章/GL_MASE/src/wyf/wpf/ViewMainMenu.java

```
1   package wyf.wpf;                                    //包名
2   import android.graphics.Canvas;                     //包引用
3   import android.view.SurfaceHolder;
4   public class ThreadSetView extends Thread{          //线程类继承 Thread 父类
5       ViewMainMenu setView;                           //创建引用
6       SurfaceHolder holder;                           //声明 SurfaceHolder 对象
7       static boolean flag=true;
8       public ThreadSetView(ViewMainMenu setView){     //构造器
9           this.setView=setView;
```

```
10          this.holder=setView.getHolder();
11      }
12      public void run(){                                          //重写 run()方法
13          Canvas canvas;                                          //声明画布
14          while(flag){                                            //判断线程循环是否执行
15              canvas=null;
16              if(true){
17                  try{
18                      canvas=this.holder.lockCanvas();            //获取画布对象
19                      synchronized(this.holder){                  //同步获取
20                          setView.onDraw(canvas);                 //绘制界面
21                      }
22                  }catch(Exception e){
23                      e.printStackTrace();                        //异常处理
24                  }
25                  finally{
26                      if(canvas!=null){                           //判断 canvas 是否为空
27                          this.holder.unlockCanvasAndPost(canvas);
//释放画布
28                      }}}
29              try{
30                  sleep(200);                                     //睡眠 200 毫秒
31              }
32              catch(Exception e){                                 //捕获异常
33                  e.printStackTrace();                            //异常处理
34              }}}}
```

提示：该类主要用于实现在主界面中单击了菜单项后，菜单边框有显示黄色边框的小动画效果。主要通过绘制黄色边框位图在指定的位置，并通过该线程控制其显示的时间为 200 毫秒。

实例 8　游戏界面的设计与实现

【实现过程】

本游戏作为 3D 游戏，游戏界面继承自 GLSurfaceView.3D 游戏界面，其中主要实现了游戏场景和各个部件的绘制和控制。为了让读者能够深入了解游戏界面的设计，笔者将先给出游戏界面的框架，然后再逐一地对本部分内容进行详细介绍。

【代码解析】

（1）下面就从游戏界面的整体框架来介绍游戏界面的开发，代码如下。

代码位置：见随书光盘中源代码/第 14 章/GL_MASE/src/wyf/wpf/MySurfaceView.java

```
1   package wyf.wpf;                                        //包名
2   import java.io.IOException;                             //引用相关包
3   ……//该处省略了部分包的导入代码，读者可自行查阅随书光盘中源代码
4   import android.graphics.Bitmap;                         //引用相关包
5   class MySurfaceView extends GLSurfaceView {
6       public SceneRenderer mRenderer;                     //场景渲染器
7       public int currentDirection=0;                      //初始方向为 NORTH
8      public float heroX=CAMERA_COL*UNIT_SIZE+UNIT_SIZE/2;      //人眼的 XYZ 坐标
9      public float heroY=0.4f;
```

```
    public float heroZ=CAMERA_ROW*UNIT_SIZE+UNIT_SIZE/2;
    public float heroXT=heroX;                              //人眼目标点的 XYZ 坐标
    public float heroYT=0.4f;
    public float heroZT=heroZ-SPEED;
     public int floorTexId;                                 //地板纹理 ID
     public int ceilTexId;                                  //屋顶纹理 ID
     public int wallTexId;                                  //墙纹理 ID
     public int robotTexId;                                 //晶体纹理 ID
     public int iconTexId;                                  //晶体图标纹理 ID
     public int numberTexId;                                //得分数字纹理 ID
     public int controlTexId;                               //虚拟摇杆纹理 ID
     public int redDotTexId;                                //虚拟摇杆红点纹理 ID
     public int objectCount=0;                              //已吃到的晶体数量
     public MySurfaceView(Context context) {
       super(context);
       mRenderer = new SceneRenderer();                     //创建场景渲染器
       setRenderer(mRenderer);                              //设置渲染器
       setRenderMode(GLSurfaceView.RENDERMODE_CONTINUOUSLY);
                                                            //设置渲染模式为主动渲染
    }
    public boolean onTouchEvent(MotionEvent e) {            //触摸事件回调方法
     if(MazeActivity.isXNJP==false){                        //如果不是传感器模式，虚拟键盘不可用
         return true;
     }
       float y = e.getY();float x = e.getX();               //获取屏幕触控点坐标
       float yRatio=y/MazeActivity.screenHeight;
                                        //获取当前触控点纵坐标位置占总屏幕高度的比例
       float xRatio=x/MazeActivity.screenWidth;
                                        //获取当前触控点横坐标位置占总屏幕宽度的比例
       switch (e.getAction()){                              //判断当前触控方式
           case MotionEvent.ACTION_DOWN:                    //按下动作
              if(yRatio>0.77f&&yRatio<0.831f&&xRatio>0.756f&&xRatio<0.85f)
{//按下前进虚拟按钮
                  KeyThread.keyState=1;                     //前进
              }
            else if(yRatio>0.886f&&yRatio<0.948f&&xRatio>0.756f&&xRatio<0.85f)
{//按下后退虚拟按钮
                  KeyThread.keyState=2;                     //后退
              }
             else if(yRatio>0.831f&&yRatio<0.886f&&xRatio>0.656f&&xRatio<
0.752f){//按下左转虚拟按钮
                  KeyThread.keyState=3;                     //向左转
              }
             else if(yRatio>0.831f&&yRatio<0.886f&&xRatio>0.85f&&xRatio
<0.958f){//按下右转虚拟按钮
                  KeyThread.keyState=4;                     //向右转
              }else{
                  KeyThread.keyState=0;                     //初始化当前状态
              }
           break;
           case MotionEvent.ACTION_UP:                      //抬起动作
             KeyThread.keyState=0;                          //初始化当前状态
        break;
          }
       return true;
     }
    /*该处省略了内部类的方法，将在后面给出*/
    /*该处省略了部分类的开发方法，将在后面给出*/
```

```
61 }
```

其中：

- 第 6～22 行为声明场景中运用的常量。
- 第 23～28 行为 MySurfaceView.java 类的构造器，其中设置了渲染器，初始化车的位置并且创建了游戏中的控制线程。
- 第 29～58 行为游戏中屏幕触控事件的处理，主要针对虚拟摇杆的触控操作，当单击摇杆上侧时，前进；当单击摇杆下侧的时候，后退；当单击摇杆左侧的时候，向左转；当单击摇杆右侧的时候，向右转。

（2）上述代码介绍的是游戏界面的框架，接下来介绍的是给出渲染器内部类 SceneRenderer.java 类的代码内容，该方法主要用于渲染游戏场景，将下列代码插入到游戏界面框架的第 59 行。

代码位置：见随书光盘中源代码/第 14 章/GL_MASE/src/wyf/wpf/MySurfaceView.java

```
1    class SceneRenderer implements GLSurfaceView.Renderer{
2         Floor floor; Ceil ceil; Wall wall;          //地板、屋顶、墙
3         TradPairGroup tpg;TextureRect icon;         //晶体组、晶体图标
4         Score score;TextureRect control;            //当前获得的晶体数量、虚拟摇杆
5         TextureRect redDot;                         //虚拟摇杆红点
6         public void onDrawFrame(GL10 gl) {
7            gl.glShadeModel(GL10.GL_SMOOTH);         //采用平滑着色
8           gl.glEnable(GL10.GL_CULL_FACE);           //设置为打开背面剪裁
9    gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
                                                     //清除颜色缓存、深度缓存
10           gl.glMatrixMode(GL10.GL_MODELVIEW);     //设置当前矩阵为模式矩阵
11           gl.glLoadIdentity();                    //设置当前矩阵为单位矩阵
12           gl.glEnable(GL10.GL_LIGHTING);          //允许光照
13           float[] positionParams={heroX,heroY,heroZ,1};
                                                     //设定白色光源的位置，采用定位光
14           gl.glLightfv(GL10.GL_LIGHT1, GL10.GL_POSITION, positionParams,0);
15           GLU.gluLookAt(gl,heroX,heroY,  heroZ,heroXT,heroYT,heroZT,0,1,0 );
                                                     //摄像机设置
16           gl.glPushMatrix();gl.glTranslatef(0,Constant.FLOOR_Y,0);
17           floor.drawSelf(gl);gl.glPopMatrix();    //绘制地板
18           gl.glPushMatrix();gl.glTranslatef(0,Constant.CELL_Y,0);
19           ceil.drawSelf(gl);gl.glPopMatrix();     //绘制屋顶
20           gl.glPushMatrix();wall.drawSelf(gl);gl.glPopMatrix();   //绘制墙
21           gl.glPushMatrix();  tpg.drawSelf(gl);gl.glPopMatrix(); //绘制晶体组
22             gl.glMatrixMode(GL10.GL_MODELVIEW); //恢复成初始状态绘制晶体图标与数量
23           gl.glLoadIdentity();                    //设置当前矩阵为单位矩阵
24           gl.glDisable(GL10.GL_LIGHTING);         //禁止光照
25           gl.glEnable(GL10.GL_BLEND);             //开启混合
26             gl.glBlendFunc(GL10.GL_SRC_ALPHA, GL10.GL_ONE_MINUS_SRC_ALPHA);
27           gl.glPushMatrix();gl.glTranslatef(-0.3f, 0.90f, -ICON_DIS-0.01f);
28           icon.drawSelf(gl);                      //绘制晶体图标
29           gl.glTranslatef(ICON_WIDTH, 0, 0);
30           score.drawSelf(gl);gl.glPopMatrix();    //绘制已获得晶体数量
31           if(MazeActivity.isXNJP) {               //选择的是虚拟键模式
32              gl.glPushMatrix();gl.glTranslatef(0.225f, -0.75f, -ICON_DIS-
0.015f); //平移操作
33              control.drawSelf(gl);gl.glPopMatrix();          //绘制
34              gl.glPushMatrix();
35              switch(KeyThread.keyState) {                    //判断按键状态
36                case 0:gl.glTranslatef(0.22f, -0.735f, -ICON_DIS-0.01f);break;
                                                     //没有键按下
```

```
37                  case 1:gl.glTranslatef(0.22f, -0.63f, -ICON_DIS-0.01f);break;
                                                          //向上
38                  case 2:gl.glTranslatef(0.22f, -0.84f, -ICON_DIS-0.01f);break;
                                                          //向下
39                 case 3:gl.glTranslatef(0.145f, -0.735f, -ICON_DIS-0.01f);break;
                                                          //向左
40                  case 4:gl.glTranslatef(0.295f, -0.735f, -ICON_DIS-0.01f);break;
                                                          //向右
41                }
42                redDot.drawSelf(gl); gl.glPopMatrix();
43             }
44             gl.glDisable(GL10.GL_BLEND);                //禁止混合
45         }
46         public void onSurfaceChanged(GL10 gl, int width, int height) {
47            gl.glViewport(0, 0, width, height);          //设置视窗大小及位置
48             gl.glMatrixMode(GL10.GL_PROJECTION);gl.glLoadIdentity();
                                                          //设置矩阵
49             float ratio = (float) width / height;       //计算透视投影的比例
50             gl.glFrustumf(-ratio*0.6f, ratio*0.6f, -1, 1,NEAR, 100);
                                                    //调用此方法计算产生透视投影矩阵
51         }
52         public void onSurfaceCreated(GL10 gl, EGLConfig config) {
53            gl.glDisable(GL10.GL_DITHER);                //关闭抗抖动
54             gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT,GL10.GL_FASTEST);
55             gl.glClearColor(0,0,0,0);                   //设置屏幕背景色为黑色 RGBA
56             gl.glEnable(GL10.GL_DEPTH_TEST);            //启用深度测试
57       floorTexId=initTexture(gl,R.drawable.floor);ceilTexId=initTexture(gl,R.
drawable.ceil);  //初始化纹理
58       wallTexId=initTexture(gl,R.drawable.wall); robotTexId=initTexture(gl,R.
drawable.robot);
59       iconTexId=initTexture(gl,R.drawable.bb);  numberTexId=initTexture(gl,R.
drawable.number);
60       controlTexId=initTexture(gl,R.drawable.control);redDotTexId=initTexture
(gl,R.drawable.reddot);
61       floor=new Floor(0,0,1,0,floorTexId,Constant.MAP[0].length,Constant.MAP.
length); //创建要绘制的物件
62             ceil=new   Ceil(0,0,1,0,ceilTexId,Constant.MAP[0].length,Constant.
MAP.length);
63             wall=new Wall(wallTexId);tpg=new TradPairGroup(robotTexId);
64             icon=new TextureRect(iconTexId,ICON_WIDTH/2,ICON_HEIGHT/2,
65                  new float[]  { 0,0,0,1,1,0,0,1,1,1,1,0
66                   });                                    //创建 icon 纹理 Id
67             score=new Score(numberTexId,MySurfaceView.this);//创建 score 纹理 Id
68             control=new TextureRect(controlTexId, 0.12f,0.2f, //创建 control 纹理 Id
69                  new float[]{0,0,0,1,1,0,0,1,1,1,1,0
70                   });
71             redDot=new TextureRect(redDotTexId,0.03f, 0.05f, //创建 redDot 纹理 Id
72                  new float[]{0,0,0,1,1,0,0,1,1,1,1,0
73                    });
74             initWhiteLight(gl);                         //初始化白色灯
75            initMaterial(gl);                            //初始化材质
76         }}
```

其中：

- 第 2～5 行为渲染器内部类中物体对象和纹理 Id 对象的声明。
- 第 6～45 行为重写 onDrawFrame()方法，用于绘制当前游戏场景，该方法为系统自动调用。第 46～51 行为重写 onSurfaceChanged()方法；第 52～76 行为重写 onSurfaceCreated()方法。

（3）上述代码介绍的是 SceneRenderer 框架的开发，接下来介绍的是纹理 Id 初始化方法和初始化光照方法，以及初始化场景材质方法，将下列代码插入到游戏界面框架的第 61 行。

代码位置：见随书光盘中源代码/第 14 章/GL_MASE/src/wyf/wpf/MySurfaceView.java

```
1        public int initTexture(GL10 gl,int drawableId){      //初始化纹理
2            int[] textures = new int[1];                     //生成纹理 Id
3            gl.glGenTextures(1, textures, 0);                //自动分配一个纹理名称
4            int currTextureId=textures[0];          //将纹理名称添加到纹理数组中
5            gl.glBindTexture(GL10.GL_TEXTURE_2D, currTextureId);//绑定纹理
6    gl.glTexParameterf(GL10.GL_TEXTURE_2D,GL10.GL_TEXTURE_MIN_FILTER,
GL10.GL_NEAREST);
7    gl.glTexParameterf(GL10.GL_TEXTURE_2D,GL10.GL_TEXTURE_MAG_FILTER,
GL10.GL_LINEAR);
8    gl.glTexParameterf(GL10.GL_TEXTURE_2D, GL10.GL_TEXTURE_WRAP_S,GL10.GL_REPEAT);
9    gl.glTexParameterf(GL10.GL_TEXTURE_2D, GL10.GL_TEXTURE_WRAP_T,GL10.GL_REPEAT);
10         InputStream is = this.getResources().openRawResource(drawableId);
                                                              //位图资源流数据
11         Bitmap bitmapTmp;
12         try{
13           bitmapTmp = BitmapFactory.decodeStream(is);     //获取当前位图对象
14         }
15         finally{                                          //运行至最后需要做的
16            try{
17               is.close();                                //关闭流
18            }
19            catch(IOException e){
20               e.printStackTrace();                       //异常处理
21            }}
22         GLUtils.texImage2D(GL10.GL_TEXTURE_2D, 0, bitmapTmp, 0);
                                                           //指定纹理 Id 对应纹理
23         bitmapTmp.recycle();                              //回收位图资源
24         return currTextureId;
25       }
26       private void initWhiteLight(GL10 gl){             //初始化白色灯
27         gl.glEnable(GL10.GL_LIGHT1);                    //打开 1 号灯
28         float[] ambientParams={0.2f,0.2f,0.05f,1.0f}; //环境光设置，光参数 RGBA
29         gl.glLightfv(GL10.GL_LIGHT1, GL10.GL_AMBIENT, ambientParams,0);
30         float[] diffuseParams={0.9f,0.9f,0.2f,1.0f};  //散射光设置，光参数 RGBA
31         gl.glLightfv(GL10.GL_LIGHT1, GL10.GL_DIFFUSE, diffuseParams,0);
32         float[] specularParams={1f,1f,1f,1.0f};       //反射光设置，光参数 RGBA
33         gl.glLightfv(GL10.GL_LIGHT1, GL10.GL_SPECULAR, specularParams,0);
34       }
35       private void initMaterial(GL10 gl){             //初始化材质
36         float ambientMaterial[] = {1.0f, 1.0f, 1.0f, 1.0f}; //环境光为白色材质
37         gl.glMaterialfv(GL10.GL_FRONT_AND_BACK, GL10.GL_AMBIENT, ambient
Material,0);
38         float diffuseMaterial[] = {1.0f, 1.0f, 1.0f, 1.0f}; //散射光为白色材质
39         gl.glMaterialfv(GL10.GL_FRONT_AND_BACK, GL10.GL_DIFFUSE, diffuse
Material,0);
40         float specularMaterial[] = {1f, 1f, 1f, 1.0f};    //高光材质为白色
41         gl.glMaterialfv(GL10.GL_FRONT_AND_BACK, GL10.GL_SPECULAR, specular
Material,0);
42         gl.glMaterialf(GL10.GL_FRONT_AND_BACK, GL10.GL_SHININESS, 4.0f);
43       }
```

提示：第 1～25 行为获取所给位图资源对应的纹理 Id。第 26～34 行为初始化光照属性。第 35～43 行为初始化场景材质属性。

实例 9　游戏界面中主要场景的绘制

【实现过程】

本游戏场景由很多部件组成，如地板、屋顶和墙壁，还有晶体及晶体技术的实现，下面来向读者一一介绍。

【代码解析】

（1）首先向读者介绍的是地板类 Floor.java 类的具体开发过程。

代码位置：见随书光盘中源代码/第 14 章/GL_MASE/src/wyf/wpf/Floor.java

```
1   package wyf.wpf;                                    //包名
2   import java.nio.ByteBuffer;                         //包引用，该处有所省略，读者可自行查看源代码
3   ......// 该处省略了导入相关包的代码，读者可自行查阅随书光盘中源代码
4   import java.nio.FloatBuffer;
5   public class Floor {                                //表示地板的类
6       private FloatBuffer  mVertexBuffer;             //顶点坐标数据缓冲
7      private FloatBuffer  mNormalBuffer;              //顶点法向量数据缓冲
8      private FloatBuffer mTextureBuffer;              //顶点纹理数据缓冲
9      int vCount=0;                                    //顶点数量
10     float yAngle;                                    //y 轴旋转角度
11     int xOffset;int zOffset;                         //偏移量
12     int texId;                                       //纹理 Id
13     int width;                                       //地板横向 width 个单位
14     int height;                                      //地板纵向 height 个单位
15     public Floor(int xOffset,int zOffset,float scale,float yAngle,int texId,int
width,int height){
16      this.xOffset=xOffset;this.zOffset=zOffset;this.yAngle=yAngle;
17      this.texId=texId;this.width=width;this.height=height;
18        vCount=width*height*6;                         //每个地板块 6 个顶点
19        float vertices[]=new float[vCount*3];int k=0;
20        for(int i=0;i<width;i++)
21           for(int j=0;j<height;j++){                  //每个地板块由两个三角形 6 个顶点构成
22             vertices[k++]=i*UNIT_SIZE*scale;vertices[k++]=0;vertices[k++]=j*
UNIT_SIZE*scale;
23             vertices[k++]=i*UNIT_SIZE*scale;vertices[k++]=0;vertices[k++]=
(j+1)*UNIT_SIZE*scale;
24             vertices[k++]=(i+1)*UNIT_SIZE*scale;vertices[k++]=0;vertices
[k++]=(j+1)*UNIT_SIZE*scale;
25             vertices[k++]=(i+1)*UNIT_SIZE*scale;vertices[k++]=0;vertices
[k++]=(j+1)*UNIT_SIZE*scale;
26             vertices[k++]=(i+1)*UNIT_SIZE*scale;vertices[k++]=0;vertices
[k++]=j*UNIT_SIZE*scale;
27             vertices[k++]=i*UNIT_SIZE*scale;vertices[k++]=0;vertices[k++]=
j*UNIT_SIZE*scale;
28             };
29       ByteBuffer vbb = ByteBuffer.allocateDirect(vertices.length*4);
                                                        //创建顶点坐标数据缓冲
30       vbb.order(ByteOrder.nativeOrder());            //设置字节顺序
31       mVertexBuffer = vbb.asFloatBuffer();           //转换为 int 型缓冲
32       mVertexBuffer.put(vertices);                   //向缓冲区中放入顶点坐标数据
33       mVertexBuffer.position(0);                     //设置缓冲区起始位置
34       float normals[]=new float[vCount*3];
35       for(int i=0;i<vCount;i++){
36         normals[i*3]=0;normals[i*3+1]=1;normals[i*3+2]=0;
```

```
37          }
38          ByteBuffer nbb = ByteBuffer.allocateDirect(normals.length*4);
39          nbb.order(ByteOrder.nativeOrder());          //设置字节顺序
40          mNormalBuffer = nbb.asFloatBuffer();         //转换为 int 型缓冲
41          mNormalBuffer.put(normals);                  //向缓冲区中放入顶点着色数据
42          mNormalBuffer.position(0);                   //设置缓冲区起始位置
43          float[] texST=new float[vCount*2];           //纹理坐标数据初始化
44          for(int i=0;i<vCount*2/12;i++){
45             texST[i*12]=0;texST[i*12+1]=0;texST[i*12+2]=0;texST[i*12+3]=2;
46             texST[i*12+4]=2;texST[i*12+5]=2;texST[i*12+6]=2;texST[i*12+7]=2;
47             texST[i*12+8]=2;texST[i*12+9]=0;texST[i*12+10]=0;texST[i*12+11]=0;
48          };
49          ByteBuffer tbb = ByteBuffer.allocateDirect(texST.length*4);
50          tbb.order(ByteOrder.nativeOrder());          //设置字节顺序
51          mTextureBuffer = tbb.asFloatBuffer();        //转换为 int 型缓冲
52          mTextureBuffer.put(texST);                   //向缓冲区中放入顶点着色数据
53          mTextureBuffer.position(0);                  //设置缓冲区起始位置
54      }
55      public void drawSelf(GL10 gl){
56          gl.glEnableClientState(GL10.GL_VERTEX_ARRAY); //启用顶点坐标数组
57          gl.glEnableClientState(GL10.GL_NORMAL_ARRAY); //启用顶点法向量数组
58          gl.glPushMatrix();                                  //保护现场
59          gl.glTranslatef(xOffset*UNIT_SIZE, 0, 0);
60          gl.glTranslatef(0, 0, zOffset*UNIT_SIZE);
61          gl.glRotatef(yAngle, 0, 1, 0);
62          gl.glVertexPointer (3,GL10.GL_FLOAT,0,mVertexBuffer);
                                                        //为画笔指定顶点坐标数据
63          gl.glNormalPointer(GL10.GL_FLOAT, 0, mNormalBuffer);
                                                        //为画笔指定顶点法向量数据
64          gl.glEnable(GL10.GL_TEXTURE_2D);            //开启纹理
65          gl.glEnableClientState(GL10.GL_TEXTURE_COORD_ARRAY);
                                                        //允许使用纹理 ST 坐标缓冲
66          gl.glTexCoordPointer(2, GL10.GL_FLOAT, 0, mTextureBuffer);
                                                        //为画笔指定纹理 ST 坐标缓冲
67          gl.glBindTexture(GL10.GL_TEXTURE_2D, texId);       //绑定当前纹理
68          gl.glDrawArrays(GL10.GL_TRIANGLES, 0, vCount );    //绘制图形
69          gl.glPopMatrix();                                  //恢复现场
70      }}
```

其中：

- 第 6～14 行为声明成员变量，包括顶点数据缓存对象、顶点着色数据缓存对象和顶点索引数据缓存对象和顶点数量记录对象声明，顶点索引数量记录对象声明，以及场景选择角度对象声明。
- 第 15～54 行为该类构造的开发，其中第 16～31 行为获取顶点数组并将其存入顶点数据缓存；第 32～46 行为获取顶点法向量数组并将其存入顶点法向量数据缓存；第 47～52 行为获取顶点纹理数据并将其存入顶点纹理数据缓存中。
- 第 55～70 行为绘制场景的方法。其中第 62 行为场景指定顶点坐标数据；第 63 行为场景指定顶点法向量数据；第 66 行为场景指定顶点纹理数据，并绘制场景。

（2）接下来向读者介绍的是屋顶类 Ceil.java 类的开发实现过程。

代码位置：见随书光盘中源代码/第 14 章/GL_MASE/src/wyf/wpf/Ceil.java。

```
1   package wyf.wpf;                                    //包名
2   import java.nio.ByteBuffer;                         //导入相关包
3   ......// 该处省略了导入相关包的代码，读者可自行查阅随书光盘中源代码
4   import static wyf.wpf.Constant.*;                   //导入相关包
```

```
5   public class Ceil {                                    //表示屋顶的类
6       private FloatBuffer   mVertexBuffer;               //顶点坐标数据缓冲
7      private FloatBuffer   mNormalBuffer;                //顶点法向量数据缓冲
8      private FloatBuffer mTextureBuffer;                 //顶点纹理数据缓冲
9      int vCount=0;                                       //顶点数量
10     float yAngle;                                       //y 轴旋转角度
11     int xOffset;                                        //x 偏移量
12     int zOffset;                                        //z 平移量
13     int texId;                                          //纹理 ID
14     int width;                                          //屋顶横向 width 个单位
15     int height;                                         //屋顶纵向 height 个单位
16     public Ceil(int xOffset,int zOffset,float scale,float yAngle,int texId,int
width,int height){
17      this.xOffset=xOffset;this.zOffset=zOffset;this.yAngle=yAngle;
18      this.texId=texId;this.width=width;this.height=height;    //行数
19        vCount=width*height*6;                            //每个屋顶块 6 个顶点
20        float vertices[]=new float[vCount*3]; int k=0;
21        for(int i=0;i<width;i++)
22          for(int j=0;j<height;j++) {          //每个屋顶块由两个三角形 6 个顶点构成
23          vertices[k++]=(i+1)*UNIT_SIZE*scale;vertices[k++]=0;vertices[k++]=
(j+1)*UNIT_SIZE*scale;
24          vertices[k++]=i*UNIT_SIZE*scale;vertices[k++]=0;vertices[k++]=
(j+1)*UNIT_SIZE*scale;
25          vertices[k++]=i*UNIT_SIZE*scale;vertices[k++]=0;vertices[k++]=
j*UNIT_SIZE*scale;
26          vertices[k++]=i*UNIT_SIZE*scale;vertices[k++]=0;vertices[k++]=
j*UNIT_SIZE*scale;
27          vertices[k++]=(i+1)*UNIT_SIZE*scale;vertices[k++]=0;vertices[k++]
=j*UNIT_SIZE*scale;
28          vertices[k++]=(i+1)*UNIT_SIZE*scale;vertices[k++]=0;vertices[k++]=
(j+1)*UNIT_SIZE*scale;
29            };
30        ByteBuffer vbb = ByteBuffer.allocateDirect(vertices.length*4);
                                                           //创建顶点坐标数据缓冲
31        vbb.order(ByteOrder.nativeOrder());              //设置字节顺序
32        mVertexBuffer = vbb.asFloatBuffer();             //转换为 float 型缓冲
33        mVertexBuffer.put(vertices);                     //向缓冲区中放入顶点坐标数据
34        mVertexBuffer.position(0);                       //设置缓冲区起始位置
35        float normals[]=new float[vCount*3];
36        for(int i=0;i<vCount;i++){
37          normals[i*3]=0;normals[i*3+1]=-1;normals[i*3+2]=0;
38        }
39        ByteBuffer nbb = ByteBuffer.allocateDirect(normals.length*4);
40        nbb.order(ByteOrder.nativeOrder());              //设置字节顺序
41        mNormalBuffer = nbb.asFloatBuffer();             //转换为 int 型缓冲
42        mNormalBuffer.put(normals);                      //向缓冲区中放入顶点着色数据
43        mNormalBuffer.position(0);                       //设置缓冲区起始位置
44        float[] texST=new float[vCount*2];               //纹理坐标数据初始化
45        for(int i=0;i<vCount*2/12;i++) {
46          texST[i*12]=0;texST[i*12+1]=0;texST[i*12+2]=0;texST[i*12+3]=2;
47          texST[i*12+4]=2;texST[i*12+5]=2;texST[i*12+6]=2;texST[i*12+7]=2;
48          texST[i*12+8]=2;texST[i*12+9]=0;texST[i*12+10]=0;texST[i*12+11]=0;
49        };
50        ByteBuffer tbb = ByteBuffer.allocateDirect(texST.length*4);
51        tbb.order(ByteOrder.nativeOrder());              //设置字节顺序
52        mTextureBuffer = tbb.asFloatBuffer();            //转换为 int 型缓冲
53        mTextureBuffer.put(texST);                       //向缓冲区中放入顶点着色数据
54        mTextureBuffer.position(0);                      //设置缓冲区起始位置
55    }
```

```
56      public void drawSelf(GL10 gl){
57         gl.glEnableClientState(GL10.GL_VERTEX_ARRAY); //启用顶点坐标数组
58         gl.glEnableClientState(GL10.GL_NORMAL_ARRAY); //启用顶点法向量数组
59         gl.glPushMatrix(); gl.glTranslatef(xOffset*UNIT_SIZE, 0, 0);
                                                          //保护现场
60         gl.glTranslatef(0, 0, zOffset*UNIT_SIZE);gl.glRotatef(yAngle, 0, 1, 0);
61         gl.glVertexPointer(3,GL10.GL_FLOAT,0, mVertexBuffer);
                                                          //为画笔指定顶点坐标数据
62         gl.glNormalPointer(GL10.GL_FLOAT, 0, mNormalBuffer);
                                                          //为画笔指定顶点法向量数据
63         gl.glEnable(GL10.GL_TEXTURE_2D);               //开启纹理
64         gl.glEnableClientState(GL10.GL_TEXTURE_COORD_ARRAY);
                                                          //允许使用纹理 ST 坐标缓冲
65         gl.glTexCoordPointer(2, GL10.GL_FLOAT, 0, mTextureBuffer);
                                                          //为画笔指定纹理 ST 坐标缓冲
66         gl.glBindTexture(GL10.GL_TEXTURE_2D, texId);  //绑定当前纹理
67         gl.glDrawArrays(GL10.GL_TRIANGLES, 0, vCount);//绘制图形
68         gl.glPopMatrix();                              //恢复现场
69      }}
```

其中：

- 第 6～16 行为声明成员变量，包括顶点数据缓存对象、顶点着色数据缓存对象和顶点索引数据缓存对象和顶点数量记录对象声明，顶点索引数量记录对象声明，以及场景选择角度对象声明。
- 第 17～55 行为该类构造的开发，其中第 19～34 行为获取顶点数组并将其存入顶点数据缓存；第 35～49 行为获取顶点法向量数组并将其存入顶点法向量数据缓存；第 50～54 行为获取顶点纹理数据并将其存入顶点纹理数据缓存中。
- 第 56～69 行为绘制场景的方法。其中第 61 行为场景指定顶点坐标数据；第 62 行为场景指定顶点法向量数据；第 65 行为场景指定顶点纹理数据，并绘制场景。

（3）然后向读者介绍迷宫墙壁类 Wall.java 类的具体开发过程。

代码位置：见随书光盘中源代码/第 14 章/GL_MASE/src/wyf/wpf/Wall.java

```
1   package wyf.wpf;                              //包名
2   import java.nio.ByteBuffer;                   //包引用，该处有所省略，读者可自行查看源代码
3
4   import java.util.ArrayList;
5   public class Wall {                                  //表示墙的类
6       private FloatBuffer  mVertexBuffer;              //顶点坐标数据缓冲
7      private FloatBuffer  mTextureBuffer;              //顶点纹理数据缓冲
8      private FloatBuffer  mNormalBuffer;               //顶点法向量数据缓冲
9      int vCount;                                       //顶点数量
10     int texId;                                        //纹理 Id
11     public Wall(int texId){
12     this.texId=texId; int rows=MAP.length; int cols=MAP[0].length;
13        ArrayList<Float> alVertex=new ArrayList<Float>();
14        ArrayList<Float> alNormal=new ArrayList<Float>();
15        ArrayList<Float> alTexture=new ArrayList<Float>();
16        ... ...//墙壁顶点的计算，该处详细内容在下面做详细介绍
17      vCount=alVertex.size()/3;
18        float vertices[]=new float[alVertex.size()];
19        for(int i=0;i<alVertex.size();i++) {
20          vertices[i]=alVertex.get(i);
21        }
22        ByteBuffer vbb = ByteBuffer.allocateDirect(vertices.length*4);
                                                          //创建顶点坐标数据缓冲
```

```
23          vbb.order(ByteOrder.nativeOrder());          //设置字节顺序
24          mVertexBuffer = vbb.asFloatBuffer();         //转换为 int 型缓冲
25          mVertexBuffer.put(vertices);                 //向缓冲区中放入顶点坐标数据
26          mVertexBuffer.position(0);                   //设置缓冲区起始位置
27          float normals[]=new float[vCount*3];
28          for(int i=0;i<vCount*3;i++){
29            normals[i]=alNormal.get(i);
30          }
31          ByteBuffer nbb = ByteBuffer.allocateDirect(normals.length*4);
32          nbb.order(ByteOrder.nativeOrder());          //设置字节顺序
33          mNormalBuffer = nbb.asFloatBuffer();         //转换为 int 型缓冲
34          mNormalBuffer.put(normals);                  //向缓冲区中放入顶点着色数据
35          mNormalBuffer.position(0);                   //设置缓冲区起始位置
36          float textures[]=new float[alTexture.size()];
37          for(int i=0;i<alTexture.size();i++){
38            textures[i]=alTexture.get(i);
39          }
40          ByteBuffer tbb = ByteBuffer.allocateDirect(textures.length*4);
                                                         //创建顶点纹理数据缓冲
41          tbb.order(ByteOrder.nativeOrder());          //设置字节顺序
42          mTextureBuffer= tbb.asFloatBuffer();         //转换为 Float 型缓冲
43          mTextureBuffer.put(textures);                //向缓冲区中放入顶点着色数据
44          mTextureBuffer.position(0);                  //设置缓冲区起始位置
45      }
46      public void drawSelf(GL10 gl) {
47          gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);      //启用顶点坐标数组
48          gl.glEnableClientState(GL10.GL_NORMAL_ARRAY);      //启用顶点法向量数组
49          gl.glVertexPointer(3,GL10.GL_FLOAT,0,mVertexBuffer);
                                                         //为画笔指定顶点坐标数据
50          gl.glNormalPointer(GL10.GL_FLOAT, 0, mNormalBuffer);
                                                         //为画笔指定顶点法向量数据
51          gl.glEnable(GL10.GL_TEXTURE_2D);             //开启纹理
52          gl.glEnableClientState(GL10.GL_TEXTURE_COORD_ARRAY);
                                                         //允许使用纹理 ST 坐标缓冲
53          gl.glTexCoordPointer(2, GL10.GL_FLOAT, 0, mTextureBuffer);
                                                         //为画笔指定纹理 ST 坐标缓冲
54          gl.glBindTexture(GL10.GL_TEXTURE_2D, texId);          //绑定当前纹理
55          gl.glDrawArrays(GL10.GL_TRIANGLES,0,vCount );         //绘制图形
56          gl.glDisableClientState(GL10.GL_TEXTURE_COORD_ARRAY); //关闭纹理
57          gl.glDisable(GL10.GL_TEXTURE_2D);
58      }}
```

其中：

- 第 6～10 行为声明成员变量，包括顶点数据缓存对象、顶点着色数据缓存对象和顶点索引数据缓存对象和顶点数量记录对象声明，顶点索引数量记录对象声明，以及场景选择角度对象声明。
- 第 11～45 行为该类构造的开发，其中第 17～26 行为获取顶点数组并将其存入顶点数据缓存；第 27～35 行为获取顶点法向量数组并将其存入顶点法向量数据缓存；第 36～44 行为获取顶点纹理数据并将其存入顶点纹理数据缓存中。
- 第 46～58 行为绘制场景的方法。其中 49 行为场景指定顶点坐标数据；第 50 行为场景指定顶点法向量数据；第 53 行为场景指定顶点纹理数据，并绘制场景。

（4）接着向读者介绍墙壁绘制时顶点数据的设置方法，将下列代码插入到 Wall.java 的第 16 行。

代码位置：见随书光盘中源代码/第 14 章/GL_MASE/src/wyf/wpf/Wall.java

```
1     for(int i=0;i<rows;i++){
```

```
2            for(int j=0;j<cols;j++){  //对地图中的每一块进行处理
3                if(MAP[i][j]==1){     //若是可通过区域则要考虑围墙问题,先考虑此块上面
                                         是否需要围墙
4                    if(i==0||MAP[i-1][j]==0){//若是最上面一行或上面一块不可通过,则此
                                                块上面需要围墙
5                        float x1=j*UNIT_SIZE;float y1=FLOOR_Y;float z1=i*UNIT_SIZE;
6                        float x2=j*UNIT_SIZE;float y2=FLOOR_Y+WALL_HEIGHT;float
                         z2=i*UNIT_SIZE;
7                float  x3=(j+1)*UNIT_SIZE;float  y3=FLOOR_Y+WALL_HEIGHT;float
                 z3=i*UNIT_SIZE;
8                float x4=(j+1)*UNIT_SIZE;float y4=FLOOR_Y;float z4=i*UNIT_SIZE;
9                        alVertex.add(x1);alVertex.add(y1);alVertex.add
                         (z1);alVertex.add(x3);
10                       alVertex.add(y3);alVertex.add(z3);alVertex.add
                         (x2);alVertex.add(y2);
11                       alVertex.add(z2);alVertex.add(x1);alVertex.add
                         (y1);alVertex.add(z1);
12                       alVertex.add(x4);alVertex.add(y4);alVertex.add
                         (z4);alVertex.add(x3);
13                       alVertex.add(y3);alVertex.add(z3);
14                   alTexture.add(0f);alTexture.add(2f);alTexture.add(2f);
                     alTexture.add(0f);
15                   alTexture.add(0f);alTexture.add(0f);alTexture.add(0f);
                      alTexture.add(2f);
16                   alTexture.add(2f);alTexture.add(2f);alTexture.add(2f);
                      alTexture.add(0f);
17                       alNormal.add(0f);alNormal.add(0f);alNormal.add(1f);
                          alNormal.add(0f);
18                       alNormal.add(0f);alNormal.add(1f);alNormal.add(0f);
                          alNormal.add(0f);
19                       alNormal.add(1f);alNormal.add(0f);alNormal.add(0f);
                          alNormal.add(1f);
20                       alNormal.add(0f);alNormal.add(0f);alNormal.add(1f);
                          alNormal.add(0f);
21                        alNormal.add(0f);alNormal.add(1f);
22                   }
23                   if(i==rows-1||MAP[i+1][j]==0)  //再考虑此块下面是否需要围墙
24                   {         //若是最下面一行或下面一块不可通过，则此块下面需要围墙
25                   float x1=j*UNIT_SIZE;float y1=FLOOR_Y;float z1=(i+1)*UNIT_SIZE;
26               float x2=j*UNIT_SIZE;float y2=FLOOR_Y+WALL_HEIGHT;float z2=(i+1)
                 *UNIT_SIZE;
27               float  x3=(j+1)*UNIT_SIZE;float  y3=FLOOR_Y+WALL_HEIGHT;float
                 z3=(i+1)*UNIT_SIZE;
28               float x4=(j+1)*UNIT_SIZE;float y4=FLOOR_Y;float z4=(i+1)
                 *UNIT_SIZE;
29                       alVertex.add(x2);alVertex.add(y2);alVertex.add(z2);
                         alVertex.add(x3);
30                       alVertex.add(y3);alVertex.add(z3);alVertex.add(x1);
                         alVertex.add(y1);
31                       alVertex.add(z1);alVertex.add(x3);alVertex.add(y3);
                         alVertex.add(z3);
32                       alVertex.add(x4);alVertex.add(y4);alVertex.add(z4);
                         alVertex.add(x1);
33                       alVertex.add(y1);alVertex.add(z1);
34                   alTexture.add(0f);alTexture.add(0f);alTexture.add(2f);
                     alTexture.add(0f);
35                   alTexture.add(0f);alTexture.add(2f);alTexture.add(2f);
                     alTexture.add(0f);
36                   alTexture.add(2f);alTexture.add(2f);alTexture.add(0f);
                     alTexture.add(2f);
37                        alNormal.add(0f);alNormal.add(0f);alNormal.
                          add(-1f);alNormal.add(0f);
```

```
38                      alNormal.add(0f);alNormal.add(-1f);alNormal.add
                        (0f);alNormal.add(0f);
39                      alNormal.add(-1f);alNormal.add(0f);alNormal.add
                        (0f);alNormal.add(-1f);
40                      alNormal.add(0f);alNormal.add(0f);alNormal.add
                        (-1f);alNormal.add(0f);
41                      alNormal.add(0f);alNormal.add(-1f);
42                  }
43              if(j==0||MAP[i][j-1]==0)  //再考虑此块左面是否需要围墙
44                  {           //若是最左面一列或左面一块不可通过则此块左面需要围墙
45              float x1=j*UNIT_SIZE;float y1=FLOOR_Y;float z1=(i+1)*UNIT_SIZE;
46              float  x2=j*UNIT_SIZE;float  y2=FLOOR_Y+WALL_HEIGHT;float  z2=
                (i+1)*UNIT_SIZE;
47              float  x3=j*UNIT_SIZE;float  y3=FLOOR_Y+WALL_HEIGHT;float  z3=i
                *UNIT_SIZE;
48              float x4=j*UNIT_SIZE;float y4=FLOOR_Y;float z4=i*UNIT_SIZE;
49                      alVertex.add(x1);alVertex.add(y1);alVertex.add(z1);
                        alVertex.add(x3);
50                      alVertex.add(y3);alVertex.add(z3);alVertex.add(x2);
                        alVertex.add(y2);
51                      alVertex.add(z2);alVertex.add(x1);alVertex.add(y1);
                        alVertex.add(z1);
52                      alVertex.add(x4);alVertex.add(y4);alVertex.add(z4);
                        alVertex.add(x3);
53                      alVertex.add(y3);alVertex.add(z3);
54                  alTexture.add(0f);alTexture.add(2f);alTexture.add(2f);
                    alTexture.add(0f);
55                  alTexture.add(0f);alTexture.add(0f);alTexture.add(0f);
                    alTexture.add(2f);
56                  alTexture.add(2f);alTexture.add(2f);alTexture.add(2f);
                    alTexture.add(0f);
57                      alNormal.add(1f);alNormal.add(0f);alNormal.add(0f);
                        alNormal.add(1f);
58                      alNormal.add(0f);alNormal.add(0f);alNormal.add(1f);
                        alNormal.add(0f);
59                      alNormal.add(0f);alNormal.add(1f);alNormal.add(0f);
                        alNormal.add(0f);
60                      alNormal.add(1f);alNormal.add(0f);alNormal.add(0f);
                        alNormal.add(1f);
61                      alNormal.add(0f);alNormal.add(0f);
62                  }
63                  if(j==cols-1||MAP[i][j+1]==0)  //再考虑此块右面是否需要围墙
64                  {           //若是最右面一列或右面一块不可通过，则此块右面需要围墙
65              float x1=(j+1)*UNIT_SIZE;float y1=FLOOR_Y;float z1=(i+1)*UNIT_
                SIZE;
66              float x2=(j+1)*UNIT_SIZE;float y2=FLOOR_Y+WALL_HEIGHT;float z2=
                (i+1)*UNIT_SIZE;
67              float x3=(j+1)*UNIT_SIZE;float y3=FLOOR_Y+WALL_HEIGHT;float z3=
                i*UNIT_SIZE;
68              float x4=(j+1)*UNIT_SIZE;float y4=FLOOR_Y;float z4=i*UNIT_SIZE;
69                      alVertex.add(x2);alVertex.add(y2);alVertex.add(z2);
                        alVertex.add(x3);
70                      alVertex.add(y3);alVertex.add(z3);alVertex.add(x1);
                        alVertex.add(y1);
71                      alVertex.add(z1);alVertex.add(x3);alVertex.add(y3);
                        alVertex.add(z3);
72                      alVertex.add(x4);alVertex.add(y4);alVertex.add(z4);
                        alVertex.add(x1);
73                      alVertex.add(y1);alVertex.add(z1);
74                  alTexture.add(0f);alTexture.add(0f);alTexture.ad d(2f);
                    alTexture.add(0f);
75                  alTexture.add(0f);alTexture.add(2f);alTexture.add(2f);
```

```
                        alTexture.add(0f);
76                      alTexture.add(2f);alTexture.add(2f);alTexture.add(0f);
                        alTexture.add(2f);
77                          alNormal.add(-1f);alNormal.add(0f);alNormal.add
                            (0f);alNormal.add(-1f);
78                          alNormal.add(0f);alNormal.add(0f);alNormal.add
                            (-1f);alNormal.add(0f);
79                          alNormal.add(0f);alNormal.add(-1f);alNormal.add
                            (0f);alNormal.add(0f);
80                          alNormal.add(-1f);alNormal.add(0f);alNormal.add
                            (0f);alNormal.add(-1f);
81                          alNormal.add(0f);alNormal.add(0f);
82                  }}}}
```

提示：上述代码用于获取墙壁的绘制顶点数据。包括水平迷宫路线、竖直迷宫路线和拐角迷宫路线上的墙壁顶点数据。

（5）然后向读者介绍晶体类 TradPair.java 类的开发过程。

代码位置：见随书光盘中源代码/第 14 章/GL_MASE/src/wyf/wpf/TradPair.java

```
1   package wyf.wpf;                                          //包名
2   import java.nio.ByteBuffer;                               //导入相关包
3   ......// 该处省略了导入相关包的代码，读者可自行查阅随书光盘中源代码
4   import java.nio.FloatBuffer;                              //导入相关包
5   public class TradPair{                                    //表示单个晶体的类
6       private FloatBuffer  mVertexBuffer;                   //顶点坐标数据缓冲
7       private FloatBuffer  mNormalBuffer;                   //顶点法向量数据缓冲
8      private FloatBuffer  mTextureBuffer;                   //顶点着色数据缓冲
9      int vCount; int texId;                                 //纹理 Id
10     public TradPair(int texId) {
11      this.texId=texId; vCount=24;       //两个对立的四面体有 8 个三角形 24 个顶点
12        float vScale=0.6f; float hScale=0.15f;
13        float vertices[]=new float[]{                       //顶点数组
14      0,WALL_HEIGHT*vScale,0,                               //三角形 1
15          UNIT_SIZE*hScale,WALL_HEIGHT*vScale/2,UNIT_SIZE*hScale,
16          UNIT_SIZE*hScale,WALL_HEIGHT*vScale/2,-UNIT_SIZE*hScale,
17          0,WALL_HEIGHT*vScale,0,                           //三角形 2
18          UNIT_SIZE*hScale,WALL_HEIGHT*vScale/2,-UNIT_SIZE*hScale,
19          -UNIT_SIZE*hScale,WALL_HEIGHT *vScale/2,-UNIT_SIZE*hScale,
20           0,WALL_HEIGHT*vScale,0,                          //三角形 3
21          -UNIT_SIZE*hScale,WALL_HEIGHT*vScale/2,-UNIT_SIZE*hScale,
22          -UNIT_SIZE*hScale,WALL_HEIGHT*vScale/2,UNIT_SIZE*hScale,
23           0,WALL_HEIGHT*vScale,0,                          //三角形 4
24          -UNIT_SIZE*hScale,WALL_HEIGHT*vScale/2,UNIT_SIZE*hScale,
25          UNIT_SIZE*hScale,WALL_HEIGHT*vScale/2,UNIT_SIZE*hScale,
26           0,0,0,                                           //三角形 5
27          UNIT_SIZE*hScale,WALL_HEIGHT*vScale/2,-UNIT_SIZE*hScale,
28          UNIT_SIZE*hScale,WALL_HEIGHT*vScale/2,UNIT_SIZE*hScale,
29           0,0,0,                                           //三角形 6
30          -UNIT_SIZE*hScale,WALL_HEIGHT*vScale/2,-UNIT_SIZE*hScale,
31          UNIT_SIZE*hScale,WALL_HEIGHT*vScale/2,-UNIT_SIZE*hScale,
32           0,0,0,                                           //三角形 7
33          -UNIT_SIZE*hScale,WALL_HEIGHT*vScale/2,UNIT_SIZE*hScale,
34          -UNIT_SIZE*hScale,WALL_HEIGHT*vScale/2,-UNIT_SIZE*hScale,
35           0,0,0,                                           //三角形 8
36          UNIT_SIZE*hScale,WALL_HEIGHT*vScale/2,UNIT_SIZE*hScale,
37          -UNIT_SIZE*hScale,WALL_HEIGHT*vScale/2,UNIT_SIZE*hScale,
38        };
```

```
39          ByteBuffer vbb = ByteBuffer.allocateDirect(vertices.length*4);
                                                        //创建顶点坐标数据缓冲
40          vbb.order(ByteOrder.nativeOrder());         //设置字节顺序
41          mVertexBuffer = vbb.asFloatBuffer();        //转换为 int 型缓冲
42          mVertexBuffer.put(vertices);                //向缓冲区中放入顶点坐标数据
43          mVertexBuffer.position(0);                  //设置缓冲区起始位置
44          float normals[]=new float[] {               //法向量数据存放数组
45              1,0,0, 1,0,0, 1,0,0, 1,0,0, 0,0,-1, 0,0,-1, 0,0,-1, 0,0,-1,
46              -1,0,0, -1,0,0, -1,0,0, -1,0,0, 0,0,1, 0,0,1, 0,0,1, 0,0,1,
47              1,0,0, 1,0,0, 1,0,0, 1,0,0, 0,0,-1, 0,0,-1, 0,0,-1, 0,0,-1,
48              -1,0,0, -1,0,0, -1,0,0, -1,0,0,0,0,1, 0,0,1, 0,0,1, 0,0,1,
49          };
50          ByteBuffer nbb = ByteBuffer.allocateDirect(normals.length*4);
51          nbb.order(ByteOrder.nativeOrder());         //设置字节顺序
52          mNormalBuffer = nbb.asFloatBuffer();        //转换为 int 型缓冲
53          mNormalBuffer.put(normals);                 //向缓冲区中放入顶点着色数据
54          mNormalBuffer.position(0);                  //设置缓冲区起始位置
55          float textures[]=new float[] {              //纹理坐标存放数组
56            0.5f,0f,   0f,1f,   1f,1f,0.5f,0f,   0f,1f,   1f,1f,0.5f,0f,   0f,1f,
1f,1f,
57            0.5f,0f,   0f,1f,   1f,1f,0.5f,0f,   0f,1f,   1f,1f,0.5f,0f,   0f,1f,
1f,1f,
58            0.5f,0f,  0f,1f,  1f,1f,0.5f,0f,  0f,1f,  1f,1f,
59          };
60          ByteBuffer tbb = ByteBuffer.allocateDirect(textures.length*4);
                                                        //创建顶点纹理数据缓冲
61          tbb.order(ByteOrder.nativeOrder());         //设置字节顺序
62          mTextureBuffer= tbb.asFloatBuffer();        //转换为 Float 型缓冲
63          mTextureBuffer.put(textures);               //向缓冲区中放入顶点着色数据
64          mTextureBuffer.position(0);                 //设置缓冲区起始位置
65      }
66      public void drawSelf(GL10 gl){
67          gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);      //启用顶点坐标数组
68          gl.glEnableClientState(GL10.GL_NORMAL_ARRAY);      //启用顶点法向量数组
69          gl.glVertexPointer(3,GL10.GL_FLOAT,0,mVertexBuffer);
                                                        //为画笔指定顶点坐标数据
70          gl.glNormalPointer(GL10.GL_FLOAT, 0, mNormalBuffer);
                                                        //为画笔指定顶点法向量数据
71          gl.glEnable(GL10.GL_TEXTURE_2D);                  //开启纹理
72          gl.glEnableClientState(GL10.GL_TEXTURE_COORD_ARRAY);
                                                        //允许使用纹理 ST 坐标缓冲
73          gl.glTexCoordPointer(2, GL10.GL_FLOAT, 0, mTextureBuffer);
                                                        //为画笔指定纹理 ST 坐标缓冲
74          gl.glBindTexture(GL10.GL_TEXTURE_2D, texId);      //绑定当前纹理
75          gl.glDrawArrays(GL10.GL_TRIANGLES, 0,vCount );    //绘制图形
76      }}
```

其中：

- 第 6～9 行为声明成员变量，包括顶点数据缓存对象、顶点着色数据缓存对象和顶点索引数据缓存对象，以及顶点数量记录对象声明、顶点索引数量记录对象声明和场景选择角度对象声明。
- 第 10～65 行为该类构造的开发，其中第 13～43 行为获取顶点数组并将其存入顶点数据缓存；第 44～54 行为获取顶点法向量数组并将其存入顶点法向量数据缓存；第 55～64 行为获取顶点纹理数据，并将其存入顶点纹理数据缓存中。
- 第 66～76 行为绘制场景的方法。其中第 69 行为场景指定顶点坐标数据；第 70 行为场

景指定顶点法向量数据；第 73 行，为场景指定顶点纹理数据并绘制场景。

（6）下面向读者介绍晶体数量标志板类 TextureRect.java 类的开发过程。

代码位置：见随书光盘中源代码/第 14 章/GL_MASE/src/wyf/wpf/TextureRect.java

```
1   package wyf.wpf;                                          //包名
2   import java.nio.ByteBuffer;                               //导入相关包
3   ......// 该处省略了导入相关包的代码，读者可自行查阅随书光盘中源代码
4   import java.nio.FloatBuffer;                              //导入相关包
5   public class TextureRect {                                //表示单个纹理矩形的类
6       private FloatBuffer  mVertexBuffer;                   //顶点坐标数据缓冲
7     private FloatBuffer  mTextureBuffer;                    //顶点着色数据缓冲
8     int vCount;int texId;
9     public TextureRect(int texId,float X_UNIT_SIZE,float Y_UNIT_SIZE,float[]
textures){
10     this.texId=texId;vCount=6;
11       float vertices[]=new float[] {                  //顶点数据存放数组
12         -1*X_UNIT_SIZE,1*Y_UNIT_SIZE,0,-1*X_UNIT_SIZE,-1*Y_UNIT_SIZE,0,
13         1*X_UNIT_SIZE,1*Y_UNIT_SIZE,0,-1*X_UNIT_SIZE,-1*Y_UNIT_SIZE,0,
14         1*X_UNIT_SIZE,-1*Y_UNIT_SIZE,0,1*X_UNIT_SIZE,1*Y_UNIT_SIZE,0
15       };
16       ByteBuffer vbb = ByteBuffer.allocateDirect(vertices.length*4);
                                                            //创建顶点坐标数据缓冲
17       vbb.order(ByteOrder.nativeOrder());                //设置字节顺序
18       mVertexBuffer = vbb.asFloatBuffer();               //转换为 int 型缓冲
19       mVertexBuffer.put(vertices);                  //向缓冲区中放入顶点坐标数据
20       mVertexBuffer.position(0);                         //设置缓冲区起始位置
21       ByteBuffer tbb = ByteBuffer.allocateDirect(textures.length*4);
                                                      //创建顶点纹理数据缓冲
22       tbb.order(ByteOrder.nativeOrder());                //设置字节顺序
23       mTextureBuffer= tbb.asFloatBuffer();               //转换为 Float 型缓冲
24       mTextureBuffer.put(textures);                 //向缓冲区中放入顶点着色数据
25       mTextureBuffer.position(0);                        //设置缓冲区起始位置
26    }
27    public void drawSelf(GL10 gl) {
28       gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);      //启用顶点坐标数组
29       gl.glVertexPointer(3,GL10.GL_FLOAT,0,mVertexBuffer);
                                                      //为画笔指定顶点坐标数据
30       gl.glEnable(GL10.GL_TEXTURE_2D);                   //开启纹理
31       gl.glEnableClientState(GL10.GL_TEXTURE_COORD_ARRAY);
                                                      //允许使用纹理 ST 坐标缓冲
32       gl.glTexCoordPointer(2, GL10.GL_FLOAT, 0, mTextureBuffer);
                                                      //为画笔指定纹理 ST 坐标缓冲
33       gl.glBindTexture(GL10.GL_TEXTURE_2D, texId);       //绑定当前纹理
34       gl.glDrawArrays(GL10.GL_TRIANGLES, 0,vCount );     //绘制图形
35       gl.glDisableClientState(GL10.GL_TEXTURE_COORD_ARRAY); //关闭纹理
36       gl.glDisable(GL10.GL_TEXTURE_2D);
37    }}
```

其中：

- 第 6～8 行为声明成员变量，包括顶点数据缓存对象、顶点着色数据缓存对象和顶点索引数据缓存对象和顶点数量记录对象声明，顶点索引数量记录对象声明，以及场景选择角度对象声明。
- 第 9～26 行为该类构造的开发，其中第 11～20 行为获取顶点数组并将其存入顶点数据缓存；第 21～25 行为获取顶点纹理数据并将其存入顶点纹理数据缓存中。
- 第 27～37 行为绘制场景的方法。其中 29 行为场景指定顶点坐标数据；第 32 行为场景

指定顶点纹理数据，并绘制场景。

实例 10　游戏中的逻辑实现与线程操控

【实现过程】

在本小节中，将向读者介绍在 3D 迷宫游戏中对所有晶体的控制类的开发，所获取的晶体数量的数据的绘制，以及键盘或虚拟摇杆的线程操控类的具体实现。

【代码解析】

（1）首先向读者介绍的是所获取的晶体数量的数据的绘制类的开发，代码如下。

代码位置：见随书光盘中源代码/第 14 章/GL_MASE/src/wyf/wpf/Score.java

```
1   package wyf.wpf;                                          //包名
2   import javax.microedition.khronos.opengles.GL10;          //包引用
3   import static wyf.wpf.Constant.*;
4   public class Score {                                      //表示获得晶体数量的类
5       MySurfaceView mv;
6       TextureRect[] numbers=new TextureRect[10];            //数字位图对象数组
7       public Score(int texId,MySurfaceView mv){
8           this.mv=mv;
9           for(int i=0;i<10;i++){                    //生成 0～9 十个数字的纹理矩形
10              numbers[i]=new TextureRect(
11               texId, ICON_WIDTH*0.7f/2, ICON_HEIGHT*0.7f/2,
12               new float[] {                                //给出纹理坐标
13                    0.1f*i,0, 0.1f*i,1, 0.1f*(i+1),0,
14                    0.1f*i,1, 0.1f*(i+1),1,  0.1f*(i+1),0
15                    } );
16          }}
17      public void drawSelf(GL10 gl){
18          String scoreStr=mv.objectCount+"";
19          for(int i=0;i<scoreStr.length();i++){  //将得分中的每个数字字符绘制
20              char c=scoreStr.charAt(i);
21              gl.glPushMatrix();
22              gl.glTranslatef(i*ICON_WIDTH*0.7f, 0, 0);   //平移操作
23              numbers[c-'0'].drawSelf(gl);                //绘制
24              gl.glPopMatrix();
25          }}}
```

其中：

- 第 5～6 行为成员变量的声明，包括界面类对象和数字纹理数组对象。
- 第 7～16 行为构造器部分，在此给出了当前数字所对应图片的纹理坐标。第 17～25 行为绘制当前数字位图。

（2）上述代码介绍绘制类的开发，接下来介绍的是游戏中对所有晶体的控制类的开发，代码如下。

代码位置：见随书光盘中源代码/第 14 章/GL_MASE/src/wyf/wpf/TradPairGroup.java

```
1   package wyf.wpf;                                  //包名
2   import static wyf.wpf.Constant.*;                 //包引用
3   import javax.microedition.khronos.opengles.GL10;
4   public class TradPairGroup {                      //表示晶体组的类
5       TradPair tp;                                  //晶体
6       float yAngle=0;                               //晶体沿 Y 轴旋转的角度
```

```
7       boolean flag=true;                                  //是否旋转的标记
8       int[][] objectMap;                                  //晶体位置地图
9       public TradPairGroup(int texId){
10          tp=new TradPair(texId);                         //创建晶体对象
11          int rows=MAP_OBJECT.length;                     //获取地图的长度
12          int cols=MAP_OBJECT[0].length;                  //获取地图的宽带
13          objectMap=new int[rows][cols];                  //创建地图对应的二维数组
14          for(int i=0;i<rows;i++){
15              for(int j=0;j<cols;j++){
16                  objectMap[i][j]=MAP_OBJECT[i][j];
17              }}
18          new Thread(){                                   //将晶体绕 Y 轴旋转的线程
19              public void run(){                          //重写 run()方法
20                  while(flag){
21                      yAngle+=5.0;                        //每次晶体旋转 5°
22                      if(yAngle>=360){
23                          yAngle=0;
24                      }
25                      try {
26                          Thread.sleep(100);              //每 100 毫秒旋转一次
27                      } catch (InterruptedException e) {
28                          e.printStackTrace();            //异常处理
29                      }}}}.start();                       //开启线程
30      }
31      public void drawSelf(GL10 gl){
32          int rows=objectMap.length;
33          int cols=objectMap[0].length;
34          for(int i=0;i<rows;i++){  //扫描晶体位置地图的每个格子，若此格有晶体则绘制
35              for(int j=0;j<cols;j++){
36                  if(objectMap[i][j]==1){
37                      gl.glPushMatrix();//保护现场
38  gl.glTranslatef((j+0.5f)*UNIT_SIZE, FLOOR_Y+0.45f, (i+0.5f)*UNIT_SIZE);
                                                            //移动晶体到此格对应的位置
39                      gl.glRotatef(yAngle, 0, 1, 0);      //将晶体绕 Y 轴旋转
40                      tp.drawSelf(gl);                    //绘制晶体
41                      gl.glPopMatrix();                   //恢复现场
42                  }}}}}
```

提示：第 10～17 行为获取地图中的所有晶体。第 18～29 行为开启一个使所有晶体都自动旋转的线程。第 31～42 行为绘制当前场景中的所有晶体。

（3）上述代码介绍的是控制类的开发，接下来介绍的是键盘或虚拟摇杆线程操控类的开发，代码如下。

代码位置：见随书光盘中源代码/第 14 章/GL_MASE/src/wyf/wpf/KeyThread.java

```
1   package wyf.wpf;                                        //包名
2   import static wyf.wpf.Constant.*;                       //包引用
3   public class KeyThread extends Thread{                  //键盘监听线程
4     MySurfaceView mGLSurfaceView; static int keyState=0;
                                                  //0-没有键按下 1-上 2-下 3-左 4-右
5     public KeyThread(MySurfaceView mGLSurfaceView) {      //构造器
6         this.mGLSurfaceView=mGLSurfaceView;
7     }
8     public void run(){                                    //重写 run()方法
9         while(threadWork) {
10            try{
11              Thread.sleep(40);                   //每 40 毫秒检测一次
```

```
            }
            catch (InterruptedException e)  {
                  e.printStackTrace();                    //异常处理
            }
            if(keyState==0) {                             //当前运动状态为静止状态
                continue;
            }
            try {
          if(keyState==1||keyState==2){                    //上下键表示前进或后退
              int tempDirection=0;
              switch(keyState){
                case 1:                                   //向上键代表前进
          tempDirection=mGLSurfaceView.currentDirection;
                                                          //前进时运动方向与当前方向相同
                  break;
                case 2:                                   //向下键代表后退
          tempDirection=(mGLSurfaceView.currentDirection+2)%4;
                                                          //后退时运动方向与当前方向相反
                  break;
              }
          ... ...//按上下键后，玩家第一人称视角人物的运动状态分析，详细内容在下面介绍。
      }
          else{
              switch(keyState){
                  case 3:                                 //向左代表左转身
                      mGLSurfaceView.currentDirection=(mGLSurfaceView.
currentDirection-1+4)%4; 86                    break;
                  case 4:                                 //向右代表右转身
                      mGLSurfaceView.currentDirection=(mGLSurfaceView.
currentDirection+1)%4;
                  break;
              }}
          mGLSurfaceView.heroXT=mGLSurfaceView.heroX+
              MOVE_XZ[mGLSurfaceView.currentDirection][0];
                                                          //设置新的观察目标点 XZ 坐标
  mGLSurfaceView.heroZT=mGLSurfaceView.heroZ+MOVE_XZ[mGLSurfaceView.
currentDirection][1];
          if(keyState==3||keyState==4){
              Thread.sleep(200);                          //睡眠 200 毫秒
          }}
            catch(Exception e) {
                e.printStackTrace();                      //异常处理
            }}}}
```

提示：在该线程中，主要用于判断第一人称视角人物的运动状态，并根据其运动状态来判断其是否碰撞到墙壁，是否获取晶体，是前进还是后退，是左转还是右转。

（4）接下来向读者介绍的是，按上下键后，玩家第一人称视角人物的运动状态分析，将下列代码插入到上述代码的第 30 行。

代码位置：见随书光盘中源代码/第 14 章/GL_MASE/src/wyf/wpf/KeyThread.java

```
1   mGLSurfaceView.heroX=mGLSurfaceView.heroX+MOVE_XZ[tempDirection][0];
                                                          //计算运动后的 XZ 值
2       mGLSurfaceView.heroZ=mGLSurfaceView.heroZ+MOVE_XZ[tempDirection][1];
3                   boolean backFlag=false;               //恢复标记
4                   float tempX=mGLSurfaceView.heroX-HALF_COLL_SIZE;
                                                          //左上侧点 XY 坐标
5                   float tempZ=mGLSurfaceView.heroZ-HALF_COLL_SIZE;
```

```
6                   int tempCol=(int)((tempX/UNIT_SIZE>=0)?(tempX/UNIT_SIZE):-1);
                                                          //左上侧点行列
7                   int tempRow=(int)((tempZ/UNIT_SIZE>=0)?(tempZ/UNIT_SIZE):-1);
8                   if(tempCol==-1||tempCol==MAP[0].length|| tempRow==-1||temp
Row==MAP.length||
9                     MAP[tempRow][tempCol]==0) {           //碰撞则设置恢复标记
10                   backFlag=true;
11                  }
12                  tempX=mGLSurfaceView.heroX+HALF_COLL_SIZE;   //右上侧点XY坐标
13                  tempZ=mGLSurfaceView.heroZ-HALF_COLL_SIZE;
14                  tempCol=(int)((tempX/UNIT_SIZE>=0)?(tempX/UNIT_SIZE):-1);
                                                          //右上侧点行列
15                  tempRow=(int)((tempZ/UNIT_SIZE>=0)?(tempZ/UNIT_SIZE):-1);
16                  if(tempCol==-1||tempCol==MAP[0].length||tempRow==-1||temp
Row==MAP.length||
17                    MAP[tempRow][tempCol]==0){            //碰撞则设置恢复标记
18                   backFlag=true;
19                  }
20                  tempX=mGLSurfaceView.heroX-HALF_COLL_SIZE;   //左下侧点XY坐标
21                  tempZ=mGLSurfaceView.heroZ+HALF_COLL_SIZE;
22                  tempCol=(int)((tempX/UNIT_SIZE>=0)?(tempX/UNIT_SIZE):-1);
                                                          //左下侧点行列
23                  tempRow=(int)((tempZ/UNIT_SIZE>=0)?(tempZ/UNIT_SIZE):-1);
24                  if(tempCol==-1||tempCol==MAP[0].length||tempRow==-1||
tempRow==MAP.length||
25                    MAP[tempRow][tempCol]==0) {           //碰撞则设置恢复标记
26                   backFlag=true;
27                  }
28                  tempX=mGLSurfaceView.heroX+HALF_COLL_SIZE;   //右下侧点XY坐标
29                  tempZ=mGLSurfaceView.heroZ+HALF_COLL_SIZE;
30                  tempCol=(int)((tempX/UNIT_SIZE>=0)?(tempX/UNIT_SIZE):-1);
                                                          //右下侧点行列
31                  tempRow=(int)((tempZ/UNIT_SIZE>=0)?(tempZ/UNIT_SIZE):-1);
32                  if(tempCol==-1||tempCol==MAP[0].length||tempRow==-1||temp
Row==MAP.length||
33                    MAP[tempRow][tempCol]==0){            //碰撞则设置恢复标记
34                   backFlag=true;
35                  }
36                  if(backFlag){                           //不允许移动则恢复
37
    mGLSurfaceView.heroX=mGLSurfaceView.heroX-MOVE_XZ[tempDirection][0];
38
    mGLSurfaceView.heroZ=mGLSurfaceView.heroZ-MOVE_XZ[tempDirection][1];
39                  }
40                  tempCol=(int)(mGLSurfaceView.heroX/UNIT_SIZE);
                                                          //计算当前行和列
41                  tempRow=(int)(mGLSurfaceView.heroZ/UNIT_SIZE);
42                  if(mGLSurfaceView.mRenderer.tpg!=null&&mGLSurfaceView.
mRenderer.tpg
43                   .objectMap[tempRow][tempCol]==1){      //碰到晶体
44                   MazeActivity ma=(MazeActivity)mGLSurfaceView.getContext();
45                   ma.playSound(1, 0);                    //播放吃的声音
46                   mGLSurfaceView.mRenderer.tpg.objectMap[tempRow][tempCol]
=0;//消除此处晶体
47                   mGLSurfaceView.objectCount++;          //已吃数量加1
48                   if(mGLSurfaceView.objectCount==OBJECT_COUNT){       //赢了
49                       MazeActivity.mpBack.pause();       //暂停播放游戏背景音
50                       MazeActivity.mpWin.start();        //播放赢的背景音
51                       ma.hd.sendEmptyMessage(MazeActivity.GAME_WIN);
52                       mGLSurfaceView.mRenderer.tpg.flag=false;  //不再旋转晶体
```

```
53                          }}
```

提示：上述代码内容主要是计算在特定状态下，第一人称视角人物下一步的运动状态，同时判断其获取的晶体数量是否达到过关的要求，进而判断游戏是否胜利。

实例 11　游戏地图设计器的界面效果与使用方法

【实现过程】

在本小节中，主要向读者介绍游戏中迷宫的地图数据的开发实现方法。在其中主要是对地图设计器技术的应用。下面首先介绍地图设计器的运行效果与具体应用方法。

(1)运行本游戏地图设计器软件后，首先进入的界面为 3D 迷宫寻宝地图设计器界面如图 14-15 所示，在该界面中，可以设置地图的长度和宽度，其大小单位为一个固定大小的立方体空间。

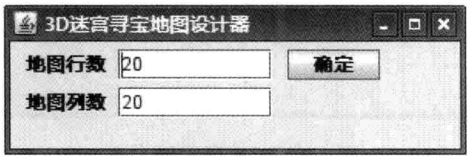


图 14-15　地图设计器行列设置界面

(2) 设置好地图的大小后，单击“确定”按钮，则会打开地图设计器的主界面（如图 14-16 所示），在主界面中包括“生成地图”按钮、“生成晶体位置”按钮、“生成摄像机位置”按钮，这三个按钮均用于返回地图设计器生成的数据。在该界面中，还包括“黑色”、“白色”、“晶体”、“摄像机”四个单项按钮，用于选择当前所有设置的对象。

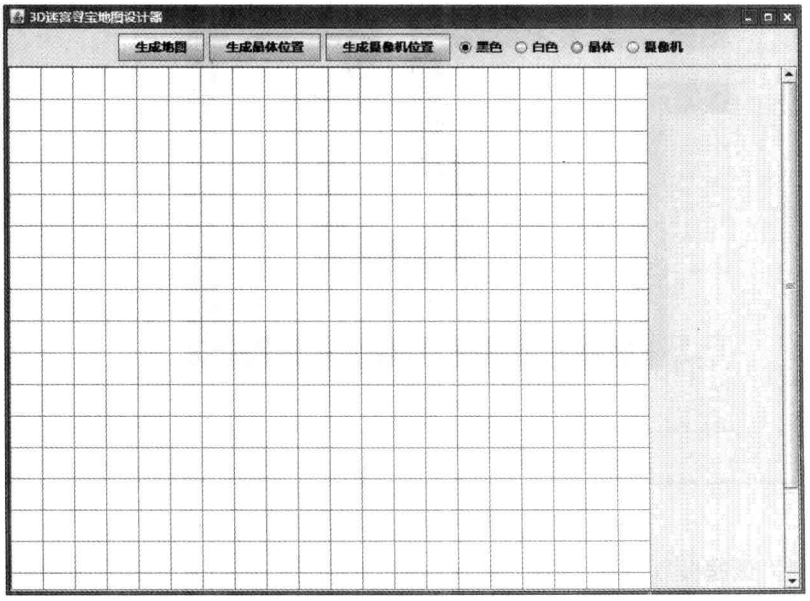


图 14-16　地图设计器主界面

(3) 在主界面中，选中“黑色”，用户可以通过用鼠标滑动的方式在格子范围中拖拉出自

已设计的迷宫路线图，如图 14-17 所示。若用户想修改路线，可选中“白色”，将错误路线擦除。

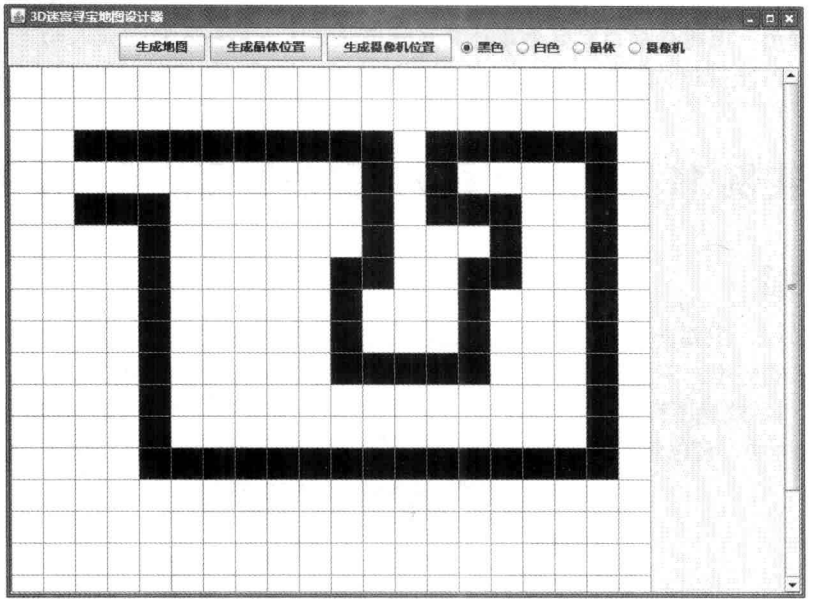


图 14-17　地图路线设置

（4）选中“晶体”，用户可以通过鼠标单击的方式，在黑色路线区域放置晶体部件，如图 14-18 所示。

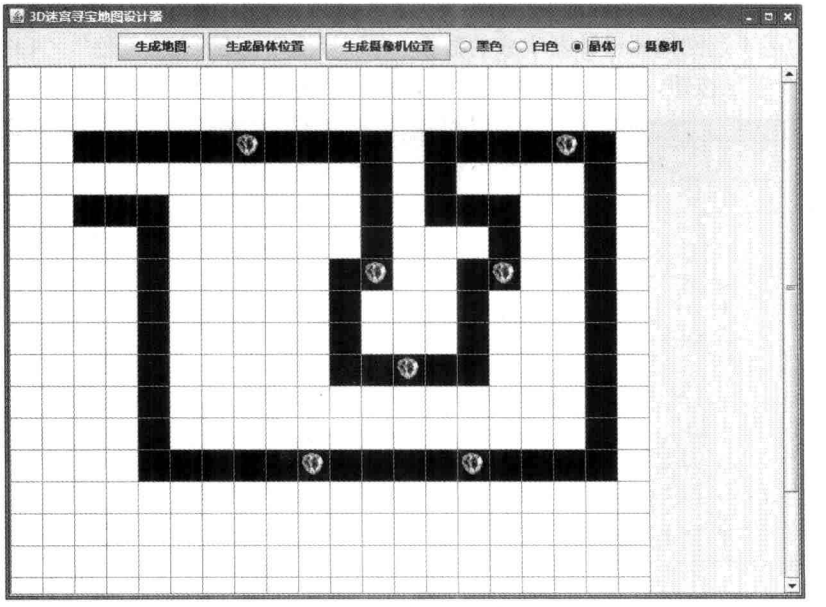


图 14-18　添加晶体到地图

（5）选中“摄像机”，用户可以通过鼠标单击的方式，来设置人物游戏的初始位置，如图 14-19 所示。

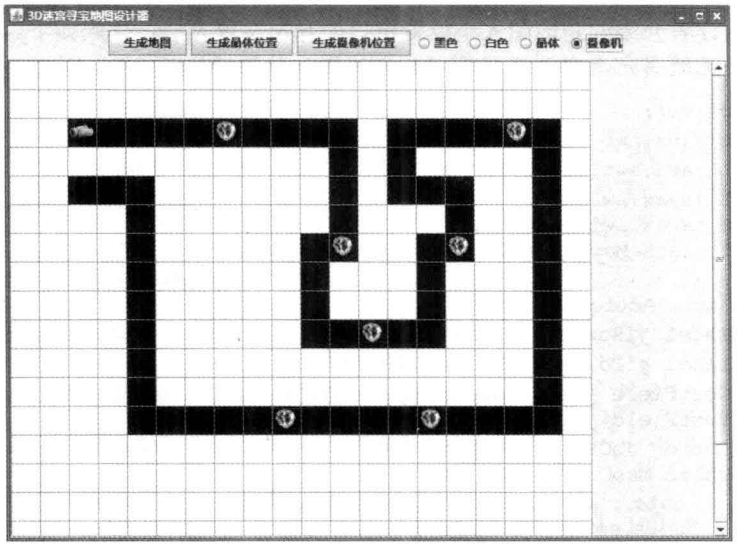


图 14-19　指定摄像机初始位置

（6）当用户对于地图设置完成后，可通过单击“生成地图”按钮来获取地图路线数据数组信息，单击“生成晶体位置”来获取晶体摆放位置的数据数组信息，单击“生成摄像机位置”来获取人物游戏初始位置的坐标数据信息，如图 14-20 所示。

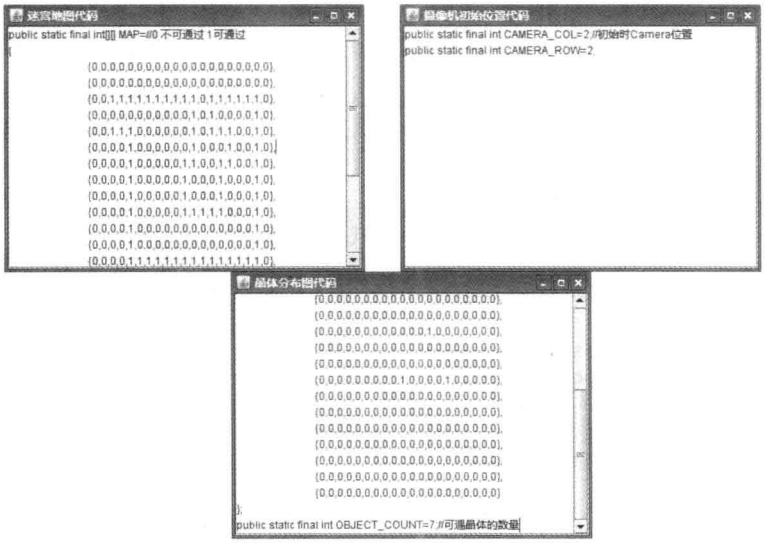


图 14-20　获取地图设计器返回的数据

将这些信息放入到游戏项目中的常量类 Constant.java 类中，即可实现新的迷宫路线的设计。

实例 12　游戏地图设计器的开发实现

【代码解析】

在本小节中，将向读者介绍地图设计器软件的具体开发过程和实现方法。

（1）首先向读者介绍的是地图大小设置窗口的开发实现方法，代码如下。

代码位置：见随书光盘中源代码/第 14 章/地图设计器/ MapColRowDialog.java

```
1   package wyf;                                          //包名
2   import java.awt.*;                                    //包引用
3   import java.awt.event.*;                              //包引用
4   import javax.swing.*;                                 //包引用
5   import javax.swing.event.*;
6   public class MapColRowDialog extends JFrame
                                          //继承自 JFrame，并实现了 ActionListener 接口
7   implements ActionListener{
8       JLabel jlRow=new JLabel("地图行数");              //创建 JLabel 控件对象
9       JLabel jlCol=new JLabel("地图列数");
10      JTextField jtfRow=new JTextField("20");           //创建 JTextField 控件对象
11      JTextField jtfCol=new JTextField("20");
12      JButton jbOk=new JButton("确定");                 //创建 JButton 控件对象
13      public MapColRowDialog(){
14          this.setTitle("3D 迷宫寻宝地图设计器");       //设置窗口标题
15          this.setLayout(null);
16          jlRow.setBounds(10,5,60,20);                  //指定地图行数位置
17          this.add(jlRow);
18          jtfRow.setBounds(70,5,100,20);                //指定地图列数位置
19          this.add(jtfRow);
20          jlCol.setBounds(10,30,60,20);                 //指定文本控件位置
21          this.add(jlCol);
22          jtfCol.setBounds(70,30,100,20);               //指定文本控件位置
23          this.add(jtfCol);
24          jbOk.setBounds(180,5,60,20);                  //指定确定按钮位置
25          this.add(jbOk);
26          jbOk.addActionListener(this);                 //为确定按钮添加监听器
27          this.setBounds(440,320,300,100);              //设置窗口的位置和大小
28          this.setVisible(true);                        //使窗口可见
29      }
30      public void actionPerformed(ActionEvent e){
31          int row=Integer.parseInt(jtfRow.getText().trim());
                                                          //获取行数文本中的数据值
32          int col=Integer.parseInt(jtfCol.getText().trim());
                                                          //获取列数文本中的数据值
33          new MapDesigner(row,col);                     //创建 MapDesigner.java 对象
34          this.dispose();
35      }
36      public static void main(String args[]){          //main 方法
37          new MapColRowDialog();
38      }}
```

其中：

- 第 8～12 行表示创建实现界面所应用到的 Swing 控件对象，主要包括 JLabel、JTextField 和 JButton 控件。
- 第 13～29 行为构造器部分。在其中实现了界面的搭建工作，使界面产生如图 14-15 所示效果。
- 第 30～35 行为 JButton 控件添加单击监听器。从中获取地图的长度和宽带，并创建 MapDesigner.java 类对象用于实现地图设计器主界面的创建。
- 第 36～38 行为地图设计器软件的 main 方法，是软件运行的开始。

（2）然后向读者介绍地图设计器主界面窗口的开发实现方法，代码如下。

代码位置：见随书光盘中源代码/第 14 章/地图设计器/ MapDesigner.java

```
1   package wyf;                                                    //包名
2   import java.awt.*;                                              //包引用
3   import java.awt.event.*;                                        //包引用
4   import javax.swing.*;                                           //包引用
5   import javax.swing.event.*;                                     //包引用
6   public class MapDesigner extends JFrame
7   implements ActionListener{
8       int row;int col;MapDesignPanel mdp;JScrollPane jsp;      //声明成员变量
9       JButton jbGenerate=new JButton("生成地图");              //创建 JButton 控件对象
10      JButton jbGenerateD=new JButton("生成晶体位置");         //创建 JButton 控件对象
11      JButton jbGenerateC=new JButton("生成摄像机位置");//创建 JButton 控件对象
12      JRadioButton jrBlack=new JRadioButton("黑色",null,true);
                                                                    //创建 JRadioButton 控件对象
13      JRadioButton jrWhite=new JRadioButton("白色",null,false);
14      JRadioButton jrCrystal=new JRadioButton("晶体",null,false);
                                                                    //创建 JButton 控件对象
15      JRadioButton jrCamera=new JRadioButton("摄像机",null,false);
                                                                    //创建 JButton 控件对象
16      ButtonGroup bg=new ButtonGroup();Image icrystal;Image iCamera;
                                                                    //创建 ButtonGroup 对象
17      JPanel jp=new JPanel();                                     //创建画布对象
18      public MapDesigner(int row,int col){
19          this.row=row;this.col=col;this.setTitle("3D 迷宫寻宝地图设计器");
                                                                    //获取地图的长度和宽度
20          icrystal=new ImageIcon("img/Diamond.png").getImage();
                                                                    //创建晶体 ImageIcon 控件
21          iCamera=new ImageIcon("img/camera.png").getImage();
                                                                    //创建摄像机 ImageIcon 控件
22          mdp=new MapDesignPanel(row,col,this);jsp=new JScrollPane(mdp);this.
                                                                        add(jsp);
23          jp.add(jbGenerate);jp.add(jbGenerateD);jp.add(jbGenerateC);
                                                                    //在界面中添加控件
24          jp.add(jrBlack);bg.add(jrBlack);
25          jp.add(jrWhite);bg.add(jrWhite);                        //在界面中添加控件
26          jp.add(jrCrystal);bg.add(jrCrystal);                    //在界面中添加控件
27          jp.add(jrCamera);bg.add(jrCamera);                      //在界面中添加控件
28          this.add(jp,BorderLayout.NORTH);                        //在界面中添加控件
29          jbGenerate.addActionListener(this);                     //在界面中添加控件
30          jbGenerateD.addActionListener(this);
31          jbGenerateC.addActionListener(this);                    //在界面中添加控件
32          this.setBounds(10,10,800,600);                          //设置窗口的大小和位置
33          this.setVisible(true);                                  //设置窗口为可见
34          this.mdp.requestFocus(true);                            //获取焦点
35      }
36      public void actionPerformed(ActionEvent e){                 //重写的方法
37         if(e.getSource()==this.jbGenerate) {                     //生成地图代码
38          String s="public static final int[][] MAP= //0 不可通过，1 可通过\n{";
39              for(int i=0;i<mdp.row;i++){                         //for 循环
40                  s=s+"\n\t{";                                    //字符串 s
41                  for(int j=0;j<mdp.col;j++){                     //for 循环
42                      s=s+mdp.mapData[i][j]+",";                  //字符串 s
43                  }
44                  s=s.substring(0,s.length()-1)+"},";             //字符串 s
45              }
46              s=s.substring(0,s.length()-1)+"\n};";new CodeFrame(s,"迷宫地图代码");
```

```
47          }
48          else if(e.getSource()==this.jbGenerateD){              //生成晶体代码
49           String s="public static final int[][] MAP_OBJECT=
                                                    //表示可遇晶体位置的矩阵\n{";
50              int ccount=0;                                      //ccount 值为零
51              for(int i=0;i<mdp.row;i++){                        //for 循环
52                  s=s+"\n\t{";                                   //字符串 s
53                  for(int j=0;j<mdp.col;j++){                    //for 循环
54                      s=s+mdp.diamondMap[i][j]+",";              //字符串 s
55                      if(mdp.diamondMap[i][j]==1){   //判断是否符合条件
56                          ccount++;                              //ccount 值加 1
57                      }}
58                  s=s.substring(0,s.length()-1)+"},";            //字符串
59              }
60              s=s.substring(0,s.length()-1)+"\n};";              //截取 s 字符串
61              s=s+"\npublic static final int OBJECT_COUNT=" +ccount+";
62              new CodeFrame(s,"晶体分布图代码");                  //new 对象
63          }
64           else if(e.getSource()==this.jbGenerateC){             //生成摄像机位置
65              String s="public static final int CAMERA_COL="+this.mdp.cameraCol+";
66                                      +"public static final int CAMERA_ROW= “+
67                                      "+this.mdp.cameraRow+";";
                                                    //初始时 Camera 位置\n"
68              new CodeFrame(s,"摄像机初始位置代码");              //new 对象
69           }}
70      public static void main(String args[]){
71          new MapColRowDialog();
72      }}
```

其中：

- 第 9～17 行表示创建 Swing 控件对象，用于构建窗口。包括 JButton、JRadioButton、ButtonGroup 以及 JPanel。
- 第 18～35 行表示该类的构造器部分。在其中为窗口添加控件，并设置窗口为可见。
- 第 36～72 行为按钮添加单击监听器。当单击“生成地图”按钮，获取地图迷宫路线数据；单击“生成摄像机位置”按钮，获取地图中摄像机初始位置数据；单击“生成晶体位置”按钮，获取地图中晶体的初始位置数据。

（3）接下来介绍地图设计器中显示获取数据信息的窗口的开发实现方法。

代码位置：见随书光盘中源代码/第 14 章/地图设计器/ CodeFrame.java

```
1   package wyf;                                          //包名
2   import java.awt.*;                                    //包引用
3   import java.awt.event.*;
4   import javax.swing.*;                                 //包引用
5   import javax.swing.event.*;                           //包引用
6   public class CodeFrame extends JFrame{                //窗口类继承自 JFrame.java 类
7       JTextArea jta=new JTextArea();                    //创建 JTextArea 控件对象
8       JScrollPane jsp=new JScrollPane(jta);             //创建 JScrollPane 控件对象
9       public CodeFrame(String codeStr,String title){
10          this.setTitle(title);                         //设置窗口标题
11          this.add(jsp);                                //添加控件
12          jta.setText(codeStr);                         //设置 JTextArea 中的显示内容
13          this.setBounds(100,100,400,300);              //设置窗口的大小和位置
14          this.setVisible(true);                        //设置窗口为可见
15      }}
```

提示：该窗口用于显示获取数据，并可通过将获取数据植入到迷宫项目的 Constant.java 类中，则可创建出新的迷宫路线。

（4）最后向读者介绍地图设计器的功能实现类的开发实现方法。

代码位置：见随书光盘中源代码/第 14 章/地图设计器/ MapDesignPanel.java

```
1   package wyf;                                                    //包名
2   import java.awt.*;import java.awt.event.*;                      //包引用
3   import javax.swing.*;import javax.swing.event.*;
4   public class MapDesignPanel extends JPanel      //继承自 JPanel.java 类
5   implements MouseListener,MouseMotionListener{
6       int row;int col;int span=32;MapDesigner md; //声明 MapDesigner.java 类对象
7       int[][] mapData; int[][] diamondMap;int cameraCol;int cameraRow;
8       boolean cameraFlag=false;
9       public MapDesignPanel(int row,int col,MapDesigner md){
10          this.row=row;this.col=col;this.md=md;  //获取地图列数和地图行数
11          this.setPreferredSize(
12              new Dimension(span*col,span*row)                    //创建地图
13          );
14          mapData=new int[row][col];                              //创建地图数据数组
15          for(int i=0;i<row;i++){                                 //for 循环
16              for(int j=0;j<col;j++){                             //for 循环
17                  mapData[i][j]=0;                                //设置值为零
18              }}
19          diamondMap=new int[row][col];                           //赋值
20          for(int i=0;i<row;i++){                                 //for 循环
21              for(int j=0;j<col;j++){                             //for 循环
22                  diamondMap[i][j]=0;                             //赋值为零
23              }}
24          this.addMouseListener(this);this.addMouseMotionListener(this);
                                                                    //添加鼠标监听器
25      }
26      public void paint(Graphics g){                              //设置画笔方法
27          g.setColor(Color.BLACK);g.fillRect(0,0,span*col,span*row);
28          for(int i=0;i<mapData.length;i++){                      //循环
29              for(int j=0;j<mapData[0].length;j++){               //循环
30                  if(mapData[i][j]==0){                           //绘制白色格子
31
g.setColor(Color.white);g.fillRect(j*span,i*span,span,span);
32                  }}}
33          for(int i=0;i<diamondMap.length;i++){                   //循环
34              for(int j=0;j<diamondMap[0].length;j++){            //循环
35                  if(diamondMap[i][j]==1){                        //绘制晶体
36                      g.drawImage(md.icrystal,j*span+1,i*span+3,this);
37                  }}}
38          if(this.cameraFlag)                                     //绘制摄像机
39          g.drawImage(md.iCamera,cameraCol*span+1,cameraRow*span+3,this);
40          g.setColor(Color.green);                                //设置颜色
41          for(int i=0;i<row+1;i++){                               //循环
42              g.drawLine(0,span*i,span*col,span*i);               //绘制横向线
43          }
44          for(int j=0;j<col+1;j++){                               //循环
45              g.drawLine(span*j,0,span*j,span*row);               //绘制纵向线
46          }}
47      public void mouseClicked(MouseEvent e){                    //声明方法
48          if(md.jrBlack.isSelected()||md.jrWhite.isSelected()){
```

```
                                                        //设置地图可通过性
49              int x=e.getX();int y=e.getY();int rowC=y/span;int colC=x/span;
50              if(rowC>=row||colC>=col){                //判断是否符合条件
51                  return;
52              }
53              mapData[rowC][colC]=(mapData[rowC][colC]+1)%2;  //赋值
54              if(mapData[rowC][colC]==0){              //清除晶体与摄像机
55                  diamondMap[rowC][colC]=0;this.cameraFlag=false; //赋值
56              }}
57          else if(md.jrCrystal.isSelected()){          //摆放晶体
58              int x=e.getX();int y=e.getY();int rowC=y/span;int colC=x/span;
59              if(rowC>=row||colC>=col){                //判断是否符合条件
60                  return;
61              }
62              if(mapData[rowC][colC]==0){              //判断是否为零
63                  JOptionPane.showMessageDialog(this,"不可通过处不能摆放晶体！",
64                      "摆放错误",JOptionPane.ERROR_MESSAGE);   //显示对话框
65                  return;                              //返回
66              }
67              diamondMap[rowC][colC]=(diamondMap[rowC][colC]+1)%2;//赋值
68          }
69          else if(md.jrCamera.isSelected()){           //摆放摄像机
70              int x=e.getX();int y=e.getY();int rowC=y/span;int colC=x/span;
                                                        //赋值
71              if(mapData[rowC][colC]==0){              //判断是否为零
72                  JOptionPane.showMessageDialog(this,
73                      "不可通过处不能摆放摄像机！","摆放错误", //对话框内容
74                      JOptionPane.ERROR_MESSAGE        //类型
75                  );                                   //显示对话框
76                  return;                              //返回
77              }
78              this.cameraCol=colC;this.cameraRow=rowC;this.cameraFlag=true;
79          }
80          this.repaint();                              //调用 repaint 方法
81      }
82      public void mousePressed(MouseEvent e){}         //重写的方法
83      public void mouseReleased(MouseEvent e){}        //重写的方法
84      public void mouseEntered(MouseEvent e){}         //重写的方法
85      public void mouseExited(MouseEvent e){}          //重写的方法
86      public void mouseDragged(MouseEvent e){          //重写的方法
87          if(md.jrBlack.isSelected()||md.jrWhite.isSelected()){
                                                        //判断是否符合条件
88              int x=e.getX();int y=e.getY();int rowC=y/span;int colC=x/span;
89              if(rowC>=row||colC>=col){                //判断是否符合条件
90                  return;                              //返回
91              }
92              mapData[rowC][colC]=md.jrBlack.isSelected()?1:0;
93              if(mapData[rowC][colC]==0){              //清除晶体与摄像机
94                  diamondMap[rowC][colC]=0;this.cameraFlag=false;
95              }}
96          this.repaint();                              //重绘
97      }
98      public void mouseMoved(MouseEvent e){}           //重写的方法
99      public static void main(String args[]){          //main 方法
100         new MapColRowDialog();
101     }}
```

其中：

- 第 9～25 行为构造器部分，为初始化主界面。设置当前初始界面中路线为空，晶体数量为空，摄像机为空。
- 第 47～81 行为鼠标单击事件监听器方法。用于实现迷宫路线的绘制，晶体的添加和摄像机的设置。

实例 13　游戏的优化与改进

【实现过程】

本章主要是对 3D 迷宫的开发。本软件具有较为复杂的逻辑结构，并且在开发过程中用了地图设计器，可以非常轻松地设计迷宫地图。通过本章的学习，读者应该对 3D 游戏开发有一定的了解。并且自己可以开发较为简单的 3D 游戏。

基于前面章节对本游戏的介绍，游戏的基本功能已经开发完毕，但仍有很多方面可以进行优化和改进，有能力的读者可以继续对本游戏进行提升，可以提升的地方如下。

1. 模式的定制

游戏应该开发成多种模式，不同的模式带来不一样的体验，单一的模式会让游戏很快失去对玩家的吸引力。同时应该添加多人模式的选择功能，使玩家能根据自己的喜好选择不同的模式进行游戏，这样才更加合理。

2. 过关奖励

每完成一次任务，或者结束一次挑战为游戏添加相应的奖励，如金钱奖励制度、荣誉制度等，玩家可以根据自己持有的金钱和荣誉度购买迷宫帮助指示灯来指明一小段路线的正确方向，游戏的可玩度会得到更近一步的提高。

3. 增设迷宫路线

可以增设多种错综复杂的迷宫路线，游戏的可玩性也能大大提高。