

The
Pragmatic
Programmers

TURING 图灵程序设计丛书

Hello, Android

Introducing Google's Mobile Development Platform

Android基础教程

[美] Ed Burnette 著
张波 高朝勤 杨越 等译



你的第一本Android书

Pragmatic系列图书品质保证

从这里，开始一个新的梦想



人民邮电出版社

POSTS & TELECOM PRESS

更多资源请访问我的新浪博客 <http://blog.sina.com.cn/ckook>

TURING 图灵程序设计丛书

Hello, Android

Introducing Google's Mobile Development Platform

Android基础教程

【美】Ed Burnette 著
张波 高朝勤 杨越 等译



人民邮电出版社
北京

更多资源请访问我的新浪博客 <http://blog.sina.com.cn/ckook>

图书在版编目 (CIP) 数据

Android 基础教程 / (美) 伯内特 (Burnette, E.) 著;
张波等译. —北京: 人民邮电出版社, 2009.11
(图灵程序设计丛书)

书名原文: Hello, Android: Introducing Google's
Mobile Development Platform
ISBN 978-7-115-21536-9

I. ① A… II. ① 伯…② 张… III. ① 移动通信—携带
电话机—应用程序—程序设计—教材 IV. ① TN929.53

中国版本图书馆CIP数据核字 (2009) 第178807号

内 容 提 要

Android 是谷歌公司开发的全新开源手机平台。本书是一部关于 Android 开发的基础教程, 采用由浅入深、循序渐进的方式讨论 Android。书中还结合数独游戏等实例更加形象生动地讲解了 Android 开发的基本流程, 且每章最后都有一个“快速阅读指南”, 更加方便了读者的阅读。

本书内容完整丰富, 具有较强的通用性, 读者都能通过本书快速学习 Android 开发, 提高相关技能。

图灵程序设计丛书

Android基础教程

-
- ◆ 著 [美] Ed Burnette
 - 译 张 波 高朝勤 杨 越 等
 - 责任编辑 傅志红
 - 执行编辑 印星星
 - ◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街14号
邮编 100061 电子函件 315@ptpress.com.cn
网址 <http://www.ptpress.com.cn>
北京艺辉印刷有限公司印刷
 - ◆ 开本: 800×1000 1/16
印张: 13
字数: 270千字 2009年11月第1版
印数: 1~4 000册 2009年11月北京第1次印刷
 - 著作权合同登记号 图字: 01-2009-3809号
ISBN 978-7-115-21536-9
-

定价: 39.00元

读者服务热线: (010)51095186 印装质量热线: (010)67129223

反盗版热线: (010)67171154

版 权 声 明

Copyright © 2008 Ed Burnette. Original English language edition, entitled *Hello, Android: Introducing Google's Mobile Development Platform*.

Simplified Chinese-language edition copyright © 2009 by Posts & Telecom Press. All rights reserved.

本书中文简体字版由The Pragmatic Programmers, LLC授权人民邮电出版社独家出版。未经出版者书面许可，不得以任何方式复制或抄袭本书内容。

版权所有，侵权必究。

对本书的赞誉

这本书极其出色，不仅文笔流畅、浅显易懂，内容也妙趣横生。本书既恰到好处地讲解了Android独有的特性，同时也突出了高质量编程的原则。

——Anthony Stevens

PocketJourney创始人兼CTO，Google Android竞赛前20强

Ed Burnette的这本书虽然篇幅不长，但内容丰富，保持了Pragmatic（实用）系列图书的一贯风格。仅凭2D和3D图形方面的内容，本书就非常值得所有Android开发人员拥有。

——Mark Murphy

CommonsWare创始人

我还记得第一次使用Android时的情景：当时感觉它就像是一座超大型迷宫。有了这本书，入门就不会那么痛苦了。我深信，通过阅读本书，上手开发Android应用程序将是一件非常轻松愉快的事。

——Gabor Paller

OnRelay公司高级软件架构师

前言

Android是一款针对手机的全新开源软件工具包，它由Google和开放手机联盟（Open Handset Alliance）共同创建。Android有望在数年内遍布于数百万部手机和其他移动设备中，从而成为应用程序开发人员的主要平台。无论你是业余爱好者还是专业程序员，无论你是自己玩玩还是为了盈利，都应该了解关于Android开发的更多信息。本书将帮助你迅速入门。

Android 的特别之处

如今，市场上已经有了许多移动平台，包括Symbian、iPhone、Windows Mobile、BlackBerry、Java Mobile Edition和Linux Mobile（LiMo）等。当我向别人说起Android时，他们的第一个疑问通常是：我们为什么还需要另一个移动标准？它有何惊人之处？

虽然Android的一些特性并非首创，但它是第一个将以下特性结合在一起的环境。

- ❑ 基于Linux，真正开放、开源、免费的发展平台。手持设备制造商钟情于它的原因，是他们可以使用和定制该平台而不需要支付版税。开发人员喜欢它的原因，是他们知道这个平台是独立的，不受任何一家厂商的限制。
- ❑ 受Internet mashup思想启发的基于组件的架构。一个应用程序的组件可以在另一个应用程序中用作其他用途。你甚至可以将Android内置的组件替换为自己改进后的版本。这将在移动领域掀起新一轮的创造风潮。
- ❑ 众多开箱即用的内置服务。基于位置的服务使用GPS或手机发射塔三角测量法，让你可根据所处位置来定制用户体验。凭借功能全面的SQL数据库，利用强大的本地存储，可以完成偶尔连接的计算和同步操作。浏览器和地图视图可以直接嵌入在应用程序中。所有这些内置服务有助于提高功能的标准，同时降低开发成本。
- ❑ 应用程序生命周期的自动化管理。多层安全措施将程序彼此分离，这将使智能电话的系统稳定性达到前所未有的水平。最终用户不再需要担心哪些应用程序是活动的，也不必在运行新程序前先关闭原有的一些程序。Android针对低能耗、低内存的设备进行了优化，这种根本性的优化是之前的平台从未尝试过的。

- 高质量的图形和声音。将类似于Flash的光滑、无锯齿的2D矢量图形和动画与3D加速的OpenGL图形相结合,可实现各种新式的游戏和商业应用程序。Android内置了最常用的行业标准音频和视频格式的编解码器,这些格式包括H.264 (AVC)、MP3和AAC。
- 当前及未来各类硬件间的可移植性。所有程序都是用Java语言编写的,并且将由Android的Dalvik虚拟机执行,所以代码在ARM、x86和其他架构之间是可以移植的。Android提供了对各种输入方法的支持,比如说键盘、触摸屏和轨迹球。用户界面可以针对任何屏幕分辨率和屏幕方向进行定制。

Android为用户与移动应用程序交互提供了全新的方式,同时也提供了实现这些交互的底层技术保障。而Android最令人心动之处,莫过于你可以为它编写软件,本书恰好可以为你提供这方面的帮助。

本书读者对象

阅读本书唯一的前提条件,是具备对Java编程或类似面向对象语言(比如说C#)的基本理解,不需要拥有为移动设备开发软件的经验。实际上,如果你确实有这方面的经验,反倒应该忘记它们。Android是如此与众不同,因此最好不要带着成见来学习它。

本书内容

本书分为三部分。大致来说,本书采用由浅入深、循序渐进的方式讨论Android。

有些章使用了一个公共的示例:Android数独游戏。通过逐渐在游戏中添加特性,你将学习Android编程的许多方面,包括用户界面、多媒体和Android生命周期。

第一部分中将首先介绍Android,内容涉及如何安装Android模拟器,如何使用IDE(Integrated Development Environment,集成开发环境)编写第一个程序。然后,我们将介绍一些基本的概念,比如Android中的生命周期。Android中的编程方式可能与你之前采用的方式不同,因此一定要在继续学习之前掌握这些概念。

第二部分讨论Android的用户界面、二维图形、多媒体组件以及简单的数据访问。这些特性在大多数程序中都用得到。

第三部分深入探讨Android平台。这一部分介绍外部通信、基于位置的服务、内置SQLite数据库和三维图形。

本书最后提供了一个附录,其中列出了Android与Java SE (Java Standard Edition, Java标准版)

之间的不同之处。

在线资源

本书网站<http://pragprog.com/titles/eband>提供了以下资源。

- 本书使用的所有示例程序的完整源代码；
- 勘误页面，列出了本书这一版中的所有错误（希望它保持空白）；
- 论坛，在此你可以直接与作者及其他Android开发人员交流（希望论坛用户越来越多）。

读者可以在自己的应用程序中随意使用源代码。

关于“快速阅读指南”

虽然大多数作者都希望读者阅读他们书中的每一句话，但我知道你可能不想这样做。你只希望阅读能够解决手头问题的部分，而在需要解决其他问题时，再回过头来阅读另外一些内容。因此，我在书中特意注明在哪里可以找到你所关心的内容。

本书每章最后都有一个“快速阅读指南”，告诉无序阅读本书的读者接下来应该阅读哪些内容。读者还可以在发现一些指向相关资源（如图书和在线文档）的链接，可以了解相关主题的更多信息。

好吧，你现在想了解点什么？第1章就将指导你完成第一个Android程序。第2章回过头来介绍Android的基本概念和原理。第3章探讨用户界面，也就是大多数Android程序中最重要的一部分。

致谢

我要感谢为本书成功出版做出贡献的许多人，包括审稿人Anthony Stevens、Gabor Paller、Fred Burke、Dianne Hackborn和Laurent Pontier，他们详尽审阅了本书；感谢编辑Susannah Pfalzer在我几乎要推迟交稿时提供的好建议并为我鼓足勇气。特别要感谢我的家人，感谢他们在我写作本书期间表现出来的极大耐心。

目 录

第一部分 Android 简介

第1章 快速入门	3
1.1 安装工具	3
1.1.1 Java 5.0+	3
1.1.2 Eclipse	4
1.1.3 Android	4
1.1.4 Eclipse插件	5
1.2 创建第一个程序	7
1.3 在模拟器上运行程序	8
1.4 在手机上运行程序	9
1.5 快速阅读指南	9
第2章 基本概念	11
2.1 Android的系统架构	11
2.1.1 Linux内核	11
2.1.2 本机库	12
2.1.3 Android运行时	13
2.1.4 应用程序框架	14
2.1.5 应用程序	15
2.2 它还活着	15
2.2.1 进程不等于应用程序	16
2.2.2 应用程序生命周期	17
2.3 构建块	19
2.3.1 活动	19
2.3.2 意图	19
2.3.3 服务	19
2.3.4 内容提供者	19
2.4 使用资源	20
2.5 安全性	20
2.6 快速阅读指南	21

第二部分 Android 基础知识

第3章 设计用户界面	25
3.1 数独游戏简介	25
3.2 声明性设计	26
3.3 创建启动界面	27
3.4 使用替代资源	34
3.5 实现About对话框	37
3.6 应用主题	41
3.7 添加菜单	43
3.8 添加设置	45
3.9 开始新游戏	47
3.10 利用日志消息调试程序	48
3.11 利用调试器调试程序	50
3.12 退出游戏	50
3.13 快速阅读指南	50
第4章 绘制 2D 图形	53
4.1 Android图形基础	53
4.1.1 Color类	53
4.1.2 Paint类	54
4.1.3 Canvas类	55
4.1.4 Path类	55
4.1.5 Drawable类	56
4.2 在Sudoku程序中添加图形	58
4.2.1 开始游戏	58
4.2.2 定义Game类	58
4.2.3 定义PuzzleView类	60
4.2.4 绘制游戏盘面	61
4.2.5 绘制数字	63
4.3 处理输入	65
4.3.1 定义和更新选定区域	66

4.3.2 输入数字	68
4.3.3 增加提示	69
4.3.4 抖动屏幕	70
4.4 其他问题	71
4.4.1 创建软键盘	76
4.4.2 实现游戏逻辑	78
4.4.3 其他功能	80
4.5 更多改进	81
4.6 快速阅读指南	83
第5章 多媒体	83
5.1 播放音频	88
5.2 播放视频	92
5.3 为数独游戏配上音乐	94
5.4 快速阅读指南	95
第6章 存储本地数据	95
6.1 为数独游戏添加选项	97
6.2 继续玩前一个游戏	99
6.3 记住当前位置	100
6.4 访问内部文件系统	101
6.5 访问SD卡	103
6.6 快速阅读指南	103

第三部分 高级主题

第7章 互联的世界	107
7.1 通过意图实现浏览	108
7.2 利用视图打开网页	111
7.3 JavaScript与Java通信	115
7.4 使用Web服务	121
7.5 快速阅读指南	131
第8章 定位与环境感知	133
8.1 位置, 位置, 位置	133
8.1.1 我在哪里	135
8.1.2 更新位置	137
8.1.3 模拟说明	138
8.2 充分利用传感器	139
8.2.1 了解传感器	139
8.2.2 解析传感器的读数	140
8.2.3 模拟说明	140

8.3 地图功能	141
8.3.1 嵌入MapView	142
8.3.2 准备就绪	145
8.3.3 模拟说明	146
8.4 快速阅读指南	147
第9章 SQL 实战	149
9.1 SQLite简介	149
9.2 SQL基础	150
9.2.1 DDL语句	151
9.2.2 修改语句	151
9.2.3 查询语句	151
9.3 你好, 数据库	152
9.3.1 使用SQLiteOpenHelper	153
9.3.2 定义主程序	155
9.3.3 添加一行	156
9.3.4 运行一个查询	157
9.3.5 显示查询结果	158
9.4 数据绑定	159
9.5 使用ContentProvider	162
9.5.1 更改主程序	164
9.5.2 添加一行	164
9.5.3 运行一个查询	165
9.6 实现ContentProvider	165
9.7 快速阅读指南	166

第10章 利用 OpenGL 实现 3D 图形	169
10.1 理解3D图形	169
10.2 OpenGL简介	170
10.3 构建一个OpenGL程序	171
10.4 管理线程	173
10.5 构建一个模型	178
10.6 光线、相机	181
10.7 动作	183
10.8 应用纹理	184
10.9 透明效果	187
10.10 快速阅读指南	189

第四部分 附录

附录 A Java 与 Android 语言及其 API	193
附录 B 参考书目	197

Part 1

第一部分

Android 简介

本部分 内容

- 第1章 快速入门
- 第2章 基本概念

第1章

快速入门

1

Android将手机的普及性、开源软件的活跃性，以及谷歌和其他开放手机联盟成员（如英特尔、TI、T-Mobile和NTT DoCoMo^①）的集体支持完美地结合在一起。最终的成果就是一个你不可不学的移动平台。

幸运的是，开发简单的Android程序很容易，甚至无需使用Android手机，只要准备一台可安装Android SDK和手机模拟器的计算机就可以了。

本章介绍如何安装所有开发工具，然后演示如何创建一个可运行的应用程序：Android版本的“Hello, World”。

1.1 安装工具

Android软件开发包（SDK）可在Windows、Linux和Mac OS X上运行。当然，你所创建的应用程序也可部署在任意Android设备上。

开始编写程序之前，需要安装Java、IDE和Android SDK。

1.1.1 Java 5.0+

首先需要安装Java，所有的Android开发工具都需要它，编写Android程序也需要使用Java语言。需要使用JDK 5或6。

只有一个运行时环境（JRE）还不够，还需要完整的开发包。我建议从Sun下载

^① 联盟成员也包括中国移动、中国联通、华为、宏碁等。——编者注

网站^①下载并安装最新的Sun JDK 6.0更新。Mac OS X用户应该从苹果公司的网站下载并安装最新版本的Mac OS X和JDK。

要确定已安装的版本是否正确，可在命令行窗口中运行以下命令。下面是运行该命令时显示的结果：

```
C:\> java -version
java version "1.6.0_02"
Java(TM) SE Runtime Environment (build 1.6.0_02-b05)
Java HotSpot(TM) Client VM (build 1.6.0_02-b05, mixed mode)
```

在计算机上运行该命令后，你应该能看到类似的输出内容，其中版本应该是“1.6.其他数字”或更高。

1.1.2 Eclipse

接下来应该安装Java开发环境（如果计算机上没有该环境）。我建议安装Eclipse，因为这个软件是免费的，并且创建Android的谷歌开发人员也使用它并且支持它。

如果不想使用Eclipse（肯定有人不想用），也可使用其他IDE，如NetBeans和JetBrains IDEA（相应的社区都有提供）。如果你是个很守旧的人，可以完全忘记IDE，仅使用命令行工具^②。

必须使用版本3.3.1以上的Eclipse，但是应该始终使用最新的可用版本。注意，所需的不仅仅是标准的Eclipse SDK“经典”平台。请访问Eclipse下载页^③，然后选择Eclipse IDE for Java Developers。按照该页上的说明下载该Eclipse，然后进行解压并在合适的位置（例如在Windows计算机上是C:\Eclipse）安装Eclipse。

1.1.3 Android

然后，从谷歌网站下载最新的Android SDK。Android下载页^④上提供了用于Windows、Mac OS X和Linux的软件包。下载合适的软件包之后，将.zip文件解压缩到适当的目录中（例如C:\Google）。

① 参见网页<http://java.sun.com/javase/downloads>。

② 参见网页<http://androidappdocs.appspot.com/guide/developing/tools/index.html>，查看有关命令行工具的文档。

③ 参见网页<http://www.eclipse.org/downloads>。

④ 参见网页http://androidappdocs.appspot.com/sdk/1.6_r1/index.html。

默认情况下,会将该SDK展开到一个子目录中,如androidsdk-windows-1.6_r1。这是SDK的安装目录。记下该目录的完整路径,便于日后参考。

Pulse提供的强大支持

要想更轻松地下载Eclipse,可以尝试访问由Genuitec赞助的新Pulse网站(<http://www.poweredbypulse.com>)。在这个网站中,只需单击几次即可下载Android SDK。与常规的Eclipse.org下载内容相比,Pulse的下载内容更小,下载速度也更快,并且不会出现大量让人困惑的.zip文件。但是,如果你的计算机受到公司防火墙的保护,那么访问这个网站时可能会遇到问题。

不需要使用特殊的安装程序,下一步就是启动Eclipse并对其进行配置。

1.1.4 Eclipse 插件

为了让开发过程更轻松,谷歌还编写了一款称为ADT(Android Development Toolkit, Android开发工具包)的Eclipse插件。要想安装该插件,请执行以下步骤(注意,这些步骤说明适用于Eclipse 3.4,如果使用其他Eclipse版本,那么菜单和选项可能稍有不同)。

- (1) 启动Eclipse,然后选择Help > Software Updates...
- (2) 单击Available Software选项卡(如果尚未选中该选项卡)。
- (3) 单击Add Site...按钮。
- (4) 输入Android更新站点的位置:<https://dl-ssl.google.com/android/eclipse/>。

输入完成后,该对话框应该如图1-1所示。单击OK按钮。

(5) 现在,该Android站点应该出现在Available Software视图中。选中该站点旁边的复选框,然后单击Install...。如果出现错误消息,原因可能是Eclipse的版本不对。我强烈建议使用预编译的Eclipse IDE for Java或Eclipse IDE for Java EE开发软件包3.4以上版本。

如果安装Eclipse时使用了自定义安装方法,那么要想使用Android编辑器,还需要安装WST(Web Standard Tools, Web标准工具)插件及其必需的其他插件。

参见Web工具平台主页^①，了解详细信息和下载链接。这些内容已经内置在前面建议使用的软件包中。

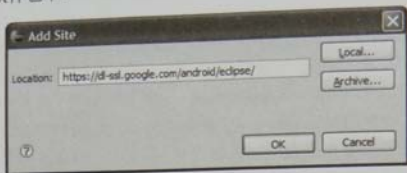


图1-1 安装Android开发工具包

(6) 单击Finish开始下载和安装过程。

(7) 安装完成后，重新启动Eclipse。

(8) Eclipse启动后你可能会看到几条错误消息，因为此时需要告诉它Android SDK位于何处。选择Window > Preferences > Android，然后输入你在前面记下的SDK安装目录。单击OK按钮。

幸运的是，上述步骤只需执行一次（或者说每次安装新版本的ADT或Eclipse后至少需要执行上述步骤一次）。现在，所需的内容都已安装好，可以编写第一个程序了。



小乔爱问……

出现Connection Error后怎么办

如果出现Connection Error（连接错误），那么最有可能的原因是系统管理员建立了某种防火墙。要想访问防火墙外的网络，需要使用代理服务器的地址来配置Eclipse。这个代理服务器与你的网络浏览器使用的代理服务器相同，但糟糕的是，Eclipse的智能不够高，不能从网络浏览器处获得代理服务器设置。

要想在Eclipse中输入代理服务器设置，可选择Window > Preferences > Network Connections，选中Manual代理服务器配置选项，输入服务器名称和端口号，然后单击OK按钮。如果看不到该选项，可能是运行的Eclipse版本较低。在Preferences > Install/Update下看看是否有相应的选项，或者搜索带有proxy字样的首选项。

^① 参见网页<http://www.eclipse.org/webtools>。

1.2 创建第一个程序

ADT附带了一个内置的示例程序，即模板，我们要使用这个模板快速地创建一个简单的“Hello, Android”程序。准备好秒表。准备就绪？归零？开始计时！

选择File > New > Android Project，打开New Project对话框。然后选择Android > Android Project，单击Next按钮。

输入如下信息：

Project name: Hello
Package name: org.example.hello
Activity name: Hello
Application name: Hello, Android

输入完成后，对话框应该如图1-2所示。

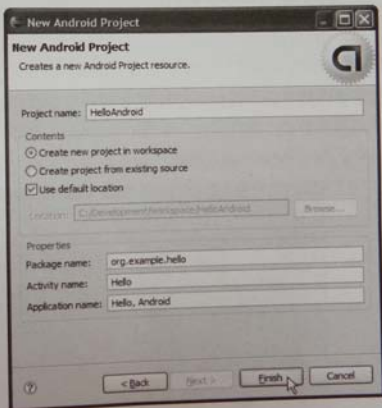


图1-2 新建Android项目

单击Finish按钮，Android插件将创建该项目，并在该项目中添加一些默认的文件。Eclipse将会生成该项目并将该项目内容打包，然后就可以执行该项目了。

程序的编写已经完成，现在剩下的就是运行该程序了。

1.3 在模拟器上运行程序

要想运行Android程序,请进入Package Explorer窗口,右键单击HelloAndroid项目,然后选择Run As > Android Application。此时会打开Android模拟器窗口并启动Android操作系统。第一次执行上述操作时,可能需要一两分钟的时间,所以请耐心等待。如果出现错误消息,显示应用程序没有响应,请选择继续等待的选项。

保持“开机”

启动模拟器需要花较长时间。可以这样想象一下——首次开机时,手机也需要启动,就像任何计算机系统一样。关闭模拟器就像是关闭手机或取出手机电池一样。因此,不要关闭它,免得再次启动又要等候。

只要Eclipse仍在运行,就要保持模拟器窗口处于打开状态。下次启动Android程序时,Eclipse会注意到模拟器已经启动了,所以只会向模拟器发送要运行的新程序。

模拟器窗口打开后,Eclipse会向模拟器发送要执行的程序的一个副本。然后会出现应用程序界面,并且运行“Hello, Android”程序(参见图1-3)。这就够了!恭喜你完成了第一个Android程序。



图1-3 正在运行的“Hello, Android”程序

1.4 在手机上运行程序

开发期间，在实际的设备（如T-Mobile G1）上运行Android程序与在模拟器上运行该程序的效果几乎相同。需要做的就是用USB电缆连接手机与计算机，并安装一个特殊的设备驱动程序^①。如果模拟器窗口已打开，请将其关闭。只要将手机与计算机相连，应用程序就会在手机上加载并运行。

在向外界发布应用程序之前，需要获得一个加密密钥并使用该密钥签名软件包。网上提供了有关该过程的详细说明^②。支付少量的注册费后，就可将你的程序上传到Android Market^③。让你的应用程序出现在Market上是你的最终目标。世界各地数以百万计的用户都可在此看到你的工作成果，对你的程序进行评级，甚至掏钱购买它。

1.5 快速阅读指南

有了Eclipse插件，只需几秒钟即可创建一个最简单的Android程序。在第3章中，我们将通过一个实际的应用程序（数独游戏）来扩展上面的简单程序。在很多章中，我们都会使用这个游戏示例演示Android API的使用。

但是在深入了解这个示例之前，应该花些时间阅读第2章。当你真正理解了像活动和生命周期这样的基本概念时，理解其他内容也就很容易了。

虽然使用Eclipse开发Android程序只是多种选择中的一种，但是我强烈建议使用这种方法。如果你以前从未使用过Eclipse，可能需要买本书来看看，如*Eclipse IDE Pocket Guide* [Bur05]。

① 在<http://androidappdocs.appspot.com/guide/developing/device.html>可以找到设备驱动程序以及安装说明。

② 参见网页http://androidappdoc.appspot.com/guide/publishing/app_signing.html。

③ 参见网页<http://market.android.com>。

现在，读者已经了解了Android是什么，接下来我们再看看它是如何工作的。你可能对Android的某些部分很熟悉，如Linux内核、OpenGL和SQL数据库，但对其他部分可能还一无所知，如Android中应用程序生命周期的概念。

要编写出功能良好的Android应用程序，需要很好地理解本章介绍的这些基本概念。所以，如果你只想阅读本书中的一章，则非本章莫属。

2.1 Android 的系统架构

我们首先看看Android的总体系统架构——组成Android开源软件栈的关键层和组件。图2-1展示了完整的Android系统构架图，请读者仔细研究一下这张图，明天我们要考你的哦。

图中的每一层都使用下面各层所提供的服务。以下几节主要介绍Android中的各个层（从最下面的层开始）。

2.1.1 Linux 内核

Android构建在一个稳定且得到广泛认可的基础之上：Linux内核。1991年，还是赫尔辛基大学学生的Linus Torvalds开发了Linux内核。现在，Linux可以说是无所不在，从手表到超级计算机中都能找到它的身影。Linux为Android提供了硬件抽象层，以便将来把Android移植到更多的平台上。

从内部来看，Android使用Linux完成其内存管理、进程管理、网络和其他操作

系统服务工作。Android手机用户永远也不会看到Linux，程序也不会直接进行Linux调用。但是作为开发人员，你需要知道Linux在Android中的用途。



图2-1 Android系统架构

开发期间需要的某些实用程序要和Linux打交道。例如，adb shell命令^①将打开一个Linux命令行窗口，从中可以输入要在设备上运行的其他命令。例如，可以通过这个命令行窗口来检查Linux文件系统、查看活动的进程等。

2.1.2 本机库

Linux内核层上面的一层中包含了Android的本机库。这些共享库都是用C或C++语言编写的，并且针对电话使用的特定硬件架构进行了编译，并已由手机制造商预先安装到手机中。

其中最重要的一些库包括下面5个部分。

- 界面管理器 (Surface Manager)。Android使用与Vista或Compiz类似的组合窗口管理器，但是它要更简单一些。它并不是将显示内容直接绘制到屏幕

① 参见网页<http://androidappdocs.appspot.com/guide/developing/tools/adb.html>。

缓冲区中,而是将绘制命令传递给屏幕外的位图,然后将该位图与其他位图组合起来,形成用户看到的显示内容。这种方法允许系统实现所有有趣的效果,如透明的窗口和奇特的过渡效果。

- **2D和3D图形。**使用Android时,二维和三维元素可结合到一个用户界面中。库将使用3D硬件(如果设备上有的话)或者快速软件渲染器(如果没有3D硬件设备)。参见第4章和第10章。
- **媒体编解码器。**Android可播放视频内容,并可用各种格式录制和播放音频,这些格式包括AAC、AVC(H.264)、H.263、MP3和MPEG-4。参见第5章中相关的示例。
- **SQL数据库。**Android提供了轻量级的SQLite数据库引擎^①,Firefox和苹果的iPhone中使用的也是该数据库引擎。使用该引擎可在应用程序中持续存储。参见第9章中相关的示例。
- **浏览器引擎。**为保证快速显示HTML内容,Android使用了WebKit库^②。Google Chrome浏览器、苹果的Safari浏览器、苹果iPhone和诺基亚的S60平台都使用了该引擎。参见第7章中相关的示例。

2.1.3 Android 运行时

在Linux内核层上面还有一个Android运行时层,该层包括Dalvik虚拟机以及Java核心库。

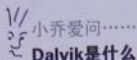
Dalvik虚拟机是Google的Java实现,专门针对移动设备进行了优化。为Android编写的所有代码使用的都是Java语言,这些代码都在虚拟机中运行。

Dalvik与传统Java虚拟机的不同之处体现在下面两个重要的方面。

- **Dalvik VM运行.dex文件,**即编译时会将标准的.class和.jar文件转换为.dex文件。.dex文件比类文件更加紧凑并且更加高效,这是针对运行Android的设备内存有限且通过电池供电的特点所作出的重要改进。
- **Android附带的Java核心库与Java SE库和Java ME(Java Mobile Edition, Java移动版)库不同。**但是,它们之间有很大一部分还是相同的。附录A比较了Android库与标准Java库。

① 参见网页<http://www.sqlite.org>。

② 参见网页<http://www.webkit.org>。



小乔爱问……

Dalvik是什么

Dalvik是由Google公司的Dan Bornstein设计并编写的一款虚拟机（VM）。你编写的代码首先编译为与机器无关的指令，称为字节码，然后由移动设备上的Dalvik VM执行这些字节码。

虽然不同的字节码格式稍有不同，但Dalvik本质上是一个针对低内存耗用而优化的Java虚拟机。它允许同时运行多个VM实例，并且能够充分利用底层操作系统（Linux）实现安全性和进程隔离。

Dalvik是Bornstein祖先居住在冰岛的一个渔村名。

2.1.4 应用程序框架

位于本机类库和运行时上面的是应用程序框架层。该层提供了在创建应用程序时需要使用的各种高级构建块。该框架已随Android一同安装，但开发人员也可以根据需要使用自己的组件扩展该框架。

该框架最重要的部分包括下面5个。

- ❑ **活动管理器。**该管理器控制应用程序的生命周期（参见2.2节），同时维护一个公共的“后退栈”（backstack）供用户导航。
- ❑ **内容提供者。**这些对象封装需要在应用程序之间共享的数据，如联系人信息。参见2.3节。
- ❑ **资源管理器。**资源是程序中涉及的任何非代码内容。参见2.4节。
- ❑ **位置管理器。**Android手机始终知道目前所处的位置。参见第8章。
- ❑ **通知管理器。**像收到短信、临近预约时间、临界状态报警、异常入侵等事件都可以通过友好的方式通知用户。

包容与扩展

Android独特且强大的特性之一，就是所有应用程序一律平等。也就是说，系统级应用程序使用的公共API，与你的应用程序使用的API完全相同。如果需要，甚至还可以让Android用你的应用程序替换标准的应用程序。

2.1.5 应用程序

Android架构图中的最高层是应用程序层。可将该层想象为浮出海面的Android冰山的一角。最终用户只能看到这些应用程序，根本不会察觉到在该层下面执行的操作。但是作为一名Android开发人员，你应该知道这些操作。

购置Android手机时，手机中会预装一些标准的系统应用程序，包括：

- 电话拨号程序；
- 电子邮件收发程序；
- 联系人管理程序；
- Web浏览器；
- Android Market。

用户可以从Android Market下载在本机运行的新应用程序。这里也将是你大显身手的地方，学习完本书后，你也能够为Android编写出令人一见钟情的应用程序。

现在，让我们仔细地了解一下Android应用程序的生命周期，它与你以前所熟悉的概念有所不同。

2.2 它还活着

标准的Linux或Windows桌面可以同时运行许多应用程序，并且可在不同的窗口中同时看到这些程序。除了其中某个窗口拥有键盘焦点外，所有程序都是平等的。用户可以轻松地在这些程序（窗口）之间切换，但是要想知道自己在做什么，或者想要关闭不再需要的程序，用户必须亲自动手。

Android不是这样管理窗口的。

Android中有一个前台应用程序，它通常会占据除状态栏以外的所有屏幕空间。用户开机时，他们看到的第一个应用程序是Home应用程序（参见图2-2）。这个程序通常会显示时间、背景图像，以及一个滚动列表，其中包含用户可以打开的其他应用程序。

用户在运行应用程序时，Android会启动该程序并将其置于前台。用户可从该

应用程序中打开其他应用程序，或者打开同一应用程序的其他窗口，然后再调用其他应用程序或打开其他窗口。所有这些程序和窗口都被系统的活动管理器记录在活动程序栈(application stack)中。用户可随时按Back按钮返回到栈中的上一个窗口。从用户的角度看，这种工作方式类似于Web浏览器中的历史功能，即按Back返回到上一个页面。



图2-2 Home应用程序

2.2.1 进程不等于应用程序

从内部来看，每个用户界面窗口都是通过一个Activity类(参见2.3节)表示的，而每个活动(activity)都有其自己的生命周期。一个应用程序就是一个或多个活动加上包含这些活动的Linux进程。这听起来很容易懂，是吧？但是先不要松懈，下面我就告诉你一个“骇人听闻”的事实。

在Android中，即使所在进程被“杀死”(结束)，相应的应用程序仍然还是“活着的”。换句话说，活动的生命周期与进程的生命周期没有关系。进程只是各种活动可随意使用的一个容器。这个事实大概与读者熟悉的其他系统中的情况不大一样，因此在继续讲解其他内容之前，有必要先搞清楚这是怎么回事。

2.2.2 应用程序生命周期

Android程序中的每个活动在其存在期间都会处于以下多种状态之一，如图2-3所示。开发人员不能控制程序处于哪个状态，这是由系统管理的。但是，通过onXX()方法调用改变状态时，系统会通知开发人员。

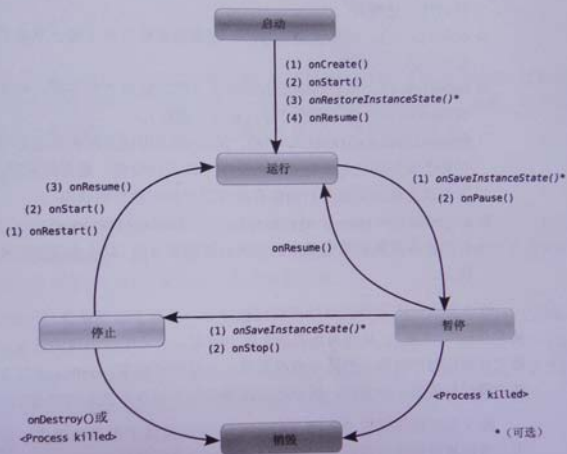


图2-3 Android活动的生命周期

你需要在Activity类中重写这些方法，而Android会在合适的时间调用下面这些方法。

- onCreate(Bundle)。首次启动活动时调用该方法。可使用该方法执行一次性的初始化工作，如创建用户界面。onCreate()接受一个参数，可以是null或由onSaveInstanceState()方法以前保存的某些状态信息。
- onStart()。该方法说明了将要显示给用户的活动。
- onResume()。用户可以开始与活动进行交互时会调用该方法。这个方法非常适合开始播放动画和音乐。

- ❑ **onPause()**。活动将要进入后台时会运行该方法，活动进入后台的原因通常是在前台启动了另一个活动。还应该在该方法中保存程序的持久性状态，如正在编辑的数据库记录。
- ❑ **onStop()**。用户无需看到某个活动，或者在一段时间内不需要某个活动时，可以调用该方法。如果内存不足，可能永远都不会调用**onStop()**（系统可能只是终止进程）。
- ❑ **onRestart()**。如果调用该方法，则表明要将已处于停止状态的活动重新显示给用户。
- ❑ **onDestroy()**。销毁活动前会调用该方法。如果内存不足，可能永远都不会调用**onDestroy()**（系统可能只是终止进程）。
- ❑ **onSaveInstanceState(Bundle)**。Android调用该方法的作用是让活动可以保存每个实例的状态，如光标在文本字段中的位置。通常你无需重写该方法，因为该方法的实现会自动保存所有用户界面控件的状态^①。
- ❑ **onRestoreInstanceState(Bundle)**。使用**onSaveInstanceState()**方法以前保存的状态重新初始化某个活动时调用该方法。默认实现会还原用户界面的状态。

没有在前台中运行的活动可能已被停止，或者是容纳这些活动的Linux进程已被“杀死”（结束），从而为新的活动腾出空间。这是经常出现的情况，所以在一开始设计应用程序时就记住这一点很重要。在某些情况下，**onPause()**方法可能是活动中调用的最后一个方法，所以才应在该方法中保存下次要继续使用的任何数据。

除了管理程序的生命周期，Android框架还提供了很多构建块，开发人员可使用这些构建块创建应用程序。下面仔细了解这些构建块。

翻盖动作

此处提供了一种可确定状态保存代码是否能正常工作的快速方法。在当前版本的Android中，一次方向变化（在纵向和横向模式之间切换）会让系统经历以下过程：保存实例的状态、暂停、停止、销毁，然后使用已保存的状态创建新的活动实例。例如，在T-Mobile G1手机上，翻动键盘上的手机盖就会触发上述过程。在Android模拟器上，按Ctrl+F11或小键盘上的7或9键也会触发上述过程。

^① 在0.9_beta以前的版本中，**onSaveInstanceState()**被称为**onFreeze()**，被保存的状态称为**icicle**。在某些文档和示例中仍会看到这些过去使用的名称。

2.3 构建块

每位开发人员都需要熟悉Android SDK中定义的一些对象。其中最为重要的就是活动、意图（intent）、服务和内容提供者。本书的其余部分提供了关于这些对象的多个示例，所以现在先简要介绍一下它们。

2.3.1 活动

一个活动就是一个用户界面屏幕。应用程序可定义一个或多个活动，以处理程序不同阶段中的任务。如2.2节所述，作为应用程序生命周期的一部分，每个活动都要保存自己的状态，以便日后还原这些状态。3.3节提供了一个示例。

2.3.2 意图

意图是一种描述具体动作的机制，例如“拍照”、“往家中拨电话”或“打开仓门”。在Android中，几乎所有事情都要经历意图这个阶段，所以有很多机会可替换或重用很多组件。3.5节提供了一个意图的示例。

例如，现在有一个“发送一封电子邮件”的意图。如果应用程序需要发送邮件，就可调用该意图。或者你正在编写一个新的电子邮件应用程序，就可注册一个活动来处理该意图，并用该电子邮件程序替换标准的邮件程序。下次其他人尝试发送电子邮件时，他们会使用你的电子邮件程序，而不是标准的邮件程序。

2.3.3 服务

服务是在后台运行的任务，无需用户直接与其交互，它与Unix的守护进程类似。例如，假设有一个音乐播放器。可以通过某个活动来播放音乐，但是我们希望即使当用户使用其他程序时，仍能继续播放音乐。所以，执行音乐播放的代码应该在某个服务中。之后，另一个活动可能会绑定到该服务上，告诉该服务切换音轨或停止播放。Android内置了许多服务，以及许多可轻松访问这些服务的API。

2.3.4 内容提供者

内容提供者是封装在自定义API中的一组数据，可以读取该数据和向API中写入数据。这是在应用程序之间共享全局数据的最佳方式。例如，谷歌为联系人程序提

供了一个内容提供者。任何要使用联系人信息的应用程序都可共享其中的所有信息，包括姓名、地址、电话号码等。9.5节提供了一个示例。

2.4 使用资源

资源是程序需要的、本地化后的文本字符串、位图或其他少量的非代码信息。程序生成时，所需的所有资源都会被编译到应用程序中。

资源由开发人员创建并存储在项目内的 `res` 目录中。Android 资源编译器 (aapt)^① 会根据资源所处的子文件夹以及文件的格式来处理这些资源。例如，PNG 和 JPG 格式的位图应该放入 `res/drawable` 目录，而描述屏幕布局的 XML 文件应该放入 `res/layout` 目录。

资源编译器会将资源压缩并打包，然后生成一个名为 `R` 的类，其中包含的标识符可用于在程序中引用这些资源。这与标准的 Java 资源稍有不同，因为我们要通过键字符串来引用 Java 资源。由于使用了这种方式，所以 Android 可以确保所有引用都是有效的，并且还可节省空间，因为它不必存储所有这些资源键。Eclipse 使用类似的方法来存储并引用 Eclipse 插件中的各种资源。

第3章将会介绍一个资源访问代码示例。

2.5 安全性

如前所述，每个应用程序都在其自己的 Linux 进程中运行。硬件禁止一个进程访问其他进程的内存。另外，每个应用程序都被分配了一个具体的用户 ID。某个应用程序所创建的任何文件都不能被其他应用程序读写。

另外，对某些关键操作的访问也是有限制的，必须在一个名为 `AndroidManifest.xml` 的文件中明确请求获得所需的权限。安装应用程序时，软件包管理器会根据证书以及（如有必要）用户提示的情况授予或不授予这些权限。下面是一些最常用的权限。

□ INTERNET：访问因特网。

① 参见网页 <http://androidappdocs.appspot.com/guide/developing> 更多资源请访问我的新浪博客 <http://blog.sina.com.cn/ckook>

- READ_CONTACTS: 读取（但不能写入）用户的联系人数据。
- WRITE_CONTACTS: 写入（但不能读取）用户的联系人数据。
- RECEIVE_SMS: 监视收到的SMS（文本）短信。
- ACCESS_COARSE_LOCATION: 使用不太精确的位置提供者，如手机基站或wifi。
- ACCESS_FINE_LOCATION: 使用更为精确的位置提供者，如GPS。

例如，要想监视收到的SMS短信，应该在manifest文件中指定以下内容：

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.google.android.app.myapplication" >
    <uses-permission android:name="android.permission.RECEIVE_SMS" />
</manifest>
```

Android甚至可以限制对整个系统各部分的访问。通过在AndroidManifest.xml中使用XML标记，可以限制谁能启动活动、启动活动并将其绑定到服务、向接收者广播意图，或者访问内容提供者中的数据。有关此类控制的内容超出了本书的范围，但是如果你想深入了解该内容，可阅读Android安全性模型的在线帮助内容^①。

2.6 快速阅读指南

本书其余部分将使用本章中介绍的所有概念，在第3章中，我们将使用活动和生命周期方法定义一个应用程序示例。第4章将使用Android本机库中的一些图形类。第5章会探讨媒体编解码器，第9章将介绍内容提供者。

^① 参见网页<http://androidappdocs.appspot.com/guide/topics/security/security.html>。

Part 2

第二部分

Android 基础知识

本 部 分 内 容

- 第3章 设计用户界面
- 第4章 绘制2D图形
- 第5章 多媒体
- 第6章 存储本地数据

第1章介绍了如何使用Android Eclipse插件快速创建一个简单的“Hello, Android”程序。这部分将创建一个更加真实的示例程序：数独游戏。通过逐步完善该游戏的功能，读者可以掌握Android程序设计的很多方面。首先介绍用户界面设计。

在网站<http://pragprog.com/titles/eband>上提供了本书使用的所有示例代码。

3.1 数独游戏简介

数独游戏之所以成为出色的Android示例程序，是因为该游戏本身非常简单。数独盘面由81个单元格（9行×9列）组成，玩家要试着在这些单元格中填入1~9之间的数字，使每个数字在每一行、每一列和每一区（3行×3列的部分）中都只出现一次。游戏开始时，部分单元格中已经填入了一些（已知）数字。玩家只需在剩下的空单元格中填入数字。一道正确的数独谜题只有唯一的答案。

人们通常用纸和笔玩数独游戏，但是用计算机玩数独游戏也非常流行。如果用纸和笔玩数独游戏，由于在游戏刚开始时容易出错，所以玩家在出现错误时，必须回退一步或几步并擦除已经填入的大部分数字。如果玩Android数独游戏，就可以随时修改单元格中的数字，而不必再清理令人讨厌的橡皮屑。

数独轶事

大多数人认为数独是一种古老的日本游戏，其实不然。虽然类似的谜题游戏可以追溯到19世纪的法国杂志，但是大部分专家认为现代数独游戏是一位已退休

的美国建筑师Howard Garns发明的。1979年，Dell杂志在美国首次刊载了此类数字谜题，当时称为“数字填空”（Number Place）。

Android数独游戏（参见图3-1）也会给出一些提示，以减少破解谜题的工作量。一种极端的情况是，Android程序正好给出该谜题的答案，但这样对玩家而言就没有任何乐趣了。因此，我们必须在提示和挑战之间进行折中，让谜题的难度适中。

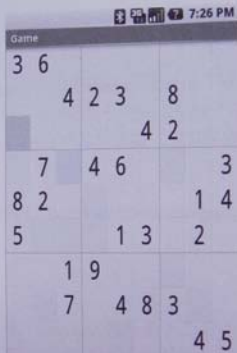


图3-1 Android数独示例程序

3.2 声明性设计

设计用户界面的方法有两种：过程性设计和声明性设计。过程性设计是指用代码设计用户界面。例如，开发一个Swing应用程序时，你必须编写Java代码，以创建和操作所有用户界面对象（例如JFrame和JButton）。因此，Swing是过程性的。

另一方面，声明性设计不涉及任何代码。例如，在设计简单的网页时会使用HTML，HTML是一种基于XML的标记语言，描述了你期望的网页布局，而不是如何实现该布局。因此，HTML是声明性的。

Android同时支持过程性设计和声明性设计，允许开发人员使用任一风格创建

用户界面：既可以几乎全部使用Java代码，也可以几乎全部使用XML描述符。如果查阅任何Android用户界面组件的相关文档，就会看到其中既有Java API，又有实现同一功能的、相应的声明性XML属性。

究竟该使用哪种方法呢？虽然二者都是合法的，但是我的建议是尽可能使用声明性的XML语言。与相应的Java代码相比，XML代码往往更加短小易懂，而且以后可能开发的Android工具（如GUI设计器）会更好地与XML协作。

下面就来介绍如何使用该方法创建数独游戏的启动界面。

3.3 创建启动界面

首先使用Eclipse插件创建一个最简单的Android程序，其方法与在1.2节中创建“Hello, Android”新项目时一样，但是这次将使用以下信息：

```
Project name: Sudoku
Package name: org.example.sudoku
Activity name: Sudoku
Application name: Sudoku
```

当然，在实际程序中，开发人员可以定义自己的名称。其中，软件包的名称尤为重要。系统中的每个应用程序都必须具有唯一的软件包名。因为在很多地方都会用到软件包名，所以一旦设定好软件包名后就不要随意修改它，否则将会引起一些小麻烦。

由于Android模拟器的执行速度很快，因此我喜欢让模拟器窗口一直处于打开状态，并在每次修改程序后立即运行程序。如果现在你已经按照前面的要求创建了Sudoku项目并运行Sudoku程序，就会看到一个空白的界面，其中只显示“Hello World, Sudoku.”这样几个单词。完成Sudoku程序的第一项工作就是修改游戏的启动界面——上面要有允许玩家开始新游戏、继续前一个游戏、获得游戏相关信息以及退出游戏的按钮。那么，必须修改什么才能创建这样的启动界面呢？

如第2章所述，Android程序是一个松散的活动集合，其中的每个活动都定义了一个用户界面屏幕。创建Sudoku项目时，Android插件会在Sudoku.java文件中建立一个活动。

```
Sudoku.java:src/org/example/sudoku/Sudoku.java
package org.example.sudoku;
```

```
import android.app.Activity;
import android.os.Bundle;

public class Sudoku extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Android调用活动的onCreate()方法来初始化该活动，onCreate()方法又调用setContentView()方法，利用一个Android视图部件填充该活动的屏幕区域内容。

当然，也可以按照过程性设计方法使用几行Java代码，以及其他一个或几个类来定义用户界面。但Android插件采用的是声明性设计方法，所以本书将继续沿用这一方法。上面代码中的R.layout.main是一个资源标识符，引用了res/layout目录中的main.xml文件（参见图3-2）。main.xml文件以XML格式声明用户界面，所以这正是我们需要修改的文件。在运行时，Android解析并实例化（解压缩）该文件中已定义的资源，并将其设置为当前活动的视图。

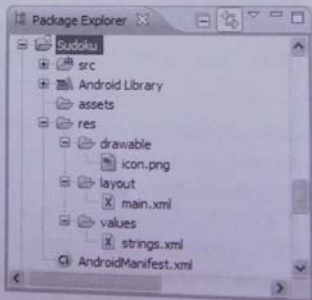


图3-2 Sudoku项目中的初始资源

注意，R类是由Android Eclipse插件自动管理的，这点非常重要。无论将一个文件放到res目录中的哪个位置，Android Eclipse插件都会注意到这一变化并自动在R.java文件中添加资源ID。如果删除或修改了某个资源文件，R.java将保持同步。如果在编辑器中打开该文件，就会看到与下面类似的内容：

```

budokun0/src/org/example/sudoku/R.java
/* AUTO-GENERATED FILE. DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found. It
 * should not be modified by hand.
 */

package org.example.sudoku;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}

```



小乔爱问……

XML的效率不是不高吗，为什么Android还要使用XML

Android是针对内存和功率都有限的移动设备而优化的，因此读者可能会对其如此广泛地使用XML感到奇怪。毕竟，XML是一种详细的可读格式，但在简洁或高效方面做得就不太好了，对不对？

虽然开发人员在编写程序时使用的是XML，但是Eclipse插件会调用Android资源编译器aapt，将XML文件预处理为压缩的二进制格式。移动设备中存储的正是这种压缩的二进制格式文件，而不是原始的XML文件。

Android资源管理器使用十六进制整数加载实际数据、字符串以及被编译到软件包中的其他资源。无需关心这些资源的值。只要记住这些值是引用数据的句柄，而不是引用包含数据的对象的句柄。需要使用对象之前不会解压缩对象。注意，几乎所有的Android程序，包括基础Android框架本身，都拥有一个R类。请参阅android.R的在线文档，了解可用的所有内置资源^①。

现在，我们知道了必须要修改main.xml文件。下面就剖析一下原始main.xml文件中的定义，了解必须要修改哪些地方。在Eclipse中双击main.xml即可打开该文件，根据Eclipse的具体设置，双击main.xml后出现的可能是一个可视化的布局编辑器，或者是一个XML编辑器。在当前的ADT版本中，可视化的布局编辑器用处不大，因此请单击main.xml或底部的Source选项卡，查看XML文件。

main.xml文件的第一行如下所示。

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

所有的Android XML文件开头都是这行代码。这行代码只是告诉编译器，该文件是XML格式的，采用UTF-8字符集编码。除了包含针对非ASCII字符（例如日文字符）的转义码以外，UTF-8字符集和常规的ASCII字符集几乎完全一样。

接下来会看到对<LinearLayout>的引用：

```
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent">
  <!-- ... -->
</LinearLayout>
```

布局是存放一个或多个子对象及某种行为的容器，行为描述了如何将这些子对象放置在屏幕上其父对象的矩形区域中。下面列出了Android中最常用的几种布局。

- **FrameLayout**（框架布局）：从屏幕的左上角开始显示子对象，主要用于选项卡视图和图像切换器。
- **LinearLayout**（线性布局）：以单列或单行的形式显示子对象，这是最常用的布局方式。
- **RelativeLayout**（相对布局）：以相对于其他子对象或父对象的位置显示子对象，这种布局通常用于表单中。
- **TableLayout**（表格布局）：以多行和多列的方式显示子对象，类似于HTML表格。

各种布局方式常用的一些参数如下：


```
xmlns:android="http://schemas.android.com/apk/res/android"
```

定义Android的XML命名空间，该参数只能在main.xml文件中的第一个XML标记处定义一次。

```
android:layout_width="fill_parent", android:layout_height="fill_parent"
```

声明此布局的高度和宽度与父对象的高度和宽度相同（此处指窗口）。可能的值有fill_parent或wrap_content。

<LinearLayout>标记中有一个如下所示的子对象部件：

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello" />
```

此处定义了一个简单的文本标签。我们使用不同文本和几个按钮来替换该标签，下面是第一次尝试：

Sudoku\src\res\layout\main1.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">

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/main_title" />

    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/continue_label" />

    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/new_game_label" />

    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/about_label" />

    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/exit_label" />

</LinearLayout>
```

如果在编辑器中看到关于缺少语法约束（DTD或XML架构）的警告，尽管忽略

这些警告。

与将英语文本硬编码到布局文件中不同，我们使用@string/resid这种语法形式引用res/values/strings.xml文件中的字符串。该文件也可以有其他写法，根据区域设置或其他参数（如屏幕分辨率和屏幕方向）的设置情况，还可以有其他资源文件。现在打开这个文件，如有必要，切换到底部的strings.xml选项卡，然后输入以下内容：

```
Sudoku/res/values/strings.xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Sudoku</string>
    <string name="main_title">Android Sudoku</string>
    <string name="continue_label">Continue</string>
    <string name="new_game_label">New Game</string>
    <string name="about_label">About</string>
    <string name="exit_label">Exit</string>
</resources>
```

现在运行Sudoku程序，应该看到如图3-3所示的结果。虽然该启动界面清晰易读，但还需要进一步美化。

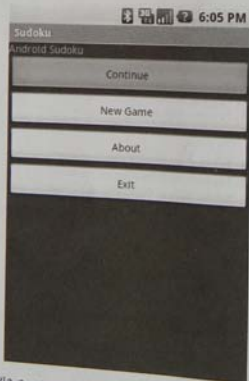


图3-3 Sudoku程序启动界面的第一个版本

接下来让标题文本更大一些并使其居中，让按钮变小并使用不同的背景色。

面是颜色定义，这些定义应放在res/values/colors.xml文件中：

SudokuV1/res/values/colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="background">#3500ffff</color>
</resources>
```

新的布局如下：

SudokuV1/res/layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="@color/background"
    android:layout_height="fill_parent"
    android:layout_width="fill_parent"
    android:padding="30dip"
    android:orientation="horizontal">
    <LinearLayout
        android:orientation="vertical"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:layout_gravity="center">
        <TextView
            android:text="@string/main_title"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:layout_gravity="center"
            android:layout_marginBottom="25dip"
            android:textSize="24.5sp" />
        <Button
            android:id="@+id/continue_button"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="@string/continue_label" />
        <Button
            android:id="@+id/new_button"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="@string/new_game_label" />
        <Button
            android:id="@+id/about_button"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="@string/about_label" />
        <Button
            android:id="@+id/exit_button"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="@string/exit_label" />
    </LinearLayout>
</LinearLayout>
```

在这个版本的main.xml文件中，我们引入了一种新的语法形式：`@+id/resid`。与引用在其他地方定义的资源ID不同，这种语法形式将创建一个新的资源ID供其他对象引用。例如，`@+id/about_button`定义了About按钮的ID，以后用户按下该按钮时，我们将使用这个ID触发某个事件。

重新运行该程序的结果如图3-4所示。新的界面在纵向模式（屏幕的高度大于其宽度）时非常美观，但在横向模式（宽屏幕）时会是什么情况？用户随时都可能切换模式，例如翻转键盘或侧放手机，因此必须考虑对横向模式的处理。



图3-4 使用新布局后的启动界面

3.4 使用替代资源

作为一种测试，现在将模拟器切换到横向模式（按Ctrl+F11或小键盘上的7或9键）。糟了，屏幕底部的Exit按钮消失了（参见图3-5）。如何解决这个问题？

可以尝试调整布局，使其在各种方向模式下都能正常显示内容。但遗憾的是，这样做往往并不能奏效，或者会让界面看起来很奇怪。因此，需要为横向模式创建一个不同的布局，这正是本章将采用的方法。



图3-5 在横向模式下，我们看不到Exit按钮

3

1// 小乔爱问……

什么是Dip和Sp

过去，程序员通常以像素为单位设计计算机用户界面。例如，定义一个宽度为300像素的表单字段，列之间的间距为5个像素，图标大小为 16×16 像素等。这样处理的问题在于，如果在一个每英寸点数（dpi）更高的新显示器上运行该程序，则用户界面会显得很小。在有些情况下，用户界面可能会小到难以看清内容。

与分辨率无关的度量单位可以解决这一问题。Android支持下列所有单位。

- px（像素）：屏幕上的点。
- in（英寸）：长度单位。
- mm（毫米）：长度单位。
- pt（磅）： $1/72$ 英寸。
- dp（与密度无关的像素）：一种基于屏幕密度的抽象单位。在每英寸160点的显示器上， $1dp = 1px$ 。
- dip：与dp相同，多用于Google示例中。
- sp（与刻度无关的像素）：与dp类似，但是可以根据用户的字体大小首选项进行缩放。

为了使用户界面能够在现在和将来的显示器类型上正常显示，我建议你始终使用sp作为文字大小的单位，将dip作为其他元素的单位。当然，也可以考虑使用矢量图形，而不是用位图（参见第4章）。

创建一个名为res/layout-land/main.xml（注意-land后缀）的文件，该文件包含下列布局：

Sudoku\src\res\layout-land\main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="@color/background"
    android:layout_height="fill_parent"
    android:layout_width="fill_parent"
    android:padding="15dip"
    android:orientation="horizontal">
    <LinearLayout
        android:orientation="vertical"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:layout_gravity="center"
        android:paddingLeft="20dip"
        android:paddingRight="20dip">
        <TextView
            android:text="@string/main_title"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:layout_gravity="center"
            android:layout_marginBottom="20dip"
            android:textSize="24.5sp" />
        <TableLayout
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:layout_gravity="center"
            android:stretchColumns="*">
            <TableRow>
                <Button
                    android:id="@+id/continue_button"
                    android:text="@string/continue_label" />
                <Button
                    android:id="@+id/new_button"
                    android:text="@string/new_game_label" />
            </TableRow>
            <TableRow>
                <Button
                    android:id="@+id/about_button"
                    android:text="@string/about_label" />
                <Button
                    android:id="@+id/exit_button"
                    android:text="@string/exit_label" />
            </TableRow>
        </TableLayout>
    </LinearLayout>
</LinearLayout>
```

该文件采用TableLayout布局方式创建两列按钮。现在再次运行该程序（结果如图3-6所示）。即使在横向模式下，所有按钮也都可以正常显示。

Android允许使用资源后缀为任何资源指定替代版本，而不仅仅是布局。例如

可以使用替代资源提供不同语种的本地化文本字符串。每个替代资源文件必须定义完全相同的ID集合。



图3-6 使用特定的横向布局，所有按钮都可以正常显示

Android支持为当前语言、区域、像素密度、分辨率、输入法等资源添加后缀。请参见Android资源文档，了解最新的后缀和继承规则列表^①。

3.5 实现 About 对话框

用户选择了About按钮时，表明他们或者触按了屏幕上的该按钮（如果用户使用触摸屏），或者使用D-pad（directional pad，方向键）或轨迹球指向该按钮并按下了选择按钮。此时，我们希望弹出一个窗口，其中显示一些有关该数独游戏的信息。

阅读完该对话框上的这些文字后，用户按下Back按钮可关闭该窗口。

可以用多种方法实现这个功能：

- ❑ 定义一个新的Activity类并启动它；
- ❑ 使用AlertDialog类并显示它；
- ❑ 子类化Android的Dialog类并显示它。

在这个例子中，我们将定义一个新的活动。与主Sudoku程序的活动一样，About按钮的活动也需要一个布局文件。我们将该布局文件命名为res/layout/about.xml：

① 参见网页<http://androidappdocs.appspot.com/guide/topics/resources/index.html>。

Sudoku\res\layout\about.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="10dip">
    <TextView
        android:id="@+id/about_content"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/about_text" />
</ScrollView>
```

我们只需要该布局文件的一个版本即可，因为该布局在纵向和横向模式下都能够正常显示。

下面将About对话框的标题及对话框中包含的文本内容添加到res/values/strings.xml文件：

Sudoku\res\values\strings.xml

```
<string name="about_title">About Android Sudoku</string>
<string name="about_text">\
Sudoku is a logic-based number placement puzzle.
Starting with a partially completed 9x9 grid, the
objective is to fill the grid so that each
row, each column, and each of the 3x3 boxes
(also called <i>blocks</i>) contains the digits
1 to 9 exactly once.
</string>
```

注意字符串资源是如何包含简单的HTML格式以及如何分布在多行中的。不知你有没有注意到，about_text中的反斜杠字符（\）避免了在第一个单词的前面出现额外的空行。

About活动应该在About.java文件中定义，只需重写onCreate()方法并调用setContentView()方法即可：

Sudoku\src\org\example\sudoku\About.java

```
package org.example.sudoku;

import android.app.Activity;
import android.os.Bundle;

public class About extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```



```

        setContentView(R.layout.about);
    }
}

```

接下来需要将该活动与Sudoku类中的About按钮关联起来。首先在Sudoku.java中添加一些导入语句：

```

Sudoku\src\org\example\sudoku\Sudoku.java
import android.content.Intent;
import android.view.View;
import android.view.View.OnClickListener;

```

在onCreate()方法中添加代码以调用findViewById()和setOnClickListener()方法，findViewById()方法用于根据资源ID查找Android视图，setOnClickListener()方法则通知Android当用户触摸或单击该视图时应触发哪个对象：

```

Sudoku\src\org\example\sudoku\Sudoku.java
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    // Set up click listeners for all the buttons
    View continueButton = this.findViewById(R.id.continue_button);
    continueButton.setOnClickListener(this);
    View newButton = this.findViewById(R.id.new_button);
    newButton.setOnClickListener(this);
    View aboutButton = this.findViewById(R.id.about_button);
    aboutButton.setOnClickListener(this);
    View exitButton = this.findViewById(R.id.exit_button);
    exitButton.setOnClickListener(this);
}

```

同时，在此需要对所有按钮进行相同的处理。还记得吗，当Eclipse插件在res/layout/main.xml文件中看到@+id/about_button这样的语法形式时，它将自动在R.java文件中创建类似R.id.about_button这样的常量。

下面的代码使用this作为接收方，因此Sudoku类需要实现OnClickListener接口并定义一个名为onClick()的方法^①：

```

Sudoku\src\org\example\sudoku\Sudoku.java
public class Sudoku extends Activity implements OnClickListener {
    // ...
}

```

① 我们可以使用一个匿名的内部类来处理单击事件，但是根据Android开发人员的经验，每个新的内部类将多占用1KB的内存。

```

public void onClick(View v) {
    switch (v.getId()) {
        case R.id.about_button:
            Intent i = new Intent(this, About.class);
            startActivity(i);
            break;
        // More buttons go here (if any) ...
    }
}
}

```

要想在Android中启动某个活动，首先需要创建Intent类的实例。Android中有两种Intent类实例：**public**（命名的）的Intent类实例是在系统注册的，可从任何应用程序中调用该实例；**private**（匿名的）的Intent类实例只能在一个应用程序内使用。在这个例子中，只需使用后一种实例。

如果现在运行该程序并选择About按钮，将会出现错误提示（参见图3-7）。这是怎么回事呢？



图3-7 Android中的错误提示

这是因为我们漏掉了一个重要的步骤：每个活动都需要在AndroidManifest.xml文件中声明。为此可双击AndroidManifest.xml打开该文件，如有必要，可以选择底部的AndroidManifest.xml选项卡切换到XML模式，并在第一个活动的结束标记之后添加一个新的<activity>标记：

```

Sudokuv1/AndroidManifest.first.xml
<activity android:name=".About"
    android:label="@string/about_title">
</activity>

```

现在，如果保存AndroidManifest.first.xml文件，再次运行该程序并选择

About按钮，将会看到如图3-8所示的结果。完成后按Back按钮（或者模拟器上的Esc键）。



图3-8 第一个版本的About对话框

About对话框看起来相当不错，但如果在关闭该对话框之后能够返回到启动界面，岂不是更好？

3.6 应用主题

主题是一个样式集合，可用于重写Android部件的外观。Android中的主题是受网页设计中使用的CSS（Cascading Style Sheets，层叠样式表）的启发，层叠样式表将屏幕上显示的内容与其外观或样式分离开来。Android附带了多个主题，可以通过名称进行引用^①，另外，通过子类化现有主题并重写其默认值，用户也可以构造自己的主题。

^① 参见<http://androidappdocs.appspot.com/reference/android/R.style.html>，了解其中以Theme_开头的內容。

可以在res/values/styles.xml文件中定义自定义主题,但是此处我们只需利用一个预定义的主题。要使用主题,首先在编辑器中再次打开AndroidManifest.xml文件,然后修改About活动的定义,以使其具有主题特性(property)。

Sudoku\1\AndroidManifest.xml

```
<activity android:name=".About"
    android:label="@string/about_title"
    android:theme="@android:style/Theme.Dialog">
</activity>
```

样式名称前的@android:前缀表示此处引用了由Android定义的资源,而不是用户自己在程序中定义的资源。

再次运行该程序,现在About对话框应该如图3-9所示。

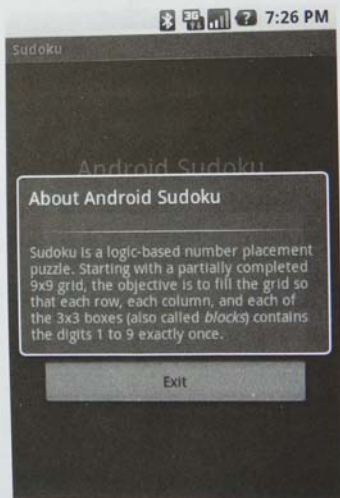


图3-9 应用对话框主题后的About界面

很多程序都需要菜单和选项,因此接下来的两节内容将介绍如何定义菜单和选项。



小乔爱问……

为什么不使用HTML视图

利用WebView类（参见7.2节），Android支持直接在视图中嵌入Web浏览器。那么，为什么不使用HTML视图来实现About对话框？

实际上，无论是用本章提供的方法还是用HTML视图，都可以实现这一功能。与简单的TextView类相比，WebView类能够支持更复杂的格式，但是WebView类也有一些局限性（例如不能使用透明背景）。另外，WebView类是一个重量级部件，其执行速度比TextView类要慢，而且占用的内存也比TextView类多。对于你自己的应用程序，可以选择最适合自己需要的实现方式。

3

3.7 添加菜单

Android支持两类菜单：一种是在用户按下Menu按钮时弹出的菜单；另一种是用户用手指按住屏幕不放（或按住D-pad的中间按钮不放）时弹出的上下文菜单。

下面来实现第一种菜单，用户按下Menu按钮时，将打开一个如图3-10所示的菜单。首先，需要定义几个随后要用到的字符串：

Sudoku\res\values\strings.xml

```
<string name="settings_label">Settings...</string>
<string name="settings_title">Sudoku settings</string>
<string name="settings_shortcut">s</string>
<string name="music_title">Music</string>
<string name="music_summary">Play background music</string>
<string name="hints_title">Hints</string>
<string name="hints_summary">Show hints during play</string>
```



图3-10 按下Menu按钮时打开的菜单

其次，使用XML在res/menu/menu.xml文件中定义菜单：

Sudoku\res\menu\menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/settings"
        android:title="@string/settings_label"
        android:alphabeticShortcut="@string/settings_shortcut" />
</menu>
```

接下来需要修改Sudoku类，将刚才定义的菜单加入到Sudoku类中。为此，需要再导入一些包：

Sudoku\src\org\example\sudoku\Sudoku.java

```
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
```

最后重写Sudoku类中的onCreateOptionsMenu()方法：

Sudoku\src\org\example\sudoku\Sudoku.java

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu);
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu, menu);
    return true;
}
```

getMenuInflater()方法返回一个MenuInflater类的实例，用于从XML文件中读取菜单定义，并将其转换为实际的视图。

用户选择任何菜单项时，将会调用onOptionsItemSelected()方法，该方法的定义如下：

Sudoku\src\org\example\sudoku\Sudoku.java

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.settings:
            startActivity(new Intent(this, Settings.class));
            return true;
        // More items go here (if any) ...
    }
    return false;
}
```

Settings是一个类，其定义在本章的稍后部分给出，该类可以显示所有的用户首选项并允许用户修改这些首选项的值。

3.8 添加设置

Android提供了方便的工具来定义所有的程序首选项，并支持在几乎不需要编写代码的情况下显示这些首选项。可以在一个名为res/xml/settings.xml的资源文件中定义这些首选项：

```
Sudoku\res\xml\settings.xml
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen
    xmlns:android="http://schemas.android.com/apk/res/android">
    <CheckBoxPreference
        android:key="music"
        android:title="@string/music_title"
        android:summary="@string/music_summary"
        android:defaultValue="true" />
    <CheckBoxPreference
        android:key="hints"
        android:title="@string/hints_title"
        android:summary="@string/hints_summary"
        android:defaultValue="true" />
</PreferenceScreen>
```

Sudoku程序中有两个设置：一个用于播放背景音乐，另一个用于显示提示信息。这两个首选项的键值都是字符串常量，将被存储在Android的首选项数据库中。

接下来定义Settings类，它继承自PreferenceActivity类：

```
Sudoku\src\org\example\sudoku\Settings.java
package org.example.sudoku;

import android.os.Bundle;
import android.preference.PreferenceActivity;

public class Settings extends PreferenceActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.settings);
    }
}
```

`addPreferencesFromResource()`方法从XML文件中读取设置定义，并将其解压缩到当前活动的视图中。所有耗时的解压缩工作都在`PreferenceActivity`类中处理。

不要忘了在`AndroidManifest.xml`文件中注册`Settings`活动：

```
Sudoku\src\AndroidManifest.xml
<activity android:name=".Settings"
    android:label="@string/settings_title">
</activity>
```

现在重新运行`Sudoku`程序，按下`Menu`按钮，选择`Settings...`菜单项，你会发现，`Sudoku`设置页面出现了（参见图3-11）。试着修改这些值并退出程序，然后再启动该程序，确定这些首选项确实生效了。



图3-11 内容并不多，但确实很轻松地做到了

第6章会介绍可读取这些设置并使用它们完成任务的代码。现在讨论一下`NextGame`按钮。

3.9 开始新游戏

如果玩过数独游戏,就会知道该游戏有时候非常容易,而有时候简直令人抓狂。因此,当用户选择New Game按钮时,就应该弹出一个对话框,要求用户从3个难度级别中选择一个。在Android中,从事件列表中选择某个事件非常容易实现。首先,需要在res/values/strings.xml文件中再添加几个字符串:

Sudoku\res/values/strings.xml

```
<string name="new_game_title">Difficulty</string>
<string name="easy_label">Easy</string>
<string name="medium_label">Medium</string>
<string name="hard_label">Hard</string>
```

3

其次,使用数组资源在res/values/arrays.xml文件中创建难度列表:

Sudoku\res/values/arrays.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <array name="difficulty">
        <item>@string/easy_label</item>
        <item>@string/medium_label</item>
        <item>@string/hard_label</item>
    </array>
</resources>
```

然后,在Sudoku类中再导入几个包:

Sudoku\src/org/example/sudoku/Sudoku.java

```
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.util.Log;
```

最后,在onClick()方法的switch语句中添加处理New Game按钮单击事件的代码:

Sudoku\src/org/example/sudoku/Sudoku.java

```
case R.id.new_button:
    openNewGameDialog();
    break;
```

其中,openNewGameDialog()方法的作用是创建一个处理难度列表的用户界面。

Sudoku\src/org/example/sudoku/Sudoku.java

```
private static final String TAG = "Sudoku";
```

```

/** Ask the user what difficulty level they want */
private void openNewGameDialog() {
    new AlertDialog.Builder(this)
        .setTitle(R.string.new_game_title)
        .setItems(R.array.difficulty,
            new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialogInterface,
                    int i) {
                    startGame(i);
                }
            })
        .show();
}

/** Start a new game with the given difficulty level */
private void startGame(int i) {
    Log.d(TAG, "Clicked on " + i);
    // Start game here...
}

```

setItems()方法有两个参数：条目列表的资源ID和一个监听器，用户选择某个条目时，该监听器将被调用。

现在，运行该程序并按下New Game，将会出现如图3-12中所示的对话框。

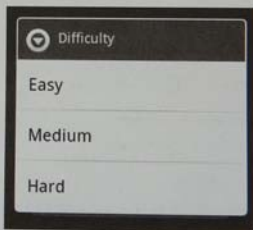


图3-12 选择难度级别对话框

实际上，此时我们并不打算开始一次新游戏。因此，当用户选择某个难度级别时，该程序只是利用Log.d()方法输出一条调试消息。Log.d()方法接收两个参数：一个标记字符串和要输出的一条消息。

3.10 利用日志消息调试程序

Log类提供了下面几个静态方法，可以将各种严重级别的消息输出到Android的

系统日志中。

- ❑ Log.eO: 错误;
- ❑ Log.wO: 警告;
- ❑ Log.iO: 信息;
- ❑ Log.dO: 调试;
- ❑ Log.vO: 详细。

用户永远也看不到该日志,但是作为开发人员,你可以用两种方法查看该日志。在Eclipse环境下,依次选择Window > Show View > Other... > Android > LogCat可以打开LogCat视图(参见图3-13)。可以根据严重性或调用Log.dO方法时指定的标记对该视图进行过滤。

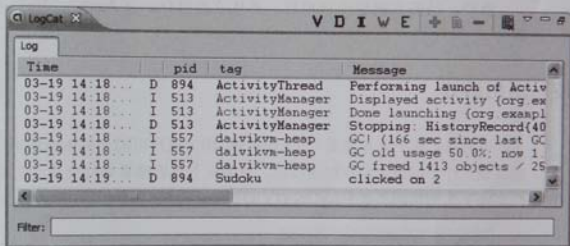


图3-13 LogCat视图中的调试输出

如果没有使用Eclipse,那么执行adb logcat命令可得到相同的输出^①。建议在一个独立的窗口中执行该命令,并在模拟器运行期间让该命令一直运行。它不会干扰任何其他的监听器。

无论如何强调Android日志在开发过程中的作用都不过分。还记得在前面选择About按钮时出现的错误吗(参见图3-7)?如果当时打开LogCat视图,就可以看到这样的消息:ActivityNotFoundException: Unable to find explicit activity class...have you declared this activity in your AndroidManifest.xml? (ActivityNotFoundException: 无法找到显式的Activity类……,你在AndroidManifest.xml中声明该活动了吗?),这条

① 参见网页<http://androidappdocs.appspot.com/guide/developing/tools/adb.html>。

消息解释的内容再明白不过了。

3.11 利用调试器调试程序

除了日志消息以外，还可以使用Eclipse调试器设置断点、单步执行以及查看程序状态。首先，通过在AndroidManifest.xml文件中添加`android:debuggable="true"`选项，使项目支持调试^①。

```
Sudokuv1/AndroidManifest.xml
<application android:icon="@drawable/icon"
    android:label="@string/app_name"
    android:debuggable="true">
```

然后，只需右键单击该项目，并依次选择Debug As > Android Application即可。

3.12 退出游戏

该游戏实际上并不需要一个Exit按钮，因为用户可能会按Back键或Home键来完成其他工作。但是我想添加该按钮，目的在于告诉读者如何终止一个活动。为此，只需将下面的代码添加到onClick()方法的switch语句部分：

```
Sudokuv1/src/org/example/sudoku/Sudoku.java
case R.id.exit_button:
    finish();
    break;
```

按下Exit按钮时，会调用finish()方法。该方法将结束当前活动，并将控制权交给Android应用程序栈中的下一个活动（通常是Home屏幕）。

3.13 快速阅读指南

本章内容很多，首先讲述了如何使用布局文件来组织各个用户界面，以及如何使用Android的文本、颜色及其他资源。然后又介绍了如何添加像按钮和文本字段

① 如果使用的是模拟器，该选项是可选的；但如果使用的是真实设备，则该选项是必选的。一定要记得在公开发布代码之前删除该选项。

这样的控制项、应用主题以更改程序的外观，甚至是添加菜单和首选项，对程序进行更好的控制。

Android是一个复杂的系统，读者不必等到掌握了其全部内容之后才开始自己的编程之旅。在编程过程中需要帮助时，数百页的在线参考资料包含了对此处用到的所有类和方法的深入介绍^①。

第4章将介绍如何使用Android的图形API来绘制数独游戏的单元格。

^① 要查看这些在线文档，可以打开Android SDK安装位置中的documentation.html文件，或者使用浏览器打开<http://androidappdocs.appspot.com/index.html>。

到目前为止，我们已经讲述了Android的基本概念和原理，介绍了如何创建带有几个按钮和对话框的简单用户界面，使读者开始领略到Android程序设计的精髓。但是，一些有趣的内容还没有涉及。

漂亮的图形会使任何应用程序增色不少。Android提供的图形库是当今功能最强大的图形库之一，可以灵活方便地在移动设备上使用。实际上，Android提供了两种图形库：一个用于二维图形，另一个用于三维图形^①。

本章介绍了2D图形，以及如何使用该内容实现数独示例程序的游戏部分。第10章将使用OpenGL ES库介绍3D图形。

4.1 Android 图形基础

Android在其android.graphics包中提供了完整的本机二维图形库。初步了解诸如Color和Canvas这样的图形类后，你很快就可以掌握这部分内容并运用这些知识进行绘图。

4.1.1 Color 类

Android中的颜色是用4个数字表示的，透明度、红色、绿色和蓝色（Alpha、Red、Green和Blue，ARGB）各占一个数字。由于每个数字有256个（8位）可能的值，因此一种颜色通常表示为一个32位整数。为了提高效率，Android代码使用整数而不是用Color类的实例来表示颜色。

^① Android已经在考虑四维图形功能，但由于时间关系目前尚未提供。

红色、绿色和蓝色的含义不言而喻，但是透明度则不然。透明度衡量了颜色的透明程度，其最小值为0，表示该像素的颜色完全透明；最大值为255，表示该像素的颜色完全不透明。如果Alpha值为0，则RGB的值将不起任何作用。透明度为0~255之间时，表示该像素的颜色半透明。在半透明状态下，可以在一定程度上看到前景中正在绘制的对象背后的内容。

要创建一个color对象，可以使用Color类的某个静态常量，代码如下：

```
int color = Color.BLUE; // solid blue
```

如果知道透明度、红色、绿色和蓝色的数值，也可以使用一个静态工厂方法：

```
// Translucent purple
color = Color.argb(127, 255, 0, 255);
```

不过，如果可能的话，最好在一个XML资源文件中定义所有的颜色。这样以后可以在一个地方轻松地更改这些颜色定义：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="mycolor">#7fff00ff</color>
</resources>
```

与在第3章中的用法一样，可以在其他XML文件中通过名称引用这些颜色定义，或者在Java代码中像下面这样引用它们：

```
color = getResources().getColor(R.color.mycolor);
```

getResources()方法返回当前活动的ResourceManager类，getColor()方法则要求管理器根据资源ID查找某个颜色。

1.2 Paint 类

Paint类是Android本机图形库中最重要的类之一。它包含样式、颜色以及绘制任何图形（包括位图、文本和几何图形）所需的其他信息。

通常情况下，你可能希望使用某种纯色在屏幕上绘图。此时，可以使用Paint.setColor()方法设置颜色。

例如：

```
cPaint.setColor(Color.LTGRAY);
```

这行代码使用了浅灰色的预定义颜色值。

4.1.3 Canvas 类

Canvas类代表可在其上绘图的画布。最初，画布启动时上面没有任何内容，就像高架投影机播放的空白幻灯片一样。利用Canvas类中的各种方法就可以在画布上绘制线条、矩形、圆以及其他任意图形。

Android中的显示屏是由Activity类的对象支配的，Activity类的对象引用View类的对象，而View类的对象又引用Canvas类的对象。通过重写View.onDraw()方法，可以在指定的画布上绘图。onDraw()方法唯一的参数就是说明要在哪个画布上绘图。

下面是一个名为Graphics的示例活动，它包含了一个名为GraphicsView的视图：

```
public class Graphics extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new GraphicsView(this));
    }

    static public class GraphicsView extends View {
        public GraphicsView(Context context) {
            super(context);
        }
        @Override
        protected void onDraw(Canvas canvas) {
            // Drawing commands go here
        }
    }
}
```

下一节将为onDraw()方法提供一些绘图命令。

4.1.4 Path 类

Path类包含一组矢量绘图命令，例如画线条、画矩形和画曲线等。下面的代码定义了一个圆形路径：

```
circle = new Path();
circle.addCircle(150, 150, 100, Direction.CW);
```

该代码所定义的圆：其圆心坐标为x=150，y=150；半径为100像素。由于已经

定义了路径，下面就可以利用该路径画圆并在圆内侧添加一圈文本：

```
private static final String QUOTE = "Now is the time for all " +
    "good men to come to the aid of their country.";
canvas.drawPath(circle, cPaint);
canvas.drawTextOnPath(QUOTE, circle, 0, 20, tPaint);
```

运行结果如图4-1所示。由于圆是按顺时针方向（`Direction.CW`）绘制的，所以文本也是按顺时针方向输出的。

Graphics



图4-1 环绕圆内侧绘制文本

为了实现很奇妙的图形特效，Android还提供了一些`PathEffect`类，利用这些类可执行很多操作，如随机置换路径，用曲线让沿着某个路径分布的所有线段变平滑或者将曲线变成多个线段，以及实现其他特殊效果。

4.1.5 Drawable 类

Android中的`Drawable`类主要针对像位图或纯色这样只用于显示的视觉元素，可以将其与其他图形结合使用，或者用于用户界面部件（例如作为按钮或视图的背景）中。

`Drawable`类支持下列格式。

- **Bitmap（位图）**。PNG或JPEG图像。
- **NinePatch（九宫格）**。一种可扩展的PNG图像，因其最初将整个图像划分为9个部分而得名。主要用作大小可调整的位图按钮的背景。
- **Shape（形状）**。基于`Path`类的矢量绘图命令。这是一种简化的SVG（**Scalable Vector Graphics**，可缩放矢量图形）格式。

- **Layers (图层)**。盛放子绘图区的容器，支持在按某些Z顺序堆叠的可绘图区上绘图。
- **States (状态)**。一个容器，能根据可绘图区的状态（位掩码）显示其某个子可绘图区。主要用于选择不同的按钮并设置按钮的焦点状态。
- **Levels (级别)**。一个容器，能根据可绘图区的级别（一系列整数）显示其某个子可绘图区。用于电池或信号强度指示器。
- **Scale (缩放)**。包含一个子绘图区的容器，能根据可绘图区的当前级别调整其大小。一种用途是可缩放的图片查看器。

可绘图区 (Drawable) 几乎总是在XML中定义。下面是一个常见的例子，定义了从一种颜色向另一种颜色渐变（本例中是从白色到灰色）的可绘图区，**angle**指定了渐变的方向（270° 表示从上向下）。这个可绘图区可用作视图的背景。

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <gradient
        android:startColor="#FFFFFF"
        android:endColor="#808080"
        android:angle="270" />
</shape>
```

使用这个可绘图区的方法有两种：一是在XML中以**android:background=**属性这种形式引用它；二是在视图的**onCreate()**方法中像下面这样调用**Canvas.setBackgroundResource()**方法：

```
setBackgroundResource(R.drawable.background);
```

这让**GraphicsView**示例视图有了非常漂亮的渐变背景，参见图4-2。

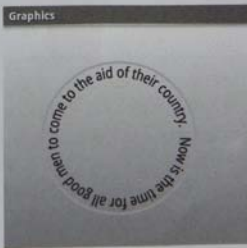


图4-2 使用在XML中定义的渐变背景

4.2 在 Sudoku 程序中添加图形

现在，我们把所学的知识应用到Sudoku示例程序中。在第3章中，我们已经创建了数独游戏的启动界面，实现了About对话框和开始新游戏的功能。但是漏掉了一个非常重要的内容：数独游戏本身！下面，我们将使用本机2D图形库来实现游戏部分。

4.2.1 开始游戏

首先需要编写可开始游戏的代码。startGame()方法接收一个参数，即从难度列表表中选择的难度名称索引，该方法的定义如下：

```
SudokuV2/src/org/example/sudoku/Sudoku.java
/** Start a new game with the given difficulty level */
private void startGame(int i) {
    Log.d(TAG, "clicked on " + i);
    Intent intent = new Intent(Sudoku.this, Game.class);
    intent.putExtra(Game.KEY_DIFFICULTY, i);
    startActivity(intent);
}
```

Sudoku程序的游戏部分将使用另一个名为Game的活动，因此需要创建一个新的Intent对象来启动游戏。我们将表示难度级别的数字赋给Intent对象的extraData域，然后调用startActivity()方法启动这个新活动。

extraData域是一个将被传递给Intent对象的键值对映射。键是String类型，值可以是任何基本数据类型、基本数据类型数组、Bundle类型以及Serializable或Parcelable的子类。在实际的程序中，你可能需要将包名作为键名的前缀。

数独轶事

“数字填空”谜题在美国发表几年后，日本的Nikoli出版社发现了该谜题游戏，并为它想了一个听起来更酷的名称——Sudoku（在日语中的意思是“唯一的数字”）。从此，数独游戏开始在世界各地流传开来，后来的事就众所周知了。可惜的是，Garns在1989年就去世了，未能亲眼看到他发明的这一游戏风靡全球。

4.2.2 定义 Game 类

Game活动的内容如下：

```

SudokuV2/src/org/example/sudoku/Game.java
package org.example.sudoku;

import android.app.Activity;
import android.app.Dialog;
import android.os.Bundle;
import android.util.Log;
import android.view.Gravity;
import android.widget.Toast;

public class Game extends Activity {
    private static final String TAG = "Sudoku";

    public static final String KEY_DIFFICULTY = "difficulty";
    public static final int DIFFICULTY_EASY = 0;
    public static final int DIFFICULTY_MEDIUM = 1;
    public static final int DIFFICULTY_HARD = 2;

    private int puzzle[] = new int[9 * 9];

    private PuzzleView puzzleView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Log.d(TAG, "onCreate");

        int diff = getIntent().getIntExtra(KEY_DIFFICULTY,
            DIFFICULTY_EASY);
        puzzle = getPuzzle(diff);
        calculateUsedTiles();

        puzzleView = new PuzzleView(this);
        setContentView(puzzleView);
        puzzleView.requestFocus();
    }
    // ...
}

```

`onCreate()` 方法首先从 `Intent` 对象中提取出表示难度的数字并选择一局要玩的游戏，然后创建一个 `PuzzleView` 类的实例，用 `PuzzleView` 类作为新的视图内容。由于这是一个完全自定义的视图，因此使用代码实现其功能比使用 XML 更容易。

对于数独盘面上的每一个单元格，`calculateUsedTiles()` 方法（此处未显示）根据数独游戏的规则计算哪些数字对该单元格无效（因为这些数字已经在该单元格所在的行、列或 3×3 宫部分中的其他单元格内出现过了）。

由于这是一个活动，所以需要在 `AndroidManifest.xml` 文件中对其注册：

Sudokuv2/AndroidManifest.xml

```
<activity android:name=".Game"
    android:label="@string/game_title"/>
```

另外，还需要在res/values/strings.xml文件中添加几个字符串资源：

Sudokuv2/res/values/strings.xml

```
<string name="game_title">Game</string>
<string name="no_moves_label">No moves</string>
<string name="keypad_title">Keypad</string>
```

4.2.3 定义 PuzzleView 类

下面需要定义PuzzleView类。这次我们完全使用Java来定义PuzzleView类，而不用XML布局。

PuzzleView类的内容如下：

Sudokuv2/src/org/example/sudoku/PuzzleView.java

```
package org.example.sudoku;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.Rect;
import android.graphics.Paint.FontMetrics;
import android.graphics.Paint.Style;
import android.util.Log;
import android.view.KeyEvent;
import android.view.MotionEvent;
import android.view.View;
import android.view.animation.AnimationUtils;

public class PuzzleView extends View {
    private static final String TAG = "Sudoku";
    private final Game game;
    public PuzzleView(Context context) {
        super(context);
        this.game = (Game) context;
        setFocusable(true);
        setFocusableInTouchMode(true);
    }
    // ...
}
```

在PuzzleView类的构造函数中，我们一直引用Game类，并通过设置选项允许用户在视图中输入。在PuzzleView类中，我们需要实现onSizeChanged()方法。该方法在视图被创建并且Android确定了视图大小以后被调用。

```
SudokuV2/src/org/example/sudoku/PuzzleView.java
```

```
private float width;    // width of one tile
private float height;   // height of one tile
private int selX;       // X index of selection
private int selY;       // Y index of selection
private final Rect selRect = new Rect();

@Override
protected void onSizeChanged(int w, int h, int oldw, int oldh) {
    width = w / 9f;
    height = h / 9f;
    getRect(selX, selY, selRect);
    Log.d(TAG, "onSizeChanged: width " + width + ", height "
        + height);
    super.onSizeChanged(w, h, oldw, oldh);
}
```

我们使用 `onSizeChanged()` 方法来计算屏幕上每个单元格的大小，其宽度和高度分别等于整个视图宽度和高度的 $1/9$ 。注意，这里的宽度和高度值都是浮点数，因此最终可能得到几分之一的一个像素。`selRect` 表示一个矩形，随后我们将使用该矩形跟踪选择光标。

至此，我们已经创建了游戏的视图并确定了视图的大小。接下来要绘制网格线，将游戏盘面上的各个单元格分开。

视图到底有多大

刚开始学习 Android 的开发人员常犯的错误是在视图的构造函数中使用视图的宽度和高度。实际上，调用一个视图的构造函数时，Android 尚不知道该视图的大小，因此视图的宽度和高度值都被设为 0。真正计算这些值是在布局阶段，此阶段发生在构造函数被调用之后、任何对象被绘制之前。可以使用 `onSizeChanged()` 方法在视图的宽度和高度值已知后通知这些值发生了变化，也可以随后在其他方法（例如 `onDraw()` 方法）中使用 `getWidth()` 和 `getHeight()` 方法获得其值。

4.2.4 绘制游戏盘面

每次需要更新视图的任何部分时，Android 就会调用视图的 `onDraw()` 方法。为简单起见，`onDraw()` 方法表面上是从头开始重绘整个屏幕。实际上，真正需要重绘的可能只是视图的一小部分，具体区域由画布的裁剪矩形定义。Android 负责替你完成裁剪操作。

首先在 `res/values/colors.xml` 文件中定义几种绘图时需要使用的新颜色：

Sudoku2/res/values/colors.xml

```

<color name="puzzle_background">#ffe6f0ff</color>
<color name="puzzle_hilite">#ffffffff</color>
<color name="puzzle_light">#64c6d4ef</color>
<color name="puzzle_dark">#6456648f</color>
<color name="puzzle_foreground">#ff000000</color>
<color name="puzzle_hint_0">#64ff0000</color>
<color name="puzzle_hint_1">#6400ff80</color>
<color name="puzzle_hint_2">#2000ff80</color>
<color name="puzzle_selected">#64ff8000</color>

```

其他实现方法

编写Sudoku示例程序时，我曾经尝试用几种不同的方法实现游戏盘面，例如为每个单元格使用一个按钮，或者使用XML声明一个ImageView类的网格等。经过多次失败后我发现，整个游戏盘面使用一个视图并在该视图中绘制线条和数字，是实现该应用程序最快捷、最简单的方法。

当然，这种方法也并非尽善尽美，例如，该方法需要绘制选定区域并显式处理键盘和屏幕触摸事件。设计自己的程序时，建议读者首先使用标准的部件和视图。只有在标准的部件和视图无法满足需求时，再考虑使用自定义绘图。

onDraw()方法的内容如下：

Sudoku2/src/org/example/sudoku/PuzzleView.java

```

@Override
protected void onDraw(Canvas canvas) {
    // Draw the background...
    Paint background = new Paint();
    background.setColor(getResources().getColor(
        R.color.puzzle_background));
    canvas.drawRect(0, 0, getWidth(), getHeight(), background);

    // Draw the board...
    // Draw the numbers...
    // Draw the hints...
    // Draw the selection...
}

```

onDraw()方法的第一个参数是Canvas，指明了要在哪个画布上绘制图形。上面的代码只是用puzzle_background颜色绘制了游戏盘面的背景。

下面是在游戏盘面上绘制网格线的代码：

Sudoku2/src/org/example/sudoku/PuzzleView.java

```

// Draw the board...
// Define colors for the grid lines

```



```

Paint dark = new Paint();
dark.setColor(getResources().getColor(R.color.puzzle_dark));

Paint hilite = new Paint();
hilite.setColor(getResources().getColor(R.color.puzzle_hilite));

Paint light = new Paint();
light.setColor(getResources().getColor(R.color.puzzle_light));

// Draw the minor grid lines
for (int i = 0; i < 9; i++) {
    canvas.drawLine(0, i * height, getWidth(), i * height,
        light);
    canvas.drawLine(0, i * height + 1, getWidth(), i * height
        + 1, hilite);
    canvas.drawLine(i * width, 0, i * width, getHeight(),
        light);
    canvas.drawLine(i * width + 1, 0, i * width + 1,
        getHeight(), hilite);
}

// Draw the major grid lines
for (int i = 0; i < 9; i++) {
    if (i % 3 != 0)
        continue;
    canvas.drawLine(0, i * height, getWidth(), i * height,
        dark);
    canvas.drawLine(0, i * height + 1, getWidth(), i * height
        + 1, hilite);
    canvas.drawLine(i * width, 0, i * width, getHeight(), dark);
    canvas.drawLine(i * width + 1, 0, i * width + 1,
        getHeight(), hilite);
}

```

上面的代码使用3种不同的颜色来绘制网格线：单元格之间使用浅色的线分隔；宫与宫之间使用深色的线分隔；每个单元格的边缘使用高亮的颜色显示，使其看起来具有一定深度。这些网格线的绘制顺序也很重要，因为后绘制的线会覆盖先绘制的线。使用这些代码绘制的游戏盘面如图4-3所示。下面需要在这些单元格中填入数字。

4.2.5 绘制数字

下面是在单元格中绘制游戏数字的代码。其中棘手的地方在于如何获得每个数字的位置和大小，以使其正好位于单元格的中心。

```

Sudoku2/src/org/example/sudoku/PuzzleView.java

// Draw the numbers...
// Define color and style for numbers
Paint foreground = new Paint(Paint.ANTI_ALIAS_FLAG);

```



```

foreground.setColor(getResources().getColor(
    R.color.puzzle_foreground));
foreground.setStyle(Style.FILL);
foreground.setTextSize(height * 0.75f);
foreground.setTextScaleX(width / height);
foreground.setTextAlign(Paint.Align.CENTER);

// Draw the number in the center of the tile
FontMetrics fm = foreground.getFontMetrics();
// Centering in X: use alignment (and X at midpoint)
float x = width / 2;
// Centering in Y: measure ascent/descent first
float y = height / 2 - (fm.ascent + fm.descent) / 2;
for (int i = 0; i < 9; i++) {
    for (int j = 0; j < 9; j++) {
        canvas.drawText(this.game.getTileString(i, j), i
            * width + x, j * height + y, foreground);
    }
}
}

```

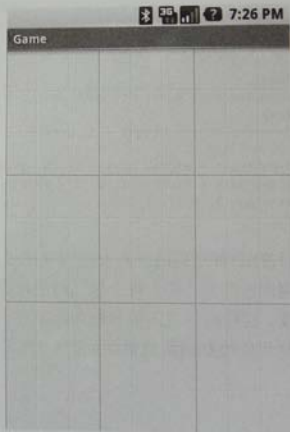


图4-3 使用3种颜色绘制网格线以增强效果

为了计算数字的大小，我们将字体高度设为单元格高度的四分之三，将字体宽
宽高比设为与单元格的宽高比相同。因为我们希望该程序能在任何分辨率的屏幕
正常显示，所以这里不能使用绝对像素或磅大小。

要确定每个数字的位置，我们使其在x和y两个方向上都处于中间位置，x

向的起始位置容易计算，只需将单元格的宽度除以2即可。但是对于y方向来说，我们必须将起始位置向下移一点，使单元格的中点与数字的中点对齐，而不是与数字的基线对齐。我们使用图形库中的FontMetrics类来获得数字在垂直方向上的总长度，然后将其除以2作为下移距离。数字在单元格内居中的效果如图4-4所示。

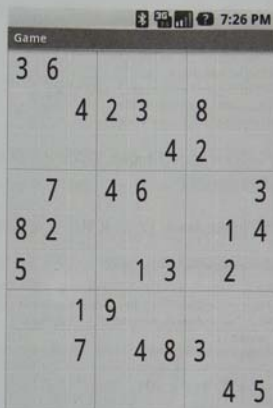


图4-4 在单元格内居中显示的数字

这部分代码负责显示游戏的初始（已知）数字。下一步是允许玩家在所有空白单元格中输入其猜测的数字。

4.3 处理输入

与iPhone等手机开发平台相比，Android程序设计要考虑到Android手机具有不同的形状、大小和各种输入方式。这些输入方式涉及键盘、D-pad、触摸屏、轨迹球或前面各种方式的组合。

因此，一个功能良好的Android程序需要能够支持任何可用的输入硬件，就像程序要能支持任何屏幕分辨率一样。

4.3.1 定义和更新选定区域

首先我们将实现一个小光标，以显示玩家当前选定的单元格。选定的单元格是指当玩家输入一个数字时将被修改内容的那个单元格。onDraw()方法中绘制选定区域的代码如下：

```
SudokuV2/src/org/example/sudoku/PuzzleView.java
// Draw the selection...
Log.d(TAG, "selRect=" + selRect);
Paint selected = new Paint();
selected.setColor(getResources().getColor(
    R.color.puzzle_selected));
canvas.drawRect(selRect, selected);
```

这里使用此前在onSizeChanged()方法中计算出的选择矩形在选定单元格上绘制出具有一定透明度的颜色。

下面通过重写onKeyDown()方法来更改选定区域：

```
SudokuV2/src/org/example/sudoku/PuzzleView.java
@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    Log.d(TAG, "onKeyDown: keycode=" + keyCode + ", event="
        + event);
    switch (keyCode) {
        case KeyEvent.KEYCODE_DPAD_UP:
            select(selX, selY - 1);
            break;
        case KeyEvent.KEYCODE_DPAD_DOWN:
            select(selX, selY + 1);
            break;
        case KeyEvent.KEYCODE_DPAD_LEFT:
            select(selX - 1, selY);
            break;
        case KeyEvent.KEYCODE_DPAD_RIGHT:
            select(selX + 1, selY);
            break;
        default:
            return super.onKeyDown(keyCode, event);
    }
    return true;
}
```

如果用户使用的是十字方向键（D-pad），当他们按下、下、左或右按钮时，select()方法会被调用，以在该方向上移动选择光标。

如何处理轨迹球呢？当然可以通过重写onTrackballEvent()方法来实现。但实

实际上如果你没有处理轨迹球事件，Android会自动把轨迹球事件转换为D-pad事件。因此，Sudoku示例程序中没有处理轨迹球事件。

`select()`方法首先计算选定区域新的x和y坐标，然后再次调用`getRect()`方法计算新的选择矩形：

`Sudoku2/src/org/example/sudoku/PuzzleView.java`

```
private void select(int x, int y) {
    invalidate(selRect);
    selX = Math.min(Math.max(x, 0), 8);
    selY = Math.min(Math.max(y, 0), 8);
    getRect(selX, selY, selRect);
    invalidate(selRect);
}

private void getRect(int x, int y, Rect rect) {
    rect.set((int) (x * width), (int) (y * height), (int) (x
        * width + width), (int) (y * height + height));
}
```

注意，`select()`方法两次调用了`invalidate()`方法。第一次调用通知Android原选择矩形覆盖的区域（图4-5中左图上的黄色单元格）需要重绘。第二次调用通知Android新的选择区域（图4-5中右侧的黄色单元格）也需要重绘。实际上这里并没有进行绘图操作。



图4-5 绘制并改变选定区域

永远不要在`onDraw()`方法以外的地方调用任何绘图函数，谨记这一点！这里只是用`invalidate()`方法将矩形标记为已过期。窗口管理器将在未来的某个时候对所有标记为已过期的矩形一起调用`onDraw()`方法。这些已过期的矩形成为裁剪区域，只重绘这些发生变化的区域，从而优化了屏幕更新操作。

优化刷新

在Sudoku示例程序的早期版本中,任何时刻移动光标都会引起整个屏幕的更新。因此,每当有键被按下时,整个游戏盘面都必须重绘。这明显降低了程序的运行速度。将这部分代码改为只在最少的选择矩形发生变化时才更新整个屏幕,将会显著提高程序的运行速度。

下面提供的方法可供玩家在选定的单元格中输入新数字。

4.3.2 输入数字

要处理键盘输入,只需在onKeyDown()方法中再增加几个对应于数字0到9的case分支(0或空格键表示清除单元格中的数字)。

SudokuV2/src/org/example/sudoku/PuzzleView.java

```
case KeyEvent.KEYCODE_0:
case KeyEvent.KEYCODE_SPACE: setSelectedTile(0); break;
case KeyEvent.KEYCODE_1:   setSelectedTile(1); break;
case KeyEvent.KEYCODE_2:   setSelectedTile(2); break;
case KeyEvent.KEYCODE_3:   setSelectedTile(3); break;
case KeyEvent.KEYCODE_4:   setSelectedTile(4); break;
case KeyEvent.KEYCODE_5:   setSelectedTile(5); break;
case KeyEvent.KEYCODE_6:   setSelectedTile(6); break;
case KeyEvent.KEYCODE_7:   setSelectedTile(7); break;
case KeyEvent.KEYCODE_8:   setSelectedTile(8); break;
case KeyEvent.KEYCODE_9:   setSelectedTile(9); break;
case KeyEvent.KEYCODE_ENTER:
case KeyEvent.KEYCODE_DPAD_CENTER:
    game.showKeypadOrError(selX, selY);
    break;
```

要支持D-pad,只需检查onKeyDown()方法中的case分支,当输入为回车键或D-pad的中心按钮时弹出一个软键盘,允许用户选择要输入的数字。

对于触摸屏,可重写onTouchEvent()方法,并显示同样的软键盘,下面是方法定义:

SudokuV2/src/org/example/sudoku/PuzzleView.java

```
@Override
public boolean onTouchEvent(MotionEvent event) {
    if (event.getAction() != MotionEvent.ACTION_DOWN)
        return super.onTouchEvent(event);

    select((int) (event.getX() / width),
           (int) (event.getY() / height));
}
```

```

game.showKeypadOnError(selX, selY);
Log.d(TAG, "onTouchEvent: x " + selX + ", y " + selY);
return true;
}

```

所有的方法最终都殊途同归，落到对setSelectedTile()方法的调用上，以修单元格中的数字：

```

SudokuV2/src/org/example/sudoku/PuzzleView.java

public void setSelectedTile(int tile) {
    if (game.setTileIfValid(selX, selY, tile)) {
        invalidate(); // may change hints
    } else {
        // Number is not valid for this tile
        Log.d(TAG, "setSelectedTile: invalid: " + tile);
    }
}

```

注意，这里调用的是不带参数的invalidate()方法。这表示将整个屏幕标记为已过期（这违反了我自己在前面提出的建议）。但是在这种情况下必须这样处理，因为输入新数字或删除旧数字后都会引起提示的改变，我们将在下一节中实现提示功能。

3.3 增加提示

如何既给玩家一些帮助，但又不会越俎代庖为其解答整个谜题？根据每个单元格可填数字的数目为其绘制不同的背景如何？将下面的代码添加到onDraw()方法中绘制选择区域之前的位置：

```

SudokuV2/src/org/example/sudoku/PuzzleView.java

// Draw the hints...
// Pick a hint color based on #moves left
Paint hint = new Paint();
int c[] = { getResources().getColor(R.color.puzzle_hint_0),
    getResources().getColor(R.color.puzzle_hint_1),
    getResources().getColor(R.color.puzzle_hint_2), };
Rect r = new Rect();
for (int i = 0; i < 9; i++) {
    for (int j = 0; j < 9; j++) {
        int movesleft = 9 - game.getUsedTiles(i, j).length;
        if (movesleft < c.length) {
            getRect(i, j, r);
            hint.setColor(c[movesleft]);
            canvas.drawRect(r, hint);
        }
    }
}
}

```

这里使用3种状态分别表示单元格中可能填入0、1和2个数字。如果某个单元格可能填入的数字为0个，则表示玩家在某个地方出错了，需要向后回退一步或几步。

增加提示之后的结果如图4-6所示，你能在其中找出玩家所犯的错误吗^①？

3	6	2				4		
		4	2	3		8		
		5			4	2		
	7	9	4	6		5	3	
8	2	3				6	1	4
5	4	6		1	3	9	2	7
4		1	9		5		6	
		7	1	4	8	3	9	2
		8	7	2		1	4	5

图4-6 根据单元格中可填数字的数目高亮显示单元格

4.3.4 抖动屏幕

如果用户试图输入一个明显无效的数字，例如一个在该单元格所在宫中已经出现过的数字，又该如何处理呢？为了增强趣味性，可在用户试图输入无效数字时让屏幕来回抖动。下面在setSelectedTile()方法中添加对无效数字情况的处理：

```
SudokuV2/src/org/example/sudoku/PuzzleView.java
// Number is not valid for this tile
Log.d(TAG, "setSelectedTile: invalid: " + tile);
startAnimation(AnimationUtils.loadAnimation(game,
    R.anim.shake));
```

上面的代码加载并运行一个名为R.anim.shake的资源，该资源在res/anim/

① 瞧下面一行中间宫内的两个数字都是错误的。

shake.xml文件中定义，它可在左右方向上以10个像素为单位抖动屏幕，抖动持续时间为1000毫秒（1秒）。

```
SudokuV2/res/anim/shake.xml
<?xml version="1.0" encoding="utf-8"?>
<translate
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:fromXDelta="0"
    android:toXDelta="10"
    android:duration="1000"
    android:interpolator="@anim/cycle_7" />
```

该动画的运行次数、速度和加速度都是由使用XML定义的一个动画插补器控制的。

```
SudokuV2/res/anim/cycle_7.xml
<?xml version="1.0" encoding="utf-8"?>
<cycleInterpolator
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:cycles="7" />
```

这个特定的动画插补器会让该动画重复运行7次。

4.4 其他问题

下面我们回过头来处理一些枝节问题，首先从Keypad类开始。这部分内容对于程序的编译和运行是不可或缺的，但是与绘图没有任何关系。如果你愿意，可以直接跳至4.5节。

4.4.1 创建软键盘

对于没有键盘的手机来说，软键盘为用户提供了很大的方便。软键盘在游戏盘面的前端显示某个活动的网格，其中含有1~9的数字。软键盘对话框的作用就是返回玩家所选择的数字。

位于res/layout/keypad.xml文件中的软键盘用户界面布局如下：

```
SudokuV2/res/layout/keypad.xml
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/keypad">
```



```

        android:orientation="vertical"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:stretchColumns="*">
<TableRow>
    <Button android:id="@+id/keypad_1"
        android:text="1">
    </Button>
    <Button android:id="@+id/keypad_2"
        android:text="2">
    </Button>
    <Button android:id="@+id/keypad_3"
        android:text="3">
    </Button>
</TableRow>
<TableRow>
    <Button android:id="@+id/keypad_4"
        android:text="4">
    </Button>
    <Button android:id="@+id/keypad_5"
        android:text="5">
    </Button>
    <Button android:id="@+id/keypad_6"
        android:text="6">
    </Button>
</TableRow>
<TableRow>
    <Button android:id="@+id/keypad_7"
        android:text="7">
    </Button>
    <Button android:id="@+id/keypad_8"
        android:text="8">
    </Button>
    <Button android:id="@+id/keypad_9"
        android:text="9">
    </Button>
</TableRow>
</TableLayout>

```

接下来定义Keypad类，其内容如下：

```
Sudoku2/src/org/example/sudoku/Keypad.java
```

```

package org.example.sudoku;

import android.app.Dialog;
import android.content.Context;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.View;

public class Keypad extends Dialog {

```

```
    protected static final String TAG = "Sudoku";
```

```

private final View keys[] = new View[9];
private View keypad;

private final int useds[];
private final PuzzleView puzzleView;

public Keypad(Context context, int useds[], PuzzleView puzzleView) {
    super(context);
    this.useds = useds;
    this.puzzleView = puzzleView;
}

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.keypad);
    findViews();
    for (int element : useds) {
        if (element != 0)
            keys[element - 1].setVisibility(View.INVISIBLE);
    }
    setListeners();
}

// ...
}

```

如果某个特定的数字是无效的（例如同一个数字在该行中已经出现过），可让软键盘中的该数字不可见，从而使玩家无法选择该数字（参见图4-7）。

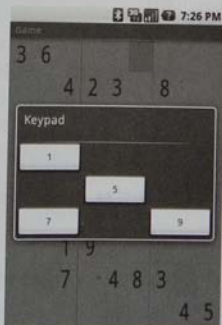


图4-7 在软键盘视图中隐藏无效的数字

`findViews()`方法提取并保存软键盘的所有键和软键盘主窗口的视图：

SudokuV2/src/org/example/sudoku/Keypad.java

```
private void findViews() {
    keypad = findViewById(R.id.keypad);
    keys[0] = findViewById(R.id.keypad_1);
    keys[1] = findViewById(R.id.keypad_2);
    keys[2] = findViewById(R.id.keypad_3);
    keys[3] = findViewById(R.id.keypad_4);
    keys[4] = findViewById(R.id.keypad_5);
    keys[5] = findViewById(R.id.keypad_6);
    keys[6] = findViewById(R.id.keypad_7);
    keys[7] = findViewById(R.id.keypad_8);
    keys[8] = findViewById(R.id.keypad_9);
}
```

setListeners()方法循环处理软键盘的各个键，并为每个键设置一个监听器。同时，该方法还为软键盘的主窗口设置了一个监听器：

SudokuV2/src/org/example/sudoku/Keypad.java

```
private void setListeners() {
    for (int i = 0; i < keys.length; i++) {
        final int t = i + 1;
        keys[i].setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                returnResult(t);
            }
        });
    }
    keypad.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            returnResult(0);
        }
    });
}
```

玩家选择软键盘上的某个按钮时，setListeners()方法用该按钮对应的数字作为参数调用returnResult()方法。如果玩家选择的是软键盘上按钮以外的其他地方，则用0作为参数调用returnResult()方法，表示删除该单元格中的数字。

玩家使用键盘输入一个数字时，则会调用onKeyDown()方法：

SudokuV2/src/org/example/sudoku/Keypad.java

```
@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    int tile = 0;
    switch (keyCode) {
        case KeyEvent.KEYCODE_0:
        case KeyEvent.KEYCODE_SPACE: tile = 0; break;
        case KeyEvent.KEYCODE_1: tile = 1; break;
        case KeyEvent.KEYCODE_2: tile = 2; break;
        case KeyEvent.KEYCODE_3: tile = 3; break;
    }
}
```

```

case KeyEvent.KEYCODE_4:    tile = 4; break;
case KeyEvent.KEYCODE_5:    tile = 5; break;
case KeyEvent.KEYCODE_6:    tile = 6; break;
case KeyEvent.KEYCODE_7:    tile = 7; break;
case KeyEvent.KEYCODE_8:    tile = 8; break;
case KeyEvent.KEYCODE_9:    tile = 9; break;
default:
    return super.onKeyDown(keyCode, event);
}
if (isValid(tile)) {
    returnResult(tile);
}
return true;
}

```

如果该数字对当前单元格而言是有效的，则onKeyDown()方法调用returnResult()方法；否则忽略本次击键。

isValid()方法用于检查给定的数字对当前单元格是否有效：

```

SudokuV2/src/org/example/sudoku/Keypad.java
private boolean isValid(int tile) {
    for (int t : useds) {
        if (tile == t)
            return false;
    }
    return true;
}

```

如果给定的数字出现在used数组中，则说明该数字对当前单元格无效，因为同一个数字已经出现在该单元格所在的行、列或宫中。

调用returnResult()方法时，它将玩家选定的数字返回给当前调用的活动：

```

SudokuV2/src/org/example/sudoku/Keypad.java
/** Return the chosen tile to the caller */
private void returnResult(int tile) {
    puzzleView.setSelectedTile(tile);
    dismiss();
}

```

此处调用PuzzleView.setSelectedTile()方法更改游戏盘面中当前单元格的数字，调用dismiss方法关闭Keypad对话框。

既然该活动已经有了返回结果，下面就在Game类中调用该活动并提取这一结果：

```
SudokuV2/src/org/example/sudoku/Game.java
```

```
/** Open the keypad if there are any valid moves */
protected void showKeypadOnError(int x, int y) {
    int tiles[] = getUsedTiles(x, y);
    if (tiles.length == 9) {
        Toast toast = Toast.makeText(this,
            R.string.no_moves_label, Toast.LENGTH_SHORT);
        toast.setGravity(Gravity.CENTER, 0, 0);
        toast.show();
    } else {
        Log.d(TAG, "showKeypad: used=" + toPuzzleString(tiles));
        Dialog v = new Keypad(this, tiles, puzzleView);
        v.show();
    }
}
```

要确定哪些数字是可用的，可以将extraData域中的字符串（包含所有已经用过的数字）作为参数传递给Keypad方法。

4.4.2 实现游戏逻辑

Game.java中其余的代码主要是为了实现游戏逻辑，特别是根据游戏规则确定哪些填入数字有效，哪些填入数字无效。setTileIfValid()方法是实现该功能的关键。给定一组x和y坐标以及一个单元格的新值，只有该数字对该坐标位置的单元格有效，setTileIfValid()方法才将该数字设为该单元格的新值。

```
SudokuV2/src/org/example/sudoku/Game.java
```

```
/** Change the tile only if it's a valid move */
protected boolean setTileIfValid(int x, int y, int value) {
    int tiles[] = getUsedTiles(x, y);
    if (value != 0) {
        for (int tile : tiles) {
            if (tile == value)
                return false;
        }
    }
    setTile(x, y, value);
    calculateUsedTiles();
    return true;
}
```

要检查填入的数字是否有效，可以为游戏盘面中的每个单元格建立一个数组。对每个单元格，此数组维护一个当前已经填入到该单元格所在行、列和宫的其他单元格中的数字列表。如果一个数字出现在这个列表中，说明这个数字对当前单元格无效。

getUsedTiles()方法用于提取给定位置单元格的数字列表:

```
SudokuV2/src/org/example/sudoku/Game.java
/** Cache of used tiles */
private final int used[][] = new int[9][9][];

/** Return cached used tiles visible from the given coords */
protected int[] getUsedTiles(int x, int y) {
    return used[x][y];
}
```

由于已填入到单元格中的数字数组的计算开销比较大,所以我们只是缓存该数组,仅在必要时才通过调用calculateUsedTiles()方法计算该数组:

```
SudokuV2/src/org/example/sudoku/Game.java
/** Compute the two dimensional array of used tiles */
private void calculateUsedTiles() {
    for (int x = 0; x < 9; x++) {
        for (int y = 0; y < 9; y++) {
            used[x][y] = calculateUsedTiles(x, y);
            // Log.d(TAG, "used[" + x + "][" + y + "] = "
            // + toString(used[x][y]));
        }
    }
}
```

calculateUsedTiles()方法只是在游戏盘面中每个单元格的位置调用calculateUsedTiles(x, y)方法:

```
SudokuV2/src/org/example/sudoku/Game.java
Line 1 /** Compute the used tiles visible from this position */
- private int[] calculateUsedTiles(int x, int y) {
-     int c[] = new int[9];
-     // horizontal
5     for (int i = 0; i < 9; i++) {
-         if (i == y)
-             continue;
-         int t = getTile(x, i);
-         if (t != 0)
10         c[t - 1] = t;
-     }
-     // vertical
-     for (int i = 0; i < 9; i++) {
-         if (i == x)
15         continue;
-         int t = getTile(i, y);
-         if (t != 0)
-             c[t - 1] = t;
-     }
20     // same cell block
```

```

-         int startx = (x / 3) * 3;
-         int starty = (y / 3) * 3;
-         for (int i = startx; i < startx + 3; i++) {
-             for (int j = starty; j < starty + 3; j++) {
25              if (i == x && j == y)
-                  continue;
-                  int t = getTile(i, j);
-                  if (t != 0)
-                      c[t - 1] = t;
30          }
-      }
-      // compress
-      int nused = 0;
-      for (int t : c) {
35          if (t != 0)
-              nused++;
-      }
-      int c1[] = new int[nused];
-      nused = 0;
40      for (int t : c) {
-          if (t != 0)
-              c1[nused++] = t;
-      }
-      return c1;
45 }

```

首先将数组中的各个元素初始化为0。在第5行中检查当前单元格所在行的其他单元格，如果某个单元格中已经填入了数字，则将该单元格中的数字添加到数组中。

在第13行中检查当前单元格所在列的其他单元格，第21行循环检查当前单元格所在宫的其他单元格。如果某个单元格中已经填入了数字，同样将该单元格中的数字添加到数组中。

最后一步从第33行开始，即对数组进行压缩，返回只包含非0值的数组。这样做的目的在于可以使用`array.length`迅速知道有多少数字已经填入到当前单元格所在的行、列和宫的其他单元格中。

4.4.3 其他功能

顺利实现Sudoku示例程序还涉及其他一些变化和实用功能。`easyPuzzle`、`mediumPuzzle`和`hardPuzzle`是3个硬编码的数独游戏，分别对应于“简单”、“中等”和“困难”3个难度级别。

```
SudokuV2/src/org/example/sudoku/Game.java
```

```
private final String easyPuzzle =
```

```

"360000000004230800000004200" +
"070460003820000014500013020" +
"001900000007048300000000045";
private final String mediumPuzzle =
"650000070000506000014000005" +
"007009000002314700000700800" +
"500000630000201000030000097";
private final String hardPuzzle =
"009000000080605020501078000" +
"000000700706040102004000000" +
"000720903090301080000000600";

```

getPuzzle()方法只是接收难度级别并返回一次数独游戏:

```
SudokuV2/src/org/example/sudoku/Game.java
```

```

/** Given a difficulty level, come up with a new puzzle */
private int[] getPuzzle(int diff) {
    String puz;
    // TODO: Continue last game
    switch (diff) {
        case DIFFICULTY_HARD:
            puz = hardPuzzle;
            break;
        case DIFFICULTY_MEDIUM:
            puz = mediumPuzzle;
            break;
        case DIFFICULTY_EASY:
            puz = easyPuzzle;
            break;
        default:
            puz = easyPuzzle;
            break;
    }
    return fromPuzzleString(puz);
}

```

4

后面我们将修改getPuzzle()方法, 以实现继续玩上一次游戏的功能。

toPuzzleString()方法将用整数数组表示的游戏转换为一个字符串, fromPuzzleString()方法的功能正好相反。

```
SudokuV2/src/org/example/sudoku/Game.java
```

```

/** Convert an array into a puzzle string */
static private String toPuzzleString(int[] puz) {
    StringBuilder buf = new StringBuilder();
    for (int element : puz) {
        buf.append(element);
    }
    return buf.toString();
}

/** Convert a puzzle string into an array */
static protected int[] fromPuzzleString(String string) {
    int[] puz = new int[string.length()];
    for (int i = 0; i < puz.length; i++) {

```



```

        puz[i] = string.charAt(i) - '0';
    }
    return puz;
}

```

getTile()方法接收单元格的x和y坐标，并返回该单元格中的当前数字。如果数字为0，则表示该单元格空白。

```

SudokuV2/src/org/example/sudoku/Game.java
/** Return the tile at the given coordinates */
private int getTile(int x, int y) {
    return puzzle[y * 9 + x];
}

/** Change the tile at the given coordinates */
private void setTile(int x, int y, int value) {
    puzzle[y * 9 + x] = value;
}

```

显示单元格时会用到getTileString()方法，该方法返回的字符串表示给定单元格中的数字。如果单元格为空，则返回一个空字符串。

```

SudokuV2/src/org/example/sudoku/Game.java
/** Return a string for the tile at the given coordinates */
protected String getTileString(int x, int y) {
    int v = getTile(x, y);
    if (v == 0)
        return "";
    else
        return String.valueOf(v);
}

```

一旦这些问题都解决了，就应该得到了一个可玩的数独游戏。试试看它能否运行？当然，正如任何代码都不可能完美无缺一样，这个程序还有可改进之处。

4.5 更多改进

对于数独游戏来说，虽然本章中这些代码的运行结果也可以接受，但是要想实现最优的设备性能，可能需要更加精心地设计更复杂的程序。尤其是onDraw()方法对程序的性能有着非常显著的影响，所以必须尽可能地降低它的这种影响。

下面是提高该方法执行效率的一些建议。

- 如果可能的话，避免在onDraw()方法中执行任何对象分配操作。
- 在onDraw()方法以外的地方（例如在视图的构造函数中）预取诸如颜色常量

这样的值。

- 预先创建自己的Paint类对象，在onDraw()方法中只使用已有的Paint类实例。
- 对于多次使用的值（如由getWidth()方法返回的宽度值），在方法的开始处将其提取出来，然后使用本地副本访问。

作为留给读者的进阶练习，建议大家考虑如何让数独游戏的图形效果更加绚丽多彩。例如，当玩家解决了某个谜题时，可以添加一些烟花效果，或者使单元格像Vanna White游戏那样旋转。游戏盘面的动态背景也会使程序更加有趣。尽情发挥你的想像力吧！如果想要设计出一流的程序，就应该像这样对程序的其他部分进行精雕细琢！

第5章将使用一些可突显氛围的音乐增强程序效果，第6章将介绍如何记住谜题状态并最终实现Continue按钮的功能。

4

6 快速阅读指南

本章只是蜻蜓点水式地介绍了Android的图形功能。Android的本机2D库非常庞大，因此，当你真正编写自己的程序时，一定要充分利用Android Eclipse插件提供的工具提示、自动完成功能和Javadoc功能。如果需要的话，包android.graphics^①的在线文档也能提供更多的细节。

如果你的程序需要更高级的图形功能，可以跳到第10章并阅读有关内容。第10章将提供如何使用基于OpenGL ES标准的Android三维图形库的信息。否则请转入下一章，开始了解Android精彩的音频和视频世界。

^① 参见网页<http://androidappdocs.appspot.com/reference/android/graphics/package-summary.html>。

还记得苹果公司电视广告中那些剪影人随着iPod播放的音乐节奏疯狂起舞吗？你可能希望自己的作品也能让人如此兴奋^①。与单纯的文字和图形相比，音乐、音效和视频可以使程序更让人喜欢，具有更大的吸引力。

本章将介绍如何在Android应用程序中实现多媒体功能。你可能暂时还无法让用户为你的呈现欢腾雀跃，但是恰当地运用多媒体特效至少可以让用户更开心。

1 播放音频

那是一个漆黑的暴风雨之夜……投篮已经开始了，他们的情况不妙……当国家队队员在最后一秒钟投中了一个三分球时，人们随即狂欢起来……

音频信号可以在环境中传播，影响我们的情感。可以将声音视作另一种影响用户情绪的手段。正如可以利用显示屏上的图形向用户传递某些信息一样，也可以使用音频支持并强化这种信息传递过程。

Android通过android.media包中的MediaPlayer类支持声音和音乐输出^②。下面我们通过一个简单的示例来体验一下Android的这种功能，让用户按下键盘或D-pad上的键时即开始播放声音。

首先创建一个“Hello, Android”项目，在New Android Project对话框中输入以下参数：

Project name: Audio

^①当然，8岁以上的正常人不会像那样疯狂地跳舞，除非我的孩子把一只蜥蜴放在我身体的某个部位……对不起，扯远了。

^②参见网页<http://androidappdocs.appspot.com/reference/android/media/package-summary.html>。

Package name: org.example.audio
 Activity name: Audio
 Application name: Audio

其次，我们需要播放一些声音。在这个示例中，我们使用Windows的录音机程序（在Windows XP中依次单击“开始”>“程序”>“附件”>“娱乐”>“录音机”）和便宜的耳机来录制自己的声音。将音量调到合适的大小以后，录下每一个声音，并从菜单上选择“文件”>“另存为……”，单击“更改……”按钮，选择Android可以识别的一种格式保存文件（参见图5-1）。本书的网站提供了这些示例的所有声音文件和源代码。

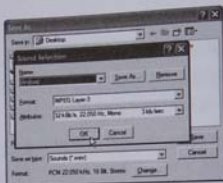


图5-1 以Android支持的压缩格式保存音效

将这些声音文件复制到Audio项目的res/raw目录下。回忆一下2.4节中讲过的内容，直接将一个文件复制到res目录下会导致Android Eclipse插件自动在R类中定义一个Java符号。复制声音文件后该项目的目录结构如图5-2所示。

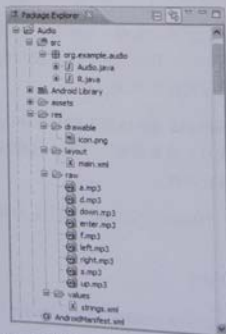


图5-2 将音频文件复制到Audio项目的res/raw目录中

小乔爱问……

Android支持哪些音频格式

有些音频格式Android在理论上支持；有些音频格式Android在模拟器上支持；还有些音频格式Android在实际设备上支持。理论上，Android支持以下文件类型（该类型清单可能会随着新版Android的发布而发生变化）：

- WAV（未压缩的PCM格式）；
- AAC（未受保护的苹果iPod格式）；
- MP3（MPEG-3）；
- WMA（Windows Media音频）；
- AMR（语音编解码器）；
- OGG（Ogg Vorbis）*；
- MIDI（乐器数字接口）。

实际上，只有OGG、WAV和MP3格式的音频可以在模拟器上正常播放，因此，推荐你使用这几种格式来开发应用程序。Android的本机音频格式采样率为44.1kHz，16位立体声。但是，由于使用该采样率的WAV格式文件过于庞大，因此应该尽量采用OGG或MP3文件格式（语音使用单声道，音乐使用立体声）。对于像游戏音效这种较短的音频剪辑来说，OGG格式似乎性能最佳。

不要使用很少见的采样率（如8kHz），因为在对声音重新采样时，这些采样率会导致声音非常刺耳。使用11kHz、22kHz或44.1kHz的采样率可获得最佳的效果。记住，尽管手机有一个微型扬声器，但很多用户还是会插入耳机（就像使用iPod一样）。因此，开发人员应该提供高品质的音频。

* 参见网页<http://www.vorbis.com>。

下面该定义Audio活动了。首先为每个声音声明一个MediaPlayer类的实例，并在onCreate()方法中初始化这些实例。

```
Audio/src/org/example/audio/Audio.java
```

```
package org.example.audio;

import android.app.Activity;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.view.KeyEvent;
```

```

public class Audio extends Activity {
    private MediaPlayer up, down, left, right, enter;
    private MediaPlayer a, s, d, f;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        // Native rate is 44.1kHz 16 bit stereo, but
        // to save space we just use MPEG-3 22kHz mono
        up = MediaPlayer.create(this, R.raw.up);
        down = MediaPlayer.create(this, R.raw.down);
        left = MediaPlayer.create(this, R.raw.left);
        right = MediaPlayer.create(this, R.raw.right);
        enter = MediaPlayer.create(this, R.raw.enter);
        a = MediaPlayer.create(this, R.raw.a);
        s = MediaPlayer.create(this, R.raw.s);
        d = MediaPlayer.create(this, R.raw.d);
        f = MediaPlayer.create(this, R.raw.f);
    }
}

```

也可以使用其他方法来播放声音，例如，可以只声明一个MediaPlayer类的实例并反复重用该实例。但是，这种方法无法实现声音重叠，而这可能是，也可能不是你想要的效果。

另一种你可能很想尝试的方法是，每次要播放声音时就创建一个新的MediaPlayer实例。然而，这种方法实际上不能正常工作。首先，该方法会在一定程度上降低程序的运行效率。更糟糕的是，在测试过程中仅播放几个声音后该方法就出现了崩溃的趋势。因此，建议尽量使用切实可靠的方法提前创建自己的播放器，并在需要时播放声音。

既然已经加载了我们自己的声音并准备就绪，接下来只需截获按键事件并播放适当的聲音。可以通过重写Activity.onKeyDown()方法来实现上述操作。

```

Audio/sec/org/example/audio/Audio.java
line 1  @Override
- public boolean onKeyDown(int keyCode, KeyEvent event) {
-     MediaPlayer mp;
-     switch (keyCode) {
5      case KeyEvent.KEYCODE_DPAD_UP:
-         mp = up;
-         break;
-     case KeyEvent.KEYCODE_DPAD_DOWN:
-         mp = down;
10     break;
-     case KeyEvent.KEYCODE_DPAD_LEFT:
-         mp = left;

```

```

        break;
    case KeyEvent.KEYCODE_DPAD_RIGHT:
15      mp = right;
        break;
    case KeyEvent.KEYCODE_DPAD_CENTER:
    case KeyEvent.KEYCODE_ENTER:
        mp = enter;
20      break;
    case KeyEvent.KEYCODE_A:
        mp = a;
        break;
    case KeyEvent.KEYCODE_S:
25      mp = s;
        break;
    case KeyEvent.KEYCODE_D:
        mp = d;
        break;
30      case KeyEvent.KEYCODE_F:
        mp = f;
        break;
    default:
        return super.onKeyDown(keyCode, event);
35  }
    mp.seekTo(0);
    mp.start();
    return true;
}

```

该方法的第一部分根据用户按下的键选择一个播放器。接下来在第36行调用 `seekTo()` 方法，以回退到相应声音文件的起始位置，在第37行调用 `start()` 方法开始播放该声音。`start()` 方法是异步播放声音的，因此无论该声音文件的持续时间有多长，在播放完毕后都可以立即返回。如果愿意，可以使用 `setOnCompletionListener()` 方法在音频剪辑播放结束时通知你。

出现问题后怎么办

如果熟悉多媒体程序设计，你很快就会发现Android的MediaPlayer类难以驾驭。它动不动就崩溃，例如不按顺序调用其方法，或者向其传递一种格式无法识别的文件等。发生这种情况的原因之一是MediaPlayer类主要是一种本机应用，它使用的是位于其上的一个简化的Java层。本机播放器代码主要是针对性能进行优化的，没有过多地考虑错误检查。

幸运的是，Android强健的Linux进程保护机制防止了可能的崩溃对其他应用程序产生任何影响。模拟器（如果在实际设备上运行程序，也可是手机）和其他应用程序都会继续正常运行。用户所看到的只是崩溃的应用程序终止了，可能还会出现其中包含错误消息的对话框。

但是，可以在开发期间获得详细的诊断信息，帮助你确定错误发生的原因。各种消息和回溯（traceback）信息将被输出到Android的系统日志中，可以通过Eclipse插件中的LogCat视图或执行adb logcat命令查看这些信息（参见3.10节）。

如果此时运行该程序并按下某个键（如回车键或D-pad的中心按钮），应该能听到声音。如果没听到声音，请检查音量控制（不要笑，问题有可能出在这里），或者在LogCat视图中查看调试消息^①。

下一节将实现只用一行代码即可播放一段电影剪辑。

5.2 播放视频

视频不只是连续显示出来的一大堆图片。它还包括声音，并且还要让声音与图像完全同步。



小乔爱问……

Android支持哪些视频格式

正如对音频格式的支持一样，Android SDK的当前版本对视频格式的支持仍然缺乏一致性，它正式支持的格式如下：

- MP4（MPEG-4低比特率）；
- H.263；
- H.264（AVC）。

在Windows计算机上，唯一能够可靠工作的格式是MP4，因此建议你尽量采用这种格式。可以使用像QuickTime Pro^{*}这样的程序将视频从一种格式转换为另一种格式。尽可能使用最低分辨率和低比特率可节省存储空间，但也不要太低，以免降低视频的质量。

^{*} 参见网页<http://www.apple.com/quicktime/pro>。

① 有些情况下音频输出可能会出现声音断断续续或延迟的现象。可以试用不同格式的文件（如OGG而不是MP3）或者降低比特率。你可能还需要研究一下SoundPool类的用法，SoundPool类的文档明确表示它支持同时播放多个流。可惜的是在1.0版本中，该类还存在较多的bug，而且文档也不够全面。

Android的MediaPlayer类的视频处理方式与普通音频的处理方式一样。唯一的差别是：处理视频时需要为播放器创建一个用于绘制图像的Surface。可使用start()和stop()这两个方法控制视频的播放。

不过，本书不准备介绍另一个MediaPlayer示例，因为还有一种更简单的方法可以将视频嵌入到应用程序中：VideoView类。为了演示该类的用法，我们使用下面的参数创建一个名为Video的Android新项目：

```
Project name: Video
Package name: org.example.video
Activity name: Video
Application name: Video
```

将布局（res/layout/main.xml）改为以下内容：

```
Video\res\layout\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">
    <VideoView
        android:id="@+id/video"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:layout_gravity="center" />
</FrameLayout>
```

打开Video.java文件，将onCreate()方法改为以下内容：

```
Video\src\org\example\video\Video.java
package org.example.video;

import android.app.Activity;
import android.os.Bundle;
import android.widget.VideoView;

public class Video extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Fill view from resource
        setContentView(R.layout.main);
        VideoView video = (VideoView) findViewById(R.id.video);

        // Load and start the movie
        video.setVideoPath("/data/samplevideo.mp4");
    }
}
```

`setVideoPath()`方法首先打开视频文件,在保持其宽高比不变的情况下,根据其所在的容器调整播放画面的大小,然后播放该视频。

现在需要上传视频文件并播放。为此,可运行下面的命令:

```
C:\> adb push c:\code\samplevideo.mp4 /data/samplevideo.mp4
377 KB/s (0 bytes in 824192.002s)
```

可以在本书的下载包中找到`samplevideo.mp4`文件,你也可以自己创建一个视频文件。此处使用的目录(`/data`)只是用于说明目的,不应真正用于存放媒体文件。

注意,Android实际上并不关心文件的扩展名。可以在Eclipse中使用Android透视图中的File Explorer视图上传或下载文件,但是对于像上面这种简单的操作来说,我发现使用命令行更容易一些。

还有一个问题,我们通常希望视频画面能够占满整个屏幕,包括播放器的标题栏和状态栏所占的部分。为此,只需在`AndroidManifest.xml`文件中指定正确的主题即可:

Video01/AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.example.video"
    android:versionCode="1"
    android:versionName="1.0.0">
    <application android:icon="@drawable/icon"
        android:label="@string/app_name">
        <activity android:name=".Video"
            android:label="@string/app_name"
            android:theme="@android:style/Theme.NoTitleBar.Fullscreen">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

一旦完成了上述所有工作,运行该程序就可以欣赏电影剪辑了(参见图5-3)。试着旋转一下屏幕,看看程序在纵向和横向模式下是否都能正常运行。瞧,视频真的可以播放了!



图5-3 利用VideoView类可以轻松地将视频嵌入到应用程序中

下面继续改进Sudoku示例程序，为其配上一点背景音乐。

11// 小乔爱问……

为什么在我旋转屏幕后程序重新播放视频

默认情况下，Android假定程序对于屏幕旋转一无所知。为了处理可能的资源变化，Android销毁并从头重建你的活动。这意味着要再次调用onCreate()方法，也意味着要重新开始播放视频（就像当前正在编写的示例程序的处理方式一样）。

这种处理方式适用于90%的应用程序，因此大多数开发人员不必关心这一问题。对于测试应用程序的生命周期和状态保存/恢复代码（参见2.2节），这甚至是一种有用的方法。但是，还有一些更好的办法能优化视频的转换。

最简单的办法是在活动中实现onRetainNonConfigurationInstance()方法，以保存一些在对onDestroy()和onCreate()方法的多次调用中需要保持的数据。在返回时，你可以在活动的新实例中使用getLastNonConfigurationInstance()方法恢复这些信息。可以保存任何信息，甚至包括对当前意图的引用和正在运行的线程。

更复杂的方法是在AndroidManifest.xml文件中使用android:configChanges = 特性这种形式将你可以处理的变化通知Android。例如，如果将特性设为keyboardHidden|orientation，那么当用户翻转键盘时，Android将不会销毁和创建活动。相反，Android将调用onConfigurationChanged(Configuration)方法，并假定你自己知道如何处理。

* 更多详细信息，请参阅<http://androidappdoc.appspot.com/reference/android/app/Activity.html#ConfigurationChanges>。

数独轶事

现在的数独游戏变体有数十种,但是没有一种能够像原始数独游戏那样流行。还有一种称作Gattai 5或有一种数独变体使用16×16的网格,用16进制数玩游戏。还有一种称作Gattai 5或武士数独(Samurai Sudoku)的数独变体,由5个9×9的网格构成,每个的角部区域都是重叠的。

5.3 为数独游戏配上音乐

本节将运用我们前面所学的知识为已经建立的数独游戏配上背景音乐。我们将在程序处于启动画面阶段时播放一段音乐,在实际玩游戏的过程中播放另一段音乐。这样既说明了如何播放音乐,还说明了一些重要的生命周期事项。

要给主屏幕配乐,只需重写Sudoku类中下面的这两个方法:

```
SudokuV3/src/org/example/sudoku/Sudoku.java

@Override
protected void onResume() {
    super.onResume();
    Music.play(this, R.raw.main);
}

@Override
protected void onPause() {
    super.onPause();
    Music.stop(this);
}
```

还记得2.2节的内容吗,当活动准备就绪,可以开始与用户进行交互时,就会调用onResume()方法。此处是开始播放音乐的一个最佳时机,因此我们在这里调用Music.start()方法。Music类将在随后定义。

R.raw.main引用res/raw/main.mp3文件。可以在本书网站上的下载包的SudokuV3项目中找到这些声音文件。

onPause()方法与onResume()方法是配对出现的。当onPause()方法被调用时,Android将在恢复一个新活动之前暂停当前活动。因此在数独游戏中,开始一个新游戏时,Sudoku活动将被暂停,然后Game活动被启动。用户按下Back键或Home键时,onPause()方法也会被调用。这些都是我们停止片头音乐的最佳时机,因此我们在onPause()方法中调用Music.stop()方法。



小乔爱问……

为什么不使用后台服务播放音乐

本书没有深入讨论Android的Service类，但是你可能在网络上看到有些音乐播放示例程序中使用了该类。从根本上来说，Service类就是一种启动后台进程（一种Linux守护进程）的方式。即使当前活动结束了，这种后台进程仍保持运行状态。如果你要编写一个通用的音乐播放器，想在用户阅读邮件或浏览网页时继续播放音乐，那么使用Service类比较合适。然而在大多数情况下，你希望程序结束时停止播放音乐，因此不需要使用Service类。

下面对Game活动进行同样的修改，以控制音乐的播放：

SudokuV3/src/org/example/sudoku/Game.java

```
@Override
protected void onResume() {
    super.onResume();
    Music.play(this, R.raw.game);
}
```

```
@Override
protected void onPause() {
    super.onPause();
    Music.stop(this);
}
```

如果比较一下对Game类和Sudoku类的修改，你会注意到我们在Game类中引用了另一个声音资源，R.raw.game（res/raw/game.mp3）。

最后一部分是Music类的定义，Music类将管理用来播放当前音乐的MediaPlayer类：

SudokuV3/src/org/example/sudoku/Music.java

```
Line 1 package org.example.sudoku;
-
- import android.content.Context;
- import android.media.MediaPlayer;
5
- public class Music {
-     private static MediaPlayer mp = null;
-
-     /** Stop old song and start new one */
10     public static void play(Context context, int resource) {
-         stop(context);
-         mp = MediaPlayer.create(context, resource);
-         mp.setLooping(true);
-         mp.start();
-
```

```

15     }
-
-     /** Stop the music */
-     public static void stop(Context context) {
-         if (mp != null) {
20             mp.stop();
-             mp.release();
-             mp = null;
-         }
-     }
25 }

```

`play()`方法首先调用`stop()`方法，停止一切正在播放的音乐。然后调用`MediaPlayer.create()`方法创建一个新的`MediaPlayer`实例，将上下文（context）和资源ID传递给该实例。

创建完播放器以后，设置一个选项可让它循环播放音乐。然后立即调用`start()`方法开始播放音乐。

从第18行开始的`stop()`方法很简单。首先是简单的保护性检查，确保要处理的`MediaPlayer`实例确实存在，然后调用该实例的`stop()`和`release()`方法。`MediaPlayer.stop()`方法停止音乐的播放（很奇怪）。`release()`方法释放与播放器有关的系统资源。由于这些资源都是本机资源，因此不能等待正常的Java垃圾回收机制来回收它们。漏掉`release()`方法可能会导致程序意外运行失败（当然并不是说这样一定会导致程序运行失败，只是提醒你你应该记住这一点）。

现在是程序中最有趣的部分——试着玩一下改进后的数独游戏。用你能够想象到的各种方式对其进行压力测试，例如切换到不同的活动，在游戏的不同时刻按下Back按钮和Home按钮，当程序已经运行时再次启动该程序，等等。正确地管理生命周期有时确实比较麻烦，但用户将会感谢你的这些努力。

5.4 快速阅读指南

本章介绍了如何使用Android SDK播放音频和视频剪辑。我们没有讨论如何录音和录像，因为大多数程序不需要这些功能，但如果你正好属于例外情况，请查阅在线文档中关于`MediaRecorder`类的内容^①。

第6章将介绍Android程序在多个调用之间存储数据的一些简单方法。如果不需要这些功能，可以直接跳至第7章，学习有关网络访问的知识。

① 参见网页<http://androidappdocs.appspot.com/reference/android/media/MediaRecorder.html>。

迄今为止，我们主要专注于编写在退出时无需保存数据的应用程序。这些程序悄然启动、运行和终止，没有留下任何它们曾经存在过的痕迹。然而，大多数的实际程序需要保留一些持久性状态，这可能是简单的字号设置，上次办公室舞会上一张令人尴尬的照片，也可能是下周的饮食计划。无论这些状态是什么，Android支持开发人员将其永久地存储在移动设备上供以后使用，同时保护其不被其他程序意外或恶意地访问。

根据数据的大小、数据结构、生存时间以及是否与其他程序共享该数据，应用程序可以使用多种不同的技术存储数据。本章将介绍3种简单的本地数据存储办法：首选项API、实例状态绑定和闪存文件。第9章将深入介绍更先进的技术：使用内置的SQLite数据库引擎。

1 为数独游戏添加选项

在3.7节中，我们使用`onCreateOptionsMenu()`方法在数独游戏主屏幕上创建了一个菜单项的菜单。用户按下Menu键并选择Settings...菜单项时，代码将启动Settings活动，该活动允许用户修改游戏选项。因为Settings类继承自PreferenceActivity类，所以这些设置的值被存储在程序的首选项位置，但是当初并没有给出相应的代码。下面来实现首选项功能。

数独轶事

标准数独一共有6 670 903 752 021 072 936 960种可能的终盘。如果不计算重复（只是对每个终盘进行数字交换，对称变换或重标号），那么“只”有5 472 730 538个终盘。

首先修改Settings类，向其添加两个取值器（getter）方法，以提取两个选项的当前值。下面是新的定义内容：

```
SudokuV4/src/org/example/sudoku/Settings.java
package org.example.sudoku;

import android.content.Context;
import android.os.Bundle;
import android.preference.PreferenceActivity;
import android.preference.PreferenceManager;

public class Settings extends PreferenceActivity {
    // Option names and default values
    private static final String OPT_MUSIC = "music";
    private static final boolean OPT_MUSIC_DEF = true;
    private static final String OPT_HINTS = "hints";
    private static final boolean OPT_HINTS_DEF = true;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.settings);
    }

    /** Get the current value of the music option */
    public static boolean getMusic(Context context) {
        return PreferenceManager.getDefaultSharedPreferences(context)
            .getBoolean(OPT_MUSIC, OPT_MUSIC_DEF);
    }

    /** Get the current value of the hints option */
    public static boolean getHints(Context context) {
        return PreferenceManager.getDefaultSharedPreferences(context)
            .getBoolean(OPT_HINTS, OPT_HINTS_DEF);
    }
}
```

小心，选项键（music和hints）要与res/xml/settings.xml文件中使用的键匹配。

必须修改Music.play()方法以检查音乐首选项：

```
SudokuV4/src/org/example/sudoku/Music.java

/** Stop old song and start new one */
public static void play(Context context, int resource) {
    stop(context);

    // Start music only if not disabled in preferences
    if (Settings.getMusic(context)) {
        mp = MediaPlayer.create(context, resource);
        mp.setLooping(true);
        mp.start();
    }
}
```


还要修改PuzzleView.onDraw()方法以检查提示(hints)首选项:

```
SudokuV4/src/org/example/sudoku/PuzzleView.java
if (Settings.getHints(getContext())) {
    // Draw the hints...
}
```

如果getHints()方法返回“真”，则高亮显示提示(参见图4-6)；否则只是跳过该部分。

下面介绍如何使用首选项API存储其他数据，而不仅仅是选项。

继续玩前一个游戏

任何时候玩家都可以退出数独游戏，转而去做其他事情。例如老板突然闯了进来，或者接了一个电话或收到一份重要的任命通知。不管原因是什么，我们都希望玩家返回后能继续玩刚才的那一盘游戏。

首先，需要在某个地方保存游戏的当前状态。除了可以用于存储选项以外，首选项API还可以存储程序运行过程中使用的任何少量独立信息块。在本例中，游戏的状态可以保存为由81个字符组成的字符串，每个单元格用一个字符表示。

在Game类中，我们首先定义两个常量：一个表示游戏的数据键，另一个是标志，说明要继续玩前一个游戏而不是开始一个新游戏。

```
SudokuV4/src/org/example/sudoku/Game.java
private static final String PREF_PUZZLE = "puzzle";
protected static final int DIFFICULTY_CONTINUE = -1;
```

然后，每当游戏被暂停时，都需要保存游戏的当前状态。关于onPause()方法和其他生命周期方法的内容，参见2.2节。

```
SudokuV4/src/org/example/sudoku/Game.java
@Override
protected void onPause() {
    super.onPause();
    Log.d(TAG, "onPause");
    Music.stop(this);

    // Save the current puzzle
    getPreferences(MODE_PRIVATE).edit().putString(PREF_PUZZLE,
        toPuzzleString(puzzle)).commit();
}
```

现在已经保存了游戏，但是如何读取已保存的数据呢？还记得吗，游戏启动时会调用getPuzzle()方法，并将难度级别传递给该方法。我们也使用该方法来继续玩前一个游戏。

```
SudokuV4/src/org/example/sudoku/Game.java
/* Given a difficulty level, come up with a new puzzle */
private int[] getPuzzle(int diff) {
    String puz;
    switch (diff) {
        case DIFFICULTY_CONTINUE:
            puz = getPreferences(MODE_PRIVATE).getString(PREF_PUZZLE,
                easyPuzzle);
            break;
            // ...
    }
    return fromPuzzleString(puz);
}
```

只需添加一个检查DIFFICULTY_CONTINUE的case分支。如果其值为“真”，则读取首选项中存储的前一个游戏，而不是开始一个新游戏。

接下来，我们需要让主屏幕上的Continue按钮（参见图3-4）具有真正的作用。为此，对onClick()方法进行如下修改：

```
SudokuV4/src/org/example/sudoku/Sudoku.java
public void onClick(View v) {
    switch (v.getId()) {
        case R.id.continue_button:
            startGame(Game.DIFFICULTY_CONTINUE);
            break;
            // ...
    }
}
```

我们在Sudoku.onClick()方法中增加一个case分支，用户按下Continue按钮时调用startGame()方法，并将DIFFICULTY_CONTINUE传递给startGame()方法。startGame()方法再将难度级别传递给Game活动，而Game.onCreate()方法又调用Intent.getIntExtra()方法以读取该难度级别，并将其传递给getPuzzle()方法（关于这些方法调用的代码，参见4.2节）。

还有一个问题：当一个活动终止，然后又返回到该活动时如何恢复已保存的游戏（例如另一个活动被启动，然后用户又返回到Game活动）。下面对Game.onCreate()方法的修改可以解决这个问题：

SudokuV4/src/org/example/sudoku/Game.java

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    // ...
    // If the activity is restarted, do a continue next time
    getIntent().putExtra(KEY_DIFFICULTY, DIFFICULTY_CONTINUE);
}
```

首选项API的内容大致介绍完了。下面介绍一下如何保存实例状态。

记住当前位置

如果Sudoku程序正在运行时改变了屏幕方向，你会发现程序忘记了其光标的位置。这是因为我们使用了自定义的PuzzleView视图。标准的Android视图会自动保存其视图状态，由于我们使用的是自定义视图，所以无法自动保存状态。

与持久状态不同，实例状态是暂时性的，它存储在Android应用程序栈的Bundle类中。实例状态设计用于较小的信息块，如光标位置。

下面是我们要实现的内容：

SudokuV4/src/org/example/sudoku/PuzzleView.java

```
Line 1 private static final String SELX = "selX";
- private static final String SELY = "selY";
- private static final String VIEW_STATE = "viewState";
- private static final int ID = 42;
5
- public PuzzleView(Context context) {
-     // ...
-     setId(ID);
- }
10
- @Override
- protected Parcelable onSaveInstanceState() {
-     Parcelable p = super.onSaveInstanceState();
-     Log.d(TAG, "onSaveInstanceState");
15     Bundle bundle = new Bundle();
-     bundle.putInt(SELX, selX);
-     bundle.putInt(SELY, selY);
-     bundle.putParcelable(VIEW_STATE, p);
-     return bundle;
20 }
- @Override
- protected void onRestoreInstanceState(Parcelable state) {
-     Log.d(TAG, "onRestoreInstanceState");
-     Bundle bundle = (Bundle) state;
25     select(bundle.getInt(SELX), bundle.getInt(SELY));
-     super.onRestoreInstanceState(bundle.getParcelable(VIEW_STATE));
-     return;
- }
```

从第1行开始我们定义了几个键的常量，用于保存和恢复光标位置。我们需要保存x和y两个方向的位置，以及基础View类所需的任何状态。

作为Activity.onSaveInstanceState()处理过程的一部分，Android将遍历视图层次结构中的每个视图，每当发现一个具有ID的视图，就调用View.onSaveInstanceState()方法。onRestoreInstanceState()方法的调用规则与View.onSaveInstanceState()方法相同。通常情况下，视图ID应该来自XML文件，但由于PuzzleView视图是用代码创建的，所以我们需要自己设置该ID值。第4行为ID随意指定了一个数字（任何正整数都是有效的），第8行调用setId()方法将该值分配给视图。

第12行定义了onSaveInstanceState()方法。这里调用PuzzleView类的父类（View类）的onSaveInstanceState()方法，以获取其父类的视图状态，然后将PuzzleView类的视图状态与其父类的视图状态保存在一个Bundle类实例中。如果调用父类的onSaveInstanceState()方法失败，则返回一个运行时错误。

随后调用onRestoreInstanceState()方法（第22行），以恢复已保存的信息。我们可以从Bundle类的实例中提取PuzzleView视图的x和y坐标，然后调用其父类的onRestoreInstanceState()方法使其父类的视图获得所需的信息。完成上述修改以后，PuzzleView就可以记住光标的位置了，就像任何其他Android视图一样。

下面我们来看看如何在普通的老式文件中保存数据。

6.4 访问内部文件系统

因为Android运行在Linux之上，所以系统中挂载了一个真实的文件系统，包含根目录和其他各种文件目录。这些文件存储在设备内置的非易失性闪存中，因此手机关机后这些文件也不会丢失。

你的程序可以使用java.io包中的所有常规Java文件I/O例程，这些例程带有表明进程具有有限权限的标志，因此程序不能破坏其他任何应用程序的数据。事实上，你的程序能够访问的内容主要是在安装时创建的包私有目录（/data/data/包名）。

Context类提供了一些帮助器方法（每一个活动的父类Activity继承自Context类，因而也可以调用这些方法），允许开发人员从文件系统读取数据并向文件系统

写入数据。下面给出了开发人员最常用的几个方法。

`deleteFile()`: 删除一个私有文件。如果操作成功则返回`true`, 否则返回`false`。

`fileList()`: 以`String`数组的形式返回该应用程序私有目录中所有文件的列表。

`openFileInput()`: 打开一个用于读取的私有文件, 其返回值为`java.io.FileInputStream`。

`openFileOutput()`: 打开一个用于写入的私有文件, 其返回值为`java.io.FileOutputStream`。

但是, 由于内部闪存容量有限, 建议只在其中保存较少的数据, 例如最多1MB~2MB, 而且在程序设计时要小心处理I/O错误, 以免闪存空间耗尽。

幸运的是, 内部闪存并不是开发人员唯一可用的存储器。

数据共享

回忆一下2.5节的内容, 每个应用程序通常在安装时会获得自己的用户ID。该用户ID是允许对此应用程序的私有目录进行读写的唯一标识。但是, 如果两个应用程序由同一个数字证书签名*, 则Android认为这两个应用程序属于同一个开发人员并为其分配相同的用户ID。

如果两个应用程序具有相同的用户ID, Android的这种处理方式一方面使它们能够互相共享各种数据。但另一方面也意味着你需要格外小心地应对, 以免它们互相影响。

* 参见网页<http://androidappdocs.appspot.com/guide/topics/security/security.html>。

5 访问 SD 卡

有些Android设备会带有一个插槽, 可插入额外的闪存卡, 通常是安全数字卡(Secure Digital, SD)。这些存储卡(如果有的话)的容量远远大于内置的闪存容量, 因此是存储较大的音频和视频文件的理想介质。SD卡不能用于存放代码, 每个应用程序都可以从SD卡上读取文件, 也可以向SD卡上写入文件。

在5.2节中, 我们向模拟设备的/`data`目录中上载了一个示例视频文件。这样做实际上是错误的, 因为不应该将过大的文件放入内部文件系统。因此, 我们现在展示一种更好的方法。

首先, 创建并格式化一个可以“插入”到模拟器中的虚拟SD卡。可以使用 **mkshcard** 实用工具来创建虚拟SD卡^①:

```
C:\> mkshcard 256M c:\temp\sd.img
C:\> dir c:\temp\sd.img
Volume in drive C is ANDROID_RULES
Volume Serial Number is 5432-ABCD

Directory of c:\temp

11/17/2008  08:58 PM          268,435,456 sd.img
               1 File(s)      268,435,456 bytes
               0 Dir(s)      4,369,543,168 bytes free
```

该命令在开发用的计算机上创建一个容量为256MB的虚拟SD卡。实际上, 虚拟SD卡的容量可以设置为任意大小, 但是如果容量设得太小, 可能会导致模拟器崩溃; 如果容量设得太大, 则又会浪费计算机硬盘驱动器上的空间。

其次, 为虚拟SD卡创建映像文件后, 还需要通知Android模拟器到什么地方查找该映像文件。与实际手机不同的是, 这里需要重新启动模拟设备以“插入”虚拟SD卡。如果模拟器窗口处于打开状态, 请先关闭该窗口, 然后编辑Video项目的Eclipse启动配置(在Eclipse 3.4中依次单击Run > Run Configurations...), 添加下面的选项(参见图6-1), 然后选择Run按钮:

```
-sdcard c:\temp\sd.img
```

如果没有使用Eclipse, 则需要手工将该选项添加到模拟器的命令行中。

再次运行模拟器, 将示例视频文件复制到虚拟SD卡:

```
C:\> adb push c:\code\samplevideo.mp4 /sdcard/samplevideo.mp4
140 KB/s (0 bytes in 824192.005s)
```

然后需要修改Video类的onCreate()方法, 以便播放虚拟SD卡上而不是/data目录中的视频:

```
VideoView2/src/org/example/video/Video.java
// Load and start the movie
video.setVideoPath("/sdcard/samplevideo.mp4");
```

现在尝试运行一下该程序。应该可以正常播放视频了。

^① 参见网页<http://androidappdocs.appspot.com/guide/developing/tools/othertools.html#mkshcard>。

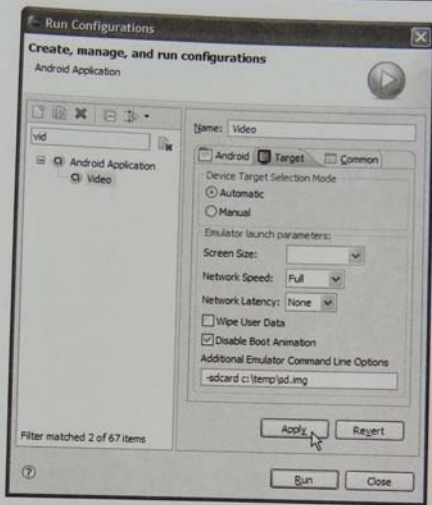


图6-1 在模拟器选项中指定SD卡映像文件的路径名称

6 快速阅读指南

本章介绍了在Android平台上存储本地数据的几种基本方法。这些方法对于简单的应用来说足够用了，但要处理结构化数据（如电话清单和食谱），还需要一些更高级的技术。第9章将介绍如何使用Android内置的SQLite数据库以及如何利用内容提供者 in 应用程序之间共享信息。

本书第二部分的内容到这里就结束了。借助Sudoku示例程序，我们已经学习了Android程序设计的所有基础知识，其中包括用户界面、2D图形、音频、视频以及简单的数据存储。

下面我们将抛开Sudoku示例程序，开始深入学习Android程序设计方面的一些内容。

Part 3

第三部分

高级主题

本 部 分 内 容

- 第7章 互联的世界
- 第8章 定位与环境感知
- 第9章 SQL实战
- 第10章 利用OpenGL实现3D图形

接下来的几章将介绍更多高级主题，例如网络访问和基于位置的服务。你可以编写很多没有这些功能的有用应用程序，但是了解Android的这些高级特性有助于你实现程序的增值，以最少的代价为其赋予更多的功能。

你通常用手机来做什么？除了打电话，越来越多的人将手机用作移动因特网设备。分析师预计在未来几年中，手机将超过台式机成为第一大因特网连接方式^①。在这个世界的某些地方已经出现了这种情况^②。

Android手机已经为新的移动因特网互联世界做好了准备。首先，Android提供了基于WebKit开源项目的全功能Web浏览器^③。它使用的引擎与谷歌公司的Chrome、苹果公司的iPhone和Safari桌面浏览器中的引擎相同，只是进行了很少的修改。通过Android，你可以将浏览器用作应用程序内部的一个组件。

其次，通过Android可让程序访问标准网络服务，如TCP/IP套接字。这样就可以使用谷歌、雅虎、亚马逊提供的Web服务和因特网上的很多其他资源。

本章将通过4个示例程序介绍如何充分利用所有这些特性以及更多的功能。

- **BrowserIntent**：演示如何使用Android中的意图（intent）打开外部Web浏览器。
- **BrowserView**：展示如何将浏览器直接嵌入应用程序。
- **LocalBrowser**：解释嵌入式WebView中的JavaScript和Android程序中的Java代码如何彼此通信。

^① 参见网页<http://www.idc.com/getdoc.jsp?containerid=prUS21303808>。

^② 参见网页<http://www.comscore.com/press/release.asp?press=1742>。

^③ 参见网页<http://webkit.org>。

□ Translate: 将数据绑定、线程和Web服务用于娱乐目的。

7.1 通过意图实现浏览

使用Android的网络API能完成的最简单事情,就是利用浏览器打开所选的网页。你可能想用这种方法在程序中提供主页的链接,或者访问某些基于服务器的应用程序,如订单系统。在Android中,只要3行代码即可完成这个任务。

我们编写一个称为BrowserIntent的新示例程序来演示这个过程。该程序中有一个编辑字段,你可以在其中输入URL,还有一个Go按钮,单击该按钮就会用浏览器打开该URL处的内容(参见图7-1)。



图7-1 使用Android意图打开浏览器

首先,在New Project向导中使用以下值创建新的“Hello, Android”项目:

```
Project name: BrowserIntent
Package name: org.example.browserintent
Activity name: BrowserIntent
Application name: BrowserIntent
```

创建基本程序之后,更改布局文件(res/layout/main.xml),更改后的内容如下所示:

```
BrowserIntent/res/layout/main.xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <EditText
        android:id="@+id/url_field"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1.0"
```

```

        android:lines="1" />
    <Button
        android:id="@+id/go_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/go_button" />
</LinearLayout>

```

上面的代码定义了两个控件：一个EditText控件和一个Button。

在EditText中设置了`android:layout_weight="1.0"`，以便文本区域填满按钮左侧的水平空间，还设置了`android:lines="1"`，将控件高度限制为一个垂直行的高度。注意，这对用户在此处可输入的文本数量没有影响，只是设置其显示方式。

同往常一样，让用户看到的文本应该放到资源文件`res/values/strings.xml`中：

```

BrowserIntent/res/values/strings.xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">BrowserIntent</string>
    <string name="go_button">Go</string>
</resources>

```

下面需要编写BrowserIntent类的onCreate()方法。这是要构建用户界面并连接所有行为的地方。如果不想输入这些代码，可以从网上获取本示例的完整源代码。

```

BrowserIntent/src/org/example/browserintent/BrowserIntent.java

```

```

Line 1 package org.example.browserintent;
-
- import android.app.Activity;
- import android.content.Intent;
5 import android.net.Uri;
- import android.os.Bundle;
- import android.view.KeyEvent;
- import android.view.View;
- import android.view.View.OnClickListener;
10 import android.view.View.OnKeyListener;
- import android.widget.Button;
- import android.widget.EditText;
-
- public class BrowserIntent extends Activity {
15     private EditText urlText;
-     private Button goButton;
-
-     @Override
-     public void onCreate(Bundle savedInstanceState) {
20         super.onCreate(savedInstanceState);
-         setContentView(R.layout.main);
-
-         // Get a handle to all user interface elements

```

```

urlText = (EditText) findViewById(R.id.url_field);
goButton = (Button) findViewById(R.id.go_button);

// Setup event handlers
goButton.setOnClickListener(new OnClickListener() {
    public void onClick(View view) {
        openBrowser();
    }
});
urlText.setOnKeyListener(new OnKeyListener() {
    public boolean onKey(View view, int keyCode, KeyEvent event) {
        if (keyCode == KeyEvent.KEYCODE_ENTER) {
            openBrowser();
            return true;
        }
        return false;
    }
});
}
}

```

在`onCreate()`方法内，我们在第21行调用了`setContentView()`方法，根据布局资源中的视图定义来加载视图，然后在第24行调用`findViewById()`方法，获得两个用户界面控件的句柄。

第28行告诉Android，在用户通过触摸屏按下Go按钮，或在该按钮上按下D-pad的中央按钮从而选中该按钮时运行一些代码。该按钮被选中时会调用`openBrowser()`方法，稍后会定义该方法。

为了方便，如果用户输入地址并按Enter键（如果他们的手机上有这个键），我们希望像单击Go按钮一样打开浏览器。为此从33行开始定义一个监听器，用户每次向编辑字段中输入内容时都会调用这个监听器。如果用户按了Enter键，就调用`openBrowser()`方法打开浏览器；否则返回`false`，让文本控件正常处理用户的输入。

下面就是你等待已久的部分：`openBrowser()`方法。如上文所言，它只有3行：

```

BrowserIntent/src/org/example/browserintent/BrowserIntent.java
/** Open a browser on the URL specified in the text box */
private void openBrowser() {
    Uri uri = Uri.parse(urlText.getText().toString());
    Intent intent = new Intent(Intent.ACTION_VIEW, uri);
    startActivity(intent);
}

```

第一行以字符串形式获取网址（例如`http://www.android.com`），并将其转换为

URI (uniform resource identifier, 统一资源标识符)。下一行创建了新的Intent类, 带有操作VIEW_ACTION, 将要查看的、刚刚作为对象创建的Uri类传递给它。最后, 调用startActivity()方法请求执行该操作。

Browser活动启动后, 它将创建自己的视图(参见图7-2)并且程序将被暂停。如果用户此时按下Back键, 浏览器窗口将消失, 应用程序将继续运行。如果想同时查看用户界面上的内容和网页又该怎么办呢? 在Android中可以使用WebView类做到这一点。



图7-2 用默认浏览器查看网页

2 利用视图打开网页

在台式机上, 具有书签、插件、Flash动画、选项卡、滚动条、打印等功能的Web浏览器是个很大的、复杂的且要耗用大量内存的程序。

我以前做Eclipse项目时, 有人建议用嵌入式Web浏览器替代一些常见的文本视图, 我觉得这种想法很疯狂。我当时反驳道, 只要增强文本查看器, 使它显示斜体

或表格或者做一些它原来不能做的事情，不是更有意义吗？

现在看来他们的想法并不疯狂，因为：

- Web浏览器可以（相对）简洁一些，这意味着可去掉基本呈现引擎之外的所有东西；
- 如果增强文本视图，为其添加更多的浏览器引擎功能，则会得到一个过于复杂、臃肿的文本查看器或者一个功能不强的浏览器。

Android围绕WebKit浏览器引擎提供了一个包装器，叫做WebView，使用它可以获得真正强大的浏览器功能，而开销只有1MB。尽管在嵌入式设备上1MB仍是个不小的开销，但有很多情况很适合使用WebView。

WebView与其他Android视图的工作方式很类似，只是它有几个特定于浏览器的方法。通过实现上一个示例程序的嵌入式版本，下面我展示一下WebView的工作方式。这个例子叫做BrowserView，而不是BrowserIntent，因为它使用了嵌入式View类而不是Intent类。

首先使用以下设置创建新的“Hello, Android”项目：

```
Project name: BrowserView
Package name: org.example.browserview
Activity name: BrowserView
Application name: BrowserView
```

BrowserView的布局文件与BrowserIntent的类似，只是要在底部添加WebView：

```
BrowserView/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
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">
        <EditText
            android:id="@+id/url_field"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1.0"
            android:lines="1" />
        <Button
            android:id="@+id/go_button"
```

```

        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/go_button" />
    </LinearLayout>
    <WebView
        android:id="@+id/web_view"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1.0" />
    </LinearLayout>

```

此处利用两个LinearLayout控件让所有内容靠右显示。最外面的控件将屏幕分成上下两个区域：上面是文本区域和按钮，下面有WebView。最里面的LinearLayout与以前的作用一样：它只是使文本区域放在左边，按钮放在右边。

BrowserView的onCreate()方法与以前的几乎一样，只是现在要查找一个额外的视图：

```

BrowserView/src/org/example/browserview/BrowserView.java

import android.webkit.WebView;
// ...

public class BrowserView extends Activity {
    private WebView webView;
    // ...
    @Override
    public void onCreate(Bundle savedInstanceState) {
        // ...
        webView = (WebView) findViewById(R.id.web_view);
        // ...
    }
}

```

但openBrowser()方法则与以前不同：

```

BrowserView/src/org/example/browserview/BrowserView.java

/** Open a browser on the URL specified in the text box */
private void openBrowser() {
    webView.loadUrl(urlText.getText().toString());
    webView.requestFocus();
}

```

loadUrl()方法会让浏览器引擎开始加载并显示给定地址的网页。实际的加载可能需要一些时间，全部完成后它会立即返回。

还需要对程序进行更多的修改。在AndroidManifest.xml文件中的<application>标签前添加下面一行代码：

```
BrowserView/AndroidManifest.xml
```

```
<uses-permission android:name="android.permission.INTERNET" />
```

如果没有这一行, Android不会让应用程序访问因特网, 还会显示“Web page not available”(网页不可用)错误。

V// 小乔爱问……

为什么BrowserIntent不需要<uses-permission>

在上一个例子中, BrowserIntent只需启动一个意图就能请求其他应用程序查看网页。其他应用程序(浏览器)需要在其自己的AndroidManifest.xml中请求获得因特网权限。

现在试着运行一下程序, 输入以http://开头的有效网址。按Enter键或选择Go按钮时, 应该会出现网页(参见图7-3)。



图7-3 使用WebView嵌入浏览器

WebView提供了很多其他方法，可以使用它们控制要显示什么内容或者获得状态更改的通知。

可以在WebView的在线文档中找到这些方法的完整列表，下面介绍一些最可能用到的方法。

- ❑ `addJavascriptInterface()`：允许通过JavaScript访问Java对象（下一节会看到有关此方面的更多内容）。
- ❑ `createSnapshot()`：创建当前页的屏幕快照。
- ❑ `getSettings()`：返回用于控制设置的WebSettings对象。
- ❑ `loadData()`：将给定的字符串数据加载到浏览器中。
- ❑ `loadDataWithBaseURL()`：使用基本URL加载给定的数据。
- ❑ `loadUrl()`：利用给定的URL加载Web页面。
- ❑ `setDownloadListener()`：为下载事件注册回调函数，如用户下载.zip或.apk文件时。
- ❑ `setWebChromeClient()`：为需要在WebView矩形之外完成的事件注册回调函数，如更新标题或进度条，或打开JavaScript对话框。
- ❑ `setWebViewClient()`：允许应用程序在浏览器中设置挂钩以获得事件，如资源加载、按键和授权请求。
- ❑ `stopLoading()`：停止加载当前的页面。

WebView控件提供的最强大功能是在WebView和容纳它的Android应用程序之间进行交流。下面将详细介绍该功能。

7

7.3 JavaScript 与 Java 通信

Android设备具有很多超酷的功能，例如存储本地数据、绘制图形、播放音乐、打电话和确定其位置。如果可以通过网页访问这个功能不是更好吗？使用嵌入式WebView控件即可做到这一点。

关键在于WebView类中的`addJavascriptInterface()`方法。可以使用它扩展嵌入式浏览器内的DOM（Document Object Model，文档对象模型），并定义JavaScript代码可以访问的新对象。JavaScript代码调用该方法时，实际上它会调用Android程序中的方法。

也可以从Android程序中调用JavaScript方法。需要做的就是调用loadUrl()方法,将URL以javascript:要执行的代码的形式传递给它。浏览器会在当前页面内执行给定的JavaScript表达式,而不是转到新页面。你可以调用一个方法,更改JavaScript变量,修改浏览器文档等。

小乔爱问……

允许JavaScript调用Java是不是很危险

只要允许网页访问本地资源或调用浏览器沙箱(sandbox)之外的函数,都需要仔细考虑一下安全性问题。例如,你肯定不想创建一个方法,允许JavaScript读取任意路径名处的数据,因为这样做可能会将私人数据暴露给了解你所用方法和文件名的恶意网站。

要牢记以下几点。首先,不要依靠于含糊的安全性。对能够使用你方法的页面和这些方法能够进行的操作进行限制。记住安全性的黄金准则:不要试图排除什么,关键是控制什么可以进来。换句话说,不要试图检查出所有疑似问题(例如,查询中不合法的字符)。你一定会漏掉某些方面。应该做的是禁止所有东西,然后仅允许一些已知安全的东西通过。

为了演示WebView中JavaScript与Android程序中Java之间的调用,我们现在构建一个程序,该程序一半是HTML/JavaScript,一半是Android(参见图7-4)。应用程序窗口的上半部分是WebView控件,下面部分是来自Android用户界面的TextView和Button。单击按钮和链接时,它会在两个环境之间进行调用。



图7-4 在Android和嵌入式WebView之间通信

首先使用以下参数创建“Hello, Android”程序:

```
Project name: LocalBrowser
Package name: org.example.localbrowser
Activity name: LocalBrowser
Application name: LocalBrowser
```

该程序的用户界面将分为两部分。第一部分在Android布局文件res/layout/main.xml中定义:

```
LocalBrowser/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">

    <WebView
        android:id="@+id/web_view"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_weight="1.0" />

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_weight="1.0"
        android:padding="5sp">
        <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:textSize="24sp"
            android:text="TextView" />
        <Button
            android:id="@+id/button"

            android:text="@string/call_javascript_from_android"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="18sp" />
        <TextView
            android:id="@+id/text_view"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:textSize="18sp" />
    </LinearLayout>
</LinearLayout>
```

第二部分是将被加载到WebView中的index.html文件。该文件会放入assets而不是res目录,因为它不是编译的资源。安装程序时,assets目录中的任何内容都被完整地复制到本地存储设备。该目录用于存放浏览器无需连接到网络即可查看的

HTML、图像和脚本的本地副本。

```

LocalBrowser/assets/index.html
Line 1  <html>
-      <head>
-      <script language="JavaScript">
-          function callJS(arg) {
5              document.getElementById('replaceme').innerHTML = arg;
-          }
-      </script>
-      </head>
-      <body>
10     <h1>WebView</h1>
-     <p>
-     <a href="#" onclick="window.alert('Alert from JavaScript')">
-         Display JavaScript alert</a>
-     </p>
15    <p>
-     <a href="#" onclick="window.android.callAndroid('Hello from Browser')">
-         Call Android from JavaScript</a>
-     </p>
-     <p id="replaceme">
20    </p>
-    </body>
-    </html>

```

index.html的第4行定义了callJS()函数，我们的Android程序稍后将调用该函数。它接收一个字符串参数，并将该参数插入第19行的replaceme标签。

在图7-4中，你会看到两个HTML链接，这些链接从第12行开始定义。第一个链接只是调用标准的window.alert()函数，以打开一个其中显示一条短信的窗口。第二个链接在第16行，它调用window.android对象的callAndroid()方法。如果通过普通Web浏览器打开这个页面，window.android是没有定义的。但是因为要将浏览器嵌入到Android应用程序，所以可以自己定义这个对象，以便页面可以使用它。

下面看一下LocalBrowser类中的Android代码。以下是其中的基本内容：

```

LocalBrowser/src/org/example/localbrowser/LocalBrowser.java
Line 1  package org.example.localbrowser;
-
-  import android.app.Activity;
-  import android.os.Bundle;
5  import android.os.Handler;
-  import android.util.Log;
-  import android.view.View;
-  import android.view.View.OnClickListener;
-  import android.webkit.JsResult;
10 import android.webkit.WebChromeClient;

```

```

import android.webkit.WebView;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

15
public class LocalBrowser extends Activity {
    private static final String TAG = "LocalBrowser";
    private final Handler handler = new Handler();
    private WebView webView;
    private TextView textView;
    private Button button;

    @Override
    public void onCreate(Bundle savedInstanceState) {
25        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        // Find the Android controls on the screen
        webView = (WebView) findViewById(R.id.web_view);
        textView = (TextView) findViewById(R.id.text_view);
        button = (Button) findViewById(R.id.button);
        // Rest of onCreate follows...
30    }
}

```

注意第18行中Handler对象的初始化。JavaScript调用进入专用于浏览器的特殊线程，但Android用户界面调用只能通过主（GUI）线程进行。我们将使用Handler类进行转换。

要通过JavaScript调用Android Java代码，需要用一個或多个方法定义普通的Java对象，如下所示：

```

LocalBrowser/src/org/example/localbrowser/LocalBrowser.java
/** Object exposed to JavaScript */
private class AndroidBridge {
    public void callAndroid(final String arg) { // must be final
        handler.post(new Runnable() {
            public void run() {
                Log.d(TAG, "callAndroid(" + arg + ")");
                textView.setText(arg);
            }
        });
    }
}

```

JavaScript调用callAndroid()方法时，应用程序会创建新的Runnable对象，并使用Handler.post()将该对象放到主线程的运行队列中。主线程一有机会就会调用run()方法。这将调用setText()方法来更改TextView对象上的文本。

现在需要将onCreate()方法中的所有东西连接起来了。首先打开JavaScript(默认它是关闭的)并注册到JavaScript的桥梁:

```
LocalBrowser/src/org/example/localbrowser/LocalBrowser.java
// Turn on JavaScript in the embedded browser
webView.getSettings().setJavaScriptEnabled(true);

// Expose a Java object to JavaScript in the browser
webView.addJavascriptInterface(new AndroidBridge(),
    "android");
```

然后创建匿名的WebChromeClient对象并用setWebChromeClient()方法注册它。

```
LocalBrowser/src/org/example/localbrowser/LocalBrowser.java
// Set up a function to be called when JavaScript tries
// to open an alert window
webView.setWebChromeClient(new WebChromeClient() {
    @Override
    public boolean onJsAlert(final WebView view,
        final String url, final String message,
        JsResult result) {
        Log.d(TAG, "onJsAlert(" + view + ", " + url + ", "
            + message + ", " + result + ")");
        Toast.makeText(LocalBrowser.this, message, 3000).show();
        return false;
    }
});
```

术语chrome此处是指浏览器窗口四周所有的装饰。如果这是一个成熟的浏览器客户端,则需要处理导航、书签、菜单等。但在本示例中,需要做的是更改在浏览器试图打开JavaScript警告(使用window.alert())时JavaScript代码执行的操作。在onJsAlert()内,使用Android Toast类创建将短暂出现(在本示例中出现3000毫秒,即3秒)的消息窗口。

完成WebView的配置后,可以使用loadUrl()方法来加载本地网页:

```
LocalBrowser/src/org/example/localbrowser/LocalBrowser.java
// Load the web page from a local asset
webView.loadUrl("file:///android_asset/index.html");
```

对于Android浏览器引擎来说, file:///android_asset/filename形式的URL(注意有3个斜杠)具有特殊的意义。你可能猜到了,它指的是assets目录中的文件。在本示例中,我们正在加载之前定义的index.html文件。

最后一件事情是连接屏幕底部的按钮，使其能够进行JavaScript调用（从Java到JavaScript的调用）。

```
LocalBrowser/src/org/example/localbrowser/LocalBrowser.java
// This function will be called when the user presses the
// button on the Android side
button.setOnClickListener(new OnClickListener() {
    public void onClick(View view) {
        Log.d(TAG, "onClick(" + view + ")");
        webView.loadUrl("javascript:callJS('Hello from Android')");
    }
});
```

为此，使用setOnClickListener()方法给按钮单击设置一个监听器。按钮被按下时，系统会调用onClick()方法，而该方法又会调用webView.loadUrl()方法，并为其传递要在浏览器中计算的JavaScript表达式。该表达式会调用在index.html中定义的callJS()函数。

现在运行该程序并尝试操作一下。单击Display JavaScript alert将出现一个Android消息窗口。单击Call Android from JavaScript，字符串Hello from Browser将显示在Android文本控件中。最后，按下Call JavaScript from Android按钮，字符串Hello from Android将被发送到浏览器并被插入到HTML中，该字符串将显示在网页末端。

有时不需要显示网页，但是需要访问某种Web服务或其他服务器端资源。在下一节中，我将介绍如何实现该任务。

7.4 使用 Web 服务

Android提供了一整套Java标准网络API集，如java.net.HttpURLConnection包，你可以在程序中使用这些API。值得一提的是可以进行异步调用，这样程序的用户界面可以随时响应。

考虑一下，如果在主（GUI）线程中进行块网络调用将会发生什么。在调用返回前（它可能永远都没有返回），你的应用程序无法响应任何用户界面事件，如击键或按下按钮。用户会感觉程序好像挂起了。很显然，这是必须避免的情况。

Java.util.concurrent包最适合做这种工作。这个包最初由Doug Lea创建，作为一个独立的库，后来被合并到Java 5中。该包支持比常规Java Thread类更高级别

的并发程序设计。ExecutorService类管理一个或多个线程，你只要将任务（Runnable或Callable的实例）提交到执行器，让它们运行即可。然后会返回Future类的一个实例，该实例是对你的任务将返回的某些未知未来值（如果有）的引用。可以对创建的线程数量进行限制，如有必要可以中断正在运行的任务。

为了说明这些概念，我们创建一个将调用Google Translation API的有趣小程序。是否嘲笑过本国语言与外语之间的奇怪翻译，尤其是计算机翻译的？该程序会让用户以一种语言输入一个短语，让Google将它翻译成另一种语言，然后再让Google将它翻译回第一种语言。理想情况是最终得到的短语与开始输入的内容相同，但情况却并非总是如此，如图7-5所示。



图7-5 机器翻译仍然在不断完善中

要使用该程序，只要选择开始语言和目标语言，然后输入短语即可。输入后，程序将使用Google Translation Web服务将文本翻译成目标语言，然后再从目标语言翻译回来。

迷失在翻译中

第一次想到这个示例时,我觉得很容易得到一些有趣的结果。糟糕的是(或者说幸运的是,看你是怎么想的),Google服务确实不错,尤其是在处理罗曼斯语言,如英语和法语时。如果你发现了翻译得非常有趣的例子,可以将它们贴到本网站的讨论论坛上(<http://pragprog.com/titles/eband>),供大家娱乐。

要创建这个应用程序,首先使用以下参数创建“Hello, Android”应用程序:

```
Project name: Translate
Package name: org.example.translate
Activity name: Translate
Application name: Translate
```

因为该示例程序将访问因特网,进行Web服务调用,所以需要Android赋予我们一定的权限。将这行代码添加到AndroidManifest.xml文件中<application>XML标签的后面:

Translate/AndroidManifest.xml

```
<uses-permission android:name="android.permission.INTERNET" />
```

本示例的布局比平常的示例要复杂一些,所以要使用TableLayout视图。利用TableLayout可按行和列的方式安排视图,可以处理对齐并延伸列使其适合内容。这类似于使用HTML中的<table>和<tr>标签。

Translate/res/layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TableLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:stretchColumns="1"
        android:padding="10dip">
        <TableRow>
            <TextView android:text="@string/from_text" />
            <Spinner android:id="@+id/from_language" />
        </TableRow>
        <EditText
            android:id="@+id/original_text"
            android:hint="@string/original_hint"
            android:padding="10dip"
            android:textSize="18sp" />
        <TableRow>
            <TextView android:text="@string/to_text" />
            <Spinner android:id="@+id/to_language" />
        </TableRow>
    </TableLayout>
</ScrollView>
```

```

</TableRow>
<TextView
    android:id="@+id/translated_text"
    android:padding="10dip"
    android:textSize="18sp" />
<TextView android:text="@string/back_text" />
<TextView
    android:id="@+id/retranslated_text"
    android:padding="10dip"
    android:textSize="18sp" />
</TableLayout>
</ScrollView>

```

在这个示例中一共有5个行，每行都包含一列或两列。注意如果一行中只有一个视图，则不必使用TableRow来包含它。另外，不必像使用LinearLayout一样，在每个视图上都使用android:layout_width=和android:layout_height=。

Spinner类是一个新面孔，我们以前没见过。它类似于其他用户界面工具箱中的组合框。用户选择微调控制项（spinner）（例如在触摸屏上触摸它）后会出现一个可能值的列表，从中可选择值^①。在这个示例中，要使用这个控件从语言列表中选择语言。

实际列表作为Android资源存储在文件res/values/arrays.xml中：

```

transcode/res/values/arrays.xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <array name="languages">
        <item>Bulgarian (bg)</item>
        <item>Chinese Simplified (zh-CN)</item>
        <item>Chinese Traditional (zh-TW)</item>
        <item>Catalan (ca)</item>
        <item>Croatian (hr)</item>
        <item>Czech (cs)</item>
        <item>Danish (da)</item>
        <item>Dutch (nl)</item>
        <item>English (en)</item>
        <item>Filipino (tl)</item>
        <item>Finnish (fi)</item>
        <item>French (fr)</item>
        <item>German (de)</item>
        <item>Greek (el)</item>
        <item>Indonesian (id)</item>
        <item>Italian (it)</item>
        <item>Japanese (ja)</item>
        <item>Korean (ko)</item>
        <item>Latvian (lv)</item>
    </array>

```

^① 在SDK的早期版本中，用户还能使用箭头或D-pad按钮在所有值之间循环选择。但是从0.9_beta版本开始，谷歌公司决定将这个功能取消，让这个类的操作变得不太灵活了。

```

<item>Lithuanian (lt)</item>
<item>Norwegian (no)</item>
<item>Polish (pl)</item>
<item>Portuguese (pt-PT)</item>
<item>Romanian (ro)</item>
<item>Russian (ru)</item>
<item>Spanish (es)</item>
<item>Serbian (sr)</item>
<item>Slovak (sk)</item>
<item>Slovenian (sl)</item>
<item>Swedish (sv)</item>
<item>Ukrainian (uk)</item>
</array>
</resources>

```

这样就定义了一个称为languages的列表，它包含Google Translation API所能识别的大部分语言。注意，每个值都有一个全称（如Spanish）和简称（如es）。将语言传递给Google翻译服务的时候会使用简称。

现在开始修改Translate类。以下是基本内容：

translate/src/org/example/translate/Translate.java

```

Line 1 package org.example.translate;
-
- import java.util.concurrent.ExecutorService;
- import java.util.concurrent.Executors;
5 import java.util.concurrent.Future;
- import java.util.concurrent.RejectedExecutionException;
-
- import android.app.Activity;
- import android.os.Bundle;
10 import android.os.Handler;
- import android.text.Editable;
- import android.text.TextWatcher;
- import android.view.View;
- import android.widget.AdapterView;
15 import android.widget.AdapterView.OnItemClickListener;
- import android.widget.ArrayAdapter;
- import android.widget.EditText;
- import android.widget.Spinner;
- import android.widget.TextView;
- import android.widget.AdapterView.OnItemClickListener;
20
- public class Translate extends Activity {
-     private Spinner fromSpinner;
-     private Spinner toSpinner;
-     private EditText origText;
25 private TextView transText;
-     private TextView retransText;
-
-     private TextWatcher textWatcher;
-     private OnItemSelectedListener itemListener;
30

```

```

private Handler guiThread;
private ExecutorService transThread;
private Runnable updateTask;
private Future transPending;

35  @Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    setContentView(R.layout.main);
    initThreading();
    findViews();
    setAdapters();
    setListeners();
45 }
}

```

声明几个变量后,从第37行开始定义onCreate()方法,进行线程和用户界面的初始化。不要担心,随后会详细介绍它所调用的其他方法。

从第42行开始调用的findViews()方法用来获取布局文件中定义的所有用户界面元素的句柄:

translate/src/org/example/translate/translate.java

```

/** Get a handle to all user interface elements */
private void findViews() {
    fromSpinner = (Spinner) this.findViewById(R.id.from_language);
    toSpinner = (Spinner) this.findViewById(R.id.to_language);
    origText = (EditText) this.findViewById(R.id.original_text);
    transText = (TextView) this.findViewById(R.id.translated_text);
    retransText = (TextView) this.findViewById(R.id.retranslated_text);
}

```

在第43行的onCreate()调用的setAdapters()方法,为微调控制项定义了数据源:

translate/src/org/example/translate/translate.java

```

/** Define data source for the spinners */
private void setAdapters() {
    // Spinner list comes from a resource,
    // Spinner user interface uses standard layouts
    ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(
        this, R.array.languages,
        android.R.layout.simple_spinner_item);
    adapter.setDropDownViewResource(
        android.R.layout.simple_spinner_dropdown_item);
    fromSpinner.setAdapter(adapter);
    toSpinner.setAdapter(adapter);

    // Automatically select two spinner items
    fromSpinner.setSelection(8); // English (en)
    toSpinner.setSelection(11); // French (fr)
}

```

在Android中, Adapter类用于将数据源(在本示例中,是在arrays.xml中定义的语言数组)绑定到用户界面控件(在本示例中是微调控制项)。我们为列表中的各个项和你选择微调控制项时看到的下拉框使用Android提供的标准布局。

接下来在setListener()例程(从第44行的onCreate()调用)中建立用户界面处理程序:

```
Translate/src/org/example/translate/Translate.java
/* Setup user interface event handlers */
private void setListeners() {
    // Define event listeners
    textWatcher = new TextWatcher() {
        public void beforeTextChanged(CharSequence s, int start,
            int count, int after) {
            /* Do nothing */
        }
        public void onTextChanged(CharSequence s, int start,
            int before, int count) {
            queueUpdate(1000 /* milliseconds */);
        }
        public void afterTextChanged(Editable s) {
            /* Do nothing */
        }
    };
    itemListener = new OnItemSelectedListener() {
        public void onItemSelected(AdapterView parent, View v,
            int position, long id) {
            queueUpdate(200 /* milliseconds */);
        }
        public void onNothingSelected(AdapterView parent) {
            /* Do nothing */
        }
    };
    // Set listeners on graphical user interface widgets
    origText.addTextChangedListener(textWatcher);
    fromSpinner.setOnItemSelectedListener(itemListener);
    toSpinner.setOnItemSelectedListener(itemListener);
}
```

此处定义了两个监听器:一个在要翻译的文本发生更改时调用,另一个是在语言发生更改时调用。queueUpdate()使用一个Handler将延迟更新请求放在了主线程的任务列表中。我们为文本变更使用了1000毫秒的延迟,为语言变更使用了200毫秒的延迟。


```

transText.setText(R.string.translating);
retransText.setText(R.string.translating);

// Begin translation now but don't wait for it
try {
    TranslateTask translateTask = new TranslateTask(
        Translate.this, // reference to activity
        original, // original text
        getLang(fromSpinner), // from language
        getLang(toSpinner) // to language
    );
    transPending = transThread.submit(translateTask);
} catch (RejectedExecutionException e) {
    // Unable to start new task
    transText.setText(R.string.translation_error);
    retransText.setText(R.string.translation_error);
}
}
};
}

```

此处有两个线程：用于用户界面的主Android线程，以及为运行实际翻译作业而将要创建的翻译线程。我们用Android Handler代表第一个线程，用Java的ExecutorService代表第二个线程。

第11行定义了更新任务，通过queueUpdate()方法调度。它开始运行后，首先获取当前要翻译的文本，然后准备将翻译作业发送到翻译线程。它会取消正在进行的任何翻译（第18行），处理没有要翻译的文本的情况（第22行），并用字符串Translating...填充两个将要显示译文的文本控件（第26行）。Translating...稍后将被实际的译文所取代。

最后在第31行创建了TranslateTask实例，为其添加了对Translate活动的引用，使其可以回调以更改文本、包含原始文本的字符串以及在微调控制项中所选两种语言的简称。第37行向翻译线程提交了新任务，返回对Future返回值的引用。在本示例中，实际上没有返回值，因为TranslateTask会直接更改GUI，但第18行仍然引用了Future，以便在必要时取消翻译。

Translate类的最后还定义了在其他地方使用的几个实用函数：

```

// translate/src/org/example/translate/translate.java

/** Extract the language code from the current spinner item */
private String getLang(Spinner spinner) {
    String result = spinner.getSelectedItem().toString();
    int lparen = result.indexOf('(');
    int rparen = result.indexOf(')');
    result = result.substring(lparen + 1, rparen);
}

```

```

        return result;
    }

    /** Request an update to start after a short delay */
    private void queueUpdate(long delayMillis) {
        // Cancel previous update if it hasn't started yet
        guiThread.removeCallbacks(updateTask);
        // Start an update if nothing happens after a few milliseconds
        guiThread.postDelayed(updateTask, delayMillis);
    }

    /** Modify text on the screen (called from another thread) */
    public void setTranslated(String text) {
        guiSetText(transText, text);
    }

    /** Modify text on the screen (called from another thread) */
    public void setRetranslated(String text) {
        guiSetText(retransText, text);
    }

    /** All changes to the GUI must be done in the GUI thread */
    private void guiSetText(final TextView view, final String text) {
        guiThread.post(new Runnable() {
            public void run() {
                view.setText(text);
            }
        });
    }
}

```

getLang()方法指出在微调控制项中当前选择了哪一项，获得该项的字符串，并解析Translation API需要的语言简称代码。

queueUpdate()方法将更新请求放到主线程的请求队列中，但是告诉它在实际运行之前等待一会。如果队列中已经有一个请求了，就将该请求删除。

Web 服务返回翻译结果后，TranslateTask 将使用 setTranslated() 和 setRetranslated() 方法来更新用户界面。这两个方法都会调用一个名为 guiSetText() 的私有函数。

guiSetText() 函数使用 Handler.post() 方法来要求主 GUI 线程更新 TextView 控件上的文本。这个额外的步骤是必需的，因为你不能从非用户界面线程调用用户界面函数，并且 guiSetText() 将被翻译线程调用。

以下是 Translate 示例的 res/values/strings.xml 文件：

```

translate/res/values/strings.xml
<?xml version="1.0" encoding="utf-8"?>
<resources>

```



```

<string name="app_name">Translate</string>
<string name="from_text">From:</string>
<string name="to_text">To:</string>
<string name="back_text">And back again:</string>
<string name="original_hint">Enter text to translate</string>
<string name="empty"></string>
<string name="translating">Translating...</string>
<string name="translation_error">(Translation error)</string>
<string name="translation_interrupted">(Translation
    interrupted)</string>
</resources>

```

这里我就不提供TranslateTask类的源代码了，因为它太长了，并且除了几条调试信息，也没有特定于Android的内容。如果想要查看通过HttpURLConnection调用RESTful Web服务、用JSON（JavaScript Object Notation，JavaScript对象表示法）格式解析结果、处理各种网络错误和中断请求的示例，请从本书的网站上下载源代码。

7.5 快速阅读指南

本章介绍了很多基础知识，从打开简单的网页到使用异步Web服务。HTML/JavaScript程序设计超出了本书的范围，但读者很容易找到相关的参考资料。如果要使用诸如ExecutorService这样的类进行并发程序设计，我建议你阅读Brian Goetz编写的*Java Concurrency in Practice*^①[Goe06]。

下一章介绍如何通过定位和传感器服务实现更高级别的交互性。如果急于了解有关数据源和数据绑定的更多内容，可直接跳至第9章。

7

① 此书中文版《Java并发编程实践》已经由电子工业出版社出版。——编者注

Android平台使用了很多不同的技术。有些是新技术，有些技术则借鉴自其他设备。Android的独特之处在于结合使用这些技术的方式。本章将介绍以下内容：

- 通过便宜的GPS设备实现位置感知；
- 手持式加速计，例如任天堂Wii无线手柄中使用的加速计；
- Mashup，通常将地图与其他信息组合。

Android开发者挑战赛的几名优胜者使用这些概念为用户提供了绝妙的体验。例如，Locale应用程序^①可以根据你所在的位置调整手机上的设置。你是否总是忘了在工作或看电影时将手机调成震动？Locale可使用本章介绍的Android Location API帮你完成这个任务。

1 位置，位置，位置

目前GPS（Global Positioning System，全球定位系统）系统有31颗卫星围绕着地球旋转，它们可以帮助你找到去杂货店的路。GPS最初是由军方开发，然后开始允许实现民用，它会向地面的接收器（例如Android手机中的接收器）发出高精度的时间信号。通过准确的信号接收和很少的计算，GPS芯片就可以指出你现在的位置，误差在15.24米以内^②。

除了GPS，Android还支持使用附近手机基站的信号信息来计算出你的位置，如

① 参见网页<http://www.androidlocale.com>。

② 你不需要了解GPS的工作方式就可以使用它，但是如果对其工作方式感兴趣，可以访问<http://adventure.howstuff-works.com/gps.htm>。

果连接到Wi-Fi热区，它还可以使用Wi-Fi。记住，所有这些位置提供者在某种程度上都是不可靠的。例如，如果你走进某栋建筑物，就收不到GPS信号了。

为了演示Android的位置服务，我们编写一个测试程序，显示你当前的位置并让它随着你的移动更新位置。可以在图8-1中看到这个程序。

1// 小乔爱问……

GPS会让其他人窥探到我的行踪吗

不会。GPS接收器只是一种接收器。只有GPS芯片以及运行在Android设备中的所有程序知道位置信息。除非某个程序故意传送该信息，否则没有人能用它找到你。

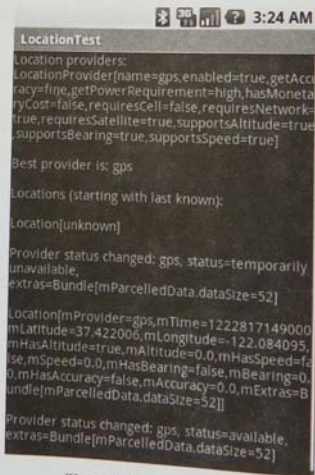


图8-1 测试LocationManager

1.1 我在哪里

首先在New Project向导中使用以下参数创建“Hello, Android”应用程序。

```
Project name: LocationTest
Package name: org.example.locationtest
Activity name: LocationTest
Application name: LocationTest
```

Android权限负责保护对位置信息的访问。要获得访问许可,需要在AndroidManifest.xml文件中将这些行添加到<application>标签前面:

LocationTest/AndroidManifest.xml

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

在这个示例中,既支持较为精确的位置提供者,如GPS,也支持较为粗略的位置提供者,如手机基站三角定位。

对于用户界面,要将所有位置数据输出到一个滚动的TextView控件中,该控件在res/layout/main.xml中定义:

LocationTest/res/layout/main.xml

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

做好这些铺垫之后就可以开始编码了。以下是LocationTest类和onCreate()方法的基本内容(现在先忽略第15行对LocationListener的引用;稍后会对此进行介绍)。

LocationTest/src/org/example/locationtest/LocationTest.java

```
1 package org.example.locationtest;
2
3 import java.util.List;
4
5 import android.app.Activity;
6 import android.location.Criteria;
7 import android.location.Location;
```

```
- import android.location.LocationListener;
- import android.location.LocationManager;
10 import android.location.LocationProvider;
- import android.os.Bundle;
- import android.widget.TextView;
-
- public class LocationTest extends Activity implements
15     LocationListener {
-     private LocationManager mgr;
-     private TextView output;
-     private String best;
-
-     @Override
20     public void onCreate(Bundle savedInstanceState) {
-         super.onCreate(savedInstanceState);
-         setContentView(R.layout.main);
-
-         mgr = (LocationManager) getSystemService(LOCATION_SERVICE);
25         output = (TextView) findViewById(R.id.output);
-
-         log("Location providers:");
-         dumpProviders();
-
30         Criteria criteria = new Criteria();
-         best = mgr.getBestProvider(criteria, true);
-         log("\nBest provider is: " + best);
-
-         log("\nLocations (starting with last known):");
35         Location location = mgr.getLastKnownLocation(best);
-         dumpLocation(location);
-     }
- }
```

Android定位服务的起点是第25行的`getSystemService()`方法调用。它返回一个`LocationManager`类，将其保存到一个字段中，稍后使用。

在第29行调用`dumpProviders()`方法输出系统中所有位置提供者的列表。

接下来需要选择一个提供者。我曾经看到过有些示例只挑选第一个可用的提供者，但是我建议你使用`getBestProvider()`方法，如这里所示。Android会根据你提供的`Criteria`（参见第31行）选择最佳的提供者。如果对成本、能耗、准确性等有所限制，就应该使用这种方法。在本示例中没有任何限制。

根据所选的提供者，设备可能要花一些时间才能计算出当前位置。这个时间可能是几秒钟、一分钟，也许更长。但是，Android会记住提供者所返回的上一个位置，因此可以立即在第36行查询和输出这个位置。该位置可能已过期，例如设备关闭或因为你的移动而改变位置了——但这总比没有好。

知道位置只是成功的一半。下一步要做什么呢?

8.1.2 更新位置

要让Android通知你位置发生了变更,可在LocationManger项目上调用requestLocationUpdates()方法。为了省电,我们只在程序位于前台时更新。因此需要重写onResume()和onPause()方法,挂钩到Android活动生命周期方法中:

```
LocationTest/src/org/example/locationtest/LocationTest.java

@Override
protected void onResume() {
    super.onResume();
    // Start updates (doc recommends delay >= 60000 ms)
    mgr.requestLocationUpdates(best, 15000, 1, this);
}

@Override
protected void onPause() {
    super.onPause();
    // Stop updates to save power while app paused
    mgr.removeUpdates(this);
}
```

应用程序继续运行时,调用requestLocationUpdates()方法开始更新进程。它需要4个参数:提供者名称、延迟(这样不会频繁收到更新)、最小距离(小于此距离的变化可以忽略)和LocationListener对象。

应用程序暂停时,调用removeUpdates()方法来停止获得更新。如果暂时不需要位置提供者,它会关闭一会儿。

现在了解了为什么LocationTest要实现LocationListener,因为这样可以只传入一个该活动的引用,而不用建立新的监听器对象。这会在运行时节省约1KB的内存。

以下是该接口所需的4个方法的定义:

```
LocationTest/src/org/example/locationtest/LocationTest.java

public void onLocationChanged(Location location) {
    dumpLocation(location);
}

public void onProviderDisabled(String provider) {
    log("\nProvider disabled: " + provider);
}
```

```
public void onProviderEnabled(String provider) {
    log("\nProvider enabled: " + provider);
}

public void onStatusChanged(String provider, int status,
    Bundle extras) {
    log("\nProvider status changed: " + provider + ", status="
        + S[status] + ", extras=" + extras);
}
```

其中最重要的方法是onLocationChanged()。

顾名思义，每次提供者通知设备位置发生了变动时都会调用它。onProviderDisabled()、onProviderEnabled()、onStatusChanged()方法可以用于切换到其他提供者，防止你选择的第一个提供者不可用。

LocationTest其他方法(log()、dumpProviders()和dumpLocation())的代码就没什么意思了，所以这里就不介绍了。可以在本书网站的可下载示例中找到所有代码。

8.1.3 模拟说明

如果在实际设备上运行LocationTest示例，它会在你四处走动时显示你的当前位置。在模拟器上，它会使用伪GPS提供者并且总是返回同一位置，除非你手动更改它。现在演示一下。

在Eclipse中，你可以使用Emulator Control视图(Window > Show View > Other... > Android > Emulator Control)更改你的模拟位置。滚动到屏幕底部，会找到一个地方可以手动输入经度和纬度信息。单击Send按钮时，Eclipse会将新位置发送到模拟设备，你会看见位置信息显示在关注该信息的所有程序中。

还可以在Eclipse之外运行DDMS程序并用同样的方式发送假的地址变更。除了手动方式一次更新一个位置，还可以从外部文件读取已记录的路径。有关此内容的更多信息，请参阅DDMS文档^①。

通过Android位置提供者，可以在全球范围内找到你的位置。如果需要更多当地的信息，如倾斜度和温度，就必须使用另一个API。这正是下一小节的主题。

^① 参见网页<http://androidappdocs.appspot.com/guide/developing/tools/ddms.html>。

8.2 充分利用传感器

假设你正在编写一款赛车游戏，要让玩家能驾驶屏幕上显示的汽车。一种方法是用按钮，像索尼Playstation或任天堂DS中的赛车游戏一样。按右键向右转，按左键向左转，按住另一个键加大油门。这种操纵方法可以工作，但是很不逼真。

看过其他人玩这种游戏吗？他们在拐发卡弯时将控制杆从一边摇到另一边，撞到另一辆汽车时控制杆会震动，加速时前倾，刹车时身体向后仰。如果真的可以通过这些运动来操纵游戏，不是很奇妙的一件事吗？现在已经可以做到这一点了。

8.2.1 了解传感器

Android SDK支持很多不同类型的传感器设备。

- **SENSOR_ACCELEROMETER**: 测量x、y和z轴方向的加速度。
- **SENSOR_LIGHT**: 告诉你周围环境的明亮程度。
- **SENSOR_MAGNETIC_FIELD**: 返回x、y和z轴方向的磁力。
- **SENSOR_ORIENTATION**: 测量设备的偏航、颠簸程度和滚动。
- **SENSOR_ORIENTATION_RAW**: 与**SENSOR_ORIENTATION**一样，只是没有过滤。
- **SENSOR_PROXIMITY**: 提供传感器与某些对象之间的距离。
- **SENSOR_TEMPERATURE**: 测量周围环境的温度。
- **SENSOR_TRICORDER**: 将你的设备转换成功能齐备的Star Trek Tricorder（星际迷航三录仪）^①。

当然，并非所有的设备都提供这些功能。

Android的SensorManager类与LocationManager类似，只是它的更新频率更高，也许可以达到每秒几百次。要访问传感器，首先要调用getSystemService()方法，如下所示：

```
SensorTest/src/org/example/sensortest/SensorTest.java
```

```
private SensorManager mgr;
// ...
mgr = (SensorManager) getSystemService(SENSOR_SERVICE);
```

然后调用onResume()方法中的registerListener()开始获取更新，调用

^① 该死的，我只是个程序员，不是救世主。

onPause()方法中的unregisterListener()方法停止获取更新。

8.2.2 解析传感器的读数

每次值发生变化时,传感器服务将调用onSensorChanged()方法。该方法如下所示:

```
SensorTest/src/org/example/sensortest/SensorTest.java
public void onSensorChanged(int sensor, float[] values) {
    for (int i = 0; i < values.length; i++) {
        // ...
    }
}
```

所有传感器都返回一个浮点值数组。数组大小取决于具体的传感器;例如,SENSOR_TEMPERATURE仅返回一个值,即以摄氏度表示的温度。你可能不需要所有的返回值。例如,如果只需要指南针方位,可以使用SENSOR_ORIENTATION传感器返回的第一个数字。

将传感器读数(特别是来自加速计的信息)转换成有意义的信息是需要点智慧的。以下是要记住的一些提示:

- 加速计读数变化很大。需要使用某种加权平均方法使数据变化表现得更平滑一些,但要注意不能平滑过度,否则你的接口将感觉到延迟,动作不清晰。
- 传感器数据的传入是随意的。因此一次可能会获得好几个数字,然后有一个短暂的停顿,然后又收到一批。不要期望着它会匀速传入数据。
- 尝试在用户操作前做出预测。例如,最后3个读数显示了开始右滚,且每次都比上一次快一些。因此你就可以猜测出下一个读数会是什么,这是有一定准确性的,并开始根据猜测做出回应。

传感器最有挑战性的用途是动作游戏,这需要将玩家移动设备的方式与屏幕上显示的内容建立一对一的连接。糟糕的是,模拟器不打算大量用于这类应用。

8.2.3 模拟说明

据谷歌公司声称,根本不可能使用模拟器来测试传感器。大部分计算机没有内置光敏元件、GPS芯片或指南针。显然,如果在模拟器中运行SensorTest程序(可以在本书网站上获得),它不会显示任何结果。但是,一个叫做OpenIntents^①的项目

^①参见网页<http://www.openintents.org>。

提供了替代传感器的API，你可以调用这个API进行测试。

它的工作方式是将模拟器连接到台式机上运行的另一个叫做Sensor Simulator的应用程序。模拟器会显示一个虚拟手机图，可以在屏幕上用鼠标到处移动它（参见图8-2），然后它会将这些移动信息反馈给运行在模拟器上的Android程序。如果你的开发计算机上确实有实际的传感器（例如苹果公司的MacBook）或者可以通过蓝牙连接到Wii无线手柄，Sensor Simulator可以使用它作为数据源。

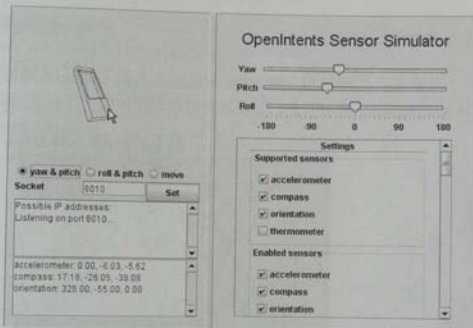


图8-2 使用Sensor Simulator欺骗传感器

这种方法的缺点是必须修改源代码才能使其工作。如果想尝试一下这种方法，访问OpenIntents网站可了解更多信息。我的建议是忘掉传感器模拟，使用实际的设备。不断地微调你的算法，直到感觉合适为止。

以上介绍了一些低级调用方法来获取你的位置以及如何查询传感器，获得诸如指南针方向等读数，实际上对于某些应用程序，你可以完全不用这些，只要使用Google Maps API就可以了。

3 地图功能

第一个Ajax“杀手级应用”就是Google Maps^①。使用JavaScript和XmlHttpRequest

① 参见网页<http://maps.google.com>。

对象，谷歌的工程师们开发了可拖动、可缩放，极其平滑的地图查看器，无需插件，即可运行在任何现代Web浏览器中。这种理念迅速被其他厂商复制，例如微软和雅虎，但是谷歌版的地图查看器无疑是最好的。

你可以在Android中使用这些基于Web的地图，可能需要使用嵌入的WebView控件，如7.2节所述。但是应用程序的架构就会比较繁杂。这也是谷歌公司创建MapView控件的原因。

8.3.1 嵌入 MapView

只要几行代码就可以将MapView直接嵌入到Android应用程序中。它提供了Google Maps的大部分功能，以及添加触摸事件的钩子（参见图8-3）。



图8-3 嵌入的地图显示你的当前位置

MapView类还可以连接到你的位置和传感器提供者。它可以在地图上显示你的当前位置，甚至显示一个指南针，指出目前的前进方向。下面我们创建一个简单的

程序来演示它的几个功能。

首先在向导中使用以下值创建“Hello, Android”应用程序:

```
Project name: MyMap
Package name: org.example.mymap
Activity name: MyMap
Application name: MyMap
```

编辑布局文件, 用占据整个屏幕的MapView取代它:

```
MyMap/res/layout/main.xml
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/frame"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <com.google.android.maps.MapView
        android:id="@+id/map"
        android:apiKey="MapAPIKey"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:clickable="true" />
</FrameLayout>
```

在此使用了FrameLayout, 以便稍后将缩放控件保持在地图的顶部。用你从谷歌获得的Google Maps API键替换MapAPIKey。

注意, 必须使用完全限定名(com.google.android.maps.MapView), 因为MapView不是标准的Android类。还需要在AndroidManifest.xml的<application>元素中使用<uses-library>标签。

```
MyMap/AndroidManifest.xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.example.mymap"
    android:versionCode="1"
    android:versionName="1.0.0">
    <uses-permission
        android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission
        android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission
        android:name="android.permission.INTERNET" />
    <application android:icon="@drawable/icon"
        android:label="@string/app_name">
        <uses-library android:name="com.google.android.maps" />
        <activity android:name=".MyMap"
```

```

        android:label="@string/app_name">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>
</manifest>

```

如果没有`<uses-library>`标签,将会得到运行时错误,提示没有发现`MapView`类。

除了较为精确和较为粗略的位置提供者, `MapView`还需要使用因特网访问,通过调用谷歌的服务器以获取多个地图图像拼接块。这些内容会自动缓存在应用程序目录中。

以下是`MyMap`类的基本内容:

`MyMap/src/org/example/mymap/MyMap.java`

```

package org.example.mymap;

import android.os.Bundle;
import android.view.Gravity;
import android.view.View;
import android.view.ViewGroup.LayoutParams;
import android.widget.FrameLayout;

import com.google.android.maps.MapActivity;
import com.google.android.maps.MapController;
import com.google.android.maps.MapView;
import com.google.android.maps.MyLocationOverlay;

public class MyMap extends MapActivity {
    private FrameLayout frame;
    private MapView map;
    private MapController controller;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        initMapView();
        initZoomControls();
        initMyLocation();
    }

    @Override
    protected boolean isRouteDisplayed() {
        // Required by MapActivity
        return false;
    }
}

```

最重要的部分是你的活动必须扩展MapActivity。MapActivity类向上连接后台线程，并连接到因特网以获得图像拼接块数据、处理缓存、播放动画、处理生命周期等。你只需正确地设置它并让它运行即可。

8.3.2 准备就绪

首先需要做的就是调用findViewById()，以访问MapView及其容器。可以在initMapView()方法中完成这个任务：

MyMap/src/org/example/mymap/MyMap.java

```
/** Find and initialize the map view. */
private void initMapView() {
    frame = (FrameLayout) findViewById(R.id.frame);
    map = (MapView) findViewById(R.id.map);
    controller = map.getController();
    map.setSatellite(true);
}
```

getController()方法返回一个MapController，使用返回的内容可定位并缩放地图。

就缩放而言，需要在initZoomControls()方法中创建和定位缩放控件。

MyMap/src/org/example/mymap/MyMap.java

```
/** Get the zoom controls and add them to the bottom of the map. */
private void initZoomControls() {
    View zoomControls = map.getZoomControls();
    FrameLayout.LayoutParams p = new FrameLayout.LayoutParams(
        LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT,
        Gravity.BOTTOM + Gravity.CENTER_HORIZONTAL);
    frame.addView(zoomControls, p);
}
```

MapView类处理关于缩放控件的一切，但它在屏幕上的位置除外。需要做的就是获得这些控件的句柄并使用将控件置于屏幕底部居中位置的布局将控件添加到父帧中。MapView将在用户平移地图时使控件可见并在平移结束时使其逐渐消失。

最后一步是告诉MapView使用initMyLocation()方法中提供的位置：

MyMap/src/org/example/mymap/MyMap.java

```
/** Start tracking the position on the map. */
private void initMyLocation() {
    final MyLocationOverlay overlay = new MyLocationOverlay(this, map);
    overlay.enableMyLocation();
    overlay.enableCompass(); // no effect in emulator
}
```

```

overlay.runOnUiThread(new Runnable() {
    public void run() {
        // Zoom in to current location
        controller.setZoom(8);
        controller.animateTo(overlay.getMyLocation());
    }
});
map.getOverlays().add(overlay);
}

```

// 小乔爱问……

为什么MapView是com.google.android.maps包的一员,而不是android.maps包的

android.*包中的所有代码都是Android核心的一部分。它是开源的,在任何Android设备上都可以得到。相反,地图是谷歌公司和数据提供者专有的,由谷歌公司支付这些提供者的地理信息和图片的费用。只要你同意某些条款,谷歌公司就会免费提供API。如果不满意这些限制,可以使用自己的视图,查找自己的数据源,但是这并不容易而且价格也不低。

Android提供了MyLocationOverlay类来完成大部分较难的工作。覆盖图(overlay)就是在地图上面所绘制的内容,在本示例中是显示你当前位置的闪烁点。可以调用enableMyLocation()告诉覆盖图开始监听位置更新,调用enableCompass()告诉覆盖图开始监听来自指南针的更新。

runOnUiThread()方法告诉覆盖图第一次从位置提供者获得位置读数时应该执行哪些操作。在本示例中设置了缩放级别,然后开始播放一个动画,将地图从现在所指的位置移到你所在的位置。

如果现在运行该程序,应该看到如图8-3所示的内容。触摸并拖动屏幕可移动地图,使用缩放按钮可放大地图。当你拿着手机四处走动时,地图上的点应该会跟着移动。

8.3.3 模拟说明

如果在模拟器上而不是实际设备上运行该程序,最初会看到缩小的世界地图,没有显示你当前位置的点。像以前一样,在Eclipse中使用Emulator Control视图(或

在独立的DDMS程序中)向示例程序提供虚假的GPS数据。在模拟器中运行时,指南针数据不会显示出来,因为无法模拟指南针传感器。

3.4 快速阅读指南

本章介绍了精彩的定位世界以及能感知环境的移动计算。这些技术配合宽带移动因特网的发展和计算与存储能力呈指数级增长等趋势,使人们与计算机以及人与人之间的交互方式发生了革命性的变化。

感知世界的另一种方法是看和听。Android提供了Camera类^①允许你使用手机的内置相机(如果有的话)拍照,当然也可以使用它做其他事情,例如作为条形码读取器。MediaRecorder类^②允许你记录并存储音频剪辑。这些内容超出了本书的范围,但是如果你的程序需要这些内容,可以参阅相关的在线文档。

说到存储,下一章将介绍如何使用SQL将结构化信息(例如位置变化记录、照片和注释)存储在手机上。如果对这个内容不感兴趣,可以跳到第10章,了解如何充分发挥Android隐藏的3D图形功能的潜能。

① 参见网页<http://androidappdocs.appspot.com/reference/android/hardware/Camera.html>。

② 参见网页<http://androidappdocs.appspot.com/reference/android/media/MediaRecorder.html>。

第6章介绍了如何以首选项或普通文件的形式保存数据。这种方法在数据量很少或者只有一种类型的数据（例如图片或音频文件）时很有效。但是，存储大量结构化数据可以采用一种更好的方法：关系型数据库。

在过去的30年中，数据库是企业应用程序开发中的重要部分，但是它们一直非常昂贵且不适用于较小规模的应用。随着小型嵌入式引擎（例如Android平台中包含的引擎）的出现，这种局面得到了改变。

本章将介绍如何使用Android的嵌入式数据库引擎SQLite。还将介绍如何使用Android的数据绑定功能将数据源与用户界面连接起来。最后介绍ContentProvider类，该类支持两个应用程序共享相同的数据。

9.1 SQLite 简介

SQLite^①是一个很小但功能强大的数据库引擎，由Richard Hipp博士于2000年开发。它无疑是全球部署最广泛的SQL数据库引擎。除了Android，你还能够在苹果公司的iPhone、Symbian电话、Mozilla Firefox、Skype、PHP、Adobe AIR、Mac OS X、Solaris和其他很多地方看到SQLite。

它之所以如此流行，有以下3个原因。

- 它是免费的。作者将其放在公共领域中，不会收取任何使用费。
- 它很小。目前的版本只有大约150KB，Android手机的内存完全可容纳下它。
- 它无需安装或管理。没有服务器，没有配置文件，也无需数据库管理员。

^① 参见网页<http://www.sqlite.org>。

SQLite许可

使用SQLite源代码不需要许可，因为它属于公共领域。尽管没有许可，但使用该源代码具有以下要求。

- 不能将其用于恶意用途。
- 获得别人对您的宽恕并宽恕别人。
- 自由分享，决不要索取多于付出的回报。

SQLite数据库就是一个文件。可以获取该文件并随意移动它，甚至将它复制到另一个系统（例如从手机复制到工作站），但它仍然能够很好地运行。Android将该文件存储在/data/data/ packagename/databases目录下（参见图9-1）。可以在Eclipse中使用adb命令或File Explorer视图（Window > Show View > Other... > Android > File Explorer）来查看、移动或删除它。



图9-1 SQLite将整个数据库存储在一个文件中

无需调用Java I/O例程来从程序中访问此文件，只需运行SQL（Structured Query Language，结构化查询语言）语句即可。通过其帮助器类和方便的方法，Android隐藏了一些语法，但是仍然需要了解一些SQL知识才能使用它。

9.2 SQL 基础

如果以前使用过Oracle、SQL Server、MySQL、DB2和其他数据库引擎，那么应该很熟悉SQL。你可以跳过本节，从9.3节继续学习。对于未使用过上述数据库引擎的人来说，这一节可以当作一次快速回顾。

要使用SQL数据库，只需提交SQL语句并获取结果。有3种主要的SQL语句类型：

DDL、修改和查询。

9.2.1 DDL 语句

一个数据库文件可以包含任意多个表。表由多行组成，每行包含特定数量的列。表的每一列拥有一个名称和一个数据类型（文本字符串、数字等）。首先要做的是运行DDL（Data Definition Language，数据定义语言）语句来定义这些表和列名称。下面这条语句创建一个包含3列的表。

SQLite/create.sql

```
create table mytable (
    _id integer primary key autoincrement,
    name text,
    phone text );
```

表中的某个列被指定为主键（primary key），这是一个数字，用于唯一地标识一行。autoincrement表示数据库会为每条记录的键加1，以确保其唯一性。根据约定，第一列通常命名为_id。_id列不是SQLite所必需的，但是在后面使用Android ContentProvider时需要它。

注意，与大多数数据库不同，SQLite中的列类型只是一些提示。如果尝试将一个字符串存储到一个整数列中或者将整数存储在字符串列中，SQLite同样能够正常运行。SQLite作者认为这个特点是一项功能，而不是一个bug。

9.2.2 修改语句

SQL提供了很多语句在数据库中插入、删除和更新记录。例如，要添加一些电话号码，可使用以下语句：

SQLite/insert.sql

```
insert into mytable values(null, 'Steven King', '555-1212');
insert into mytable values(null, 'John Smith', '555-2345');
insert into mytable values(null, 'Fred Smitheizen', '555-4321');
```

指定这些值的顺序与在create table语句中使用的顺序相同。为_id指定的值是null，因为SQLite将为我们计算出这个值。

9.2.3 查询语句

将数据加载到表之后，可以使用select语句来查询该表。例如，如果想要获

取第3个条目，可以执行以下语句：

```
SQLite/select3.sql
select * from mytable where(_id=3);
```

你可能更想根据一个人的姓名来查找其电话号码。可以使用以下语句查找姓名中包含Smith的所有记录。

```
SQLite/selectwhere.sql
select name, phone from mytable where(name like "%smith%");
```

记住，SQL是区分大小写的。关键字、列名称，甚至搜索字符串都可以用大写形式或小写形式来指定。

现在已经了解了足够多的SQL知识，让我们看看如何在一个简单的程序中利用这些知识。

9.3 你好，数据库

为了演示SQLite，我们创建一个名为Events的小应用程序，该应用程序将记录存储在一个数据库中并在以后显示它们。我们由简到难开始构建。在项目向导中使用以下值建立一个新的“Hello, Android”程序：

```
Project name: Events
Package name: org.example.events
Activity name: Events
Application name: Events
```

同样，也可以从本书的网站上下载完整的源代码。

需要在某个地方放置该数据库中描述的一些常量，所以要创建一个Constants接口：

```
Event1/src/org/example/events/Constants.java
package org.example.events;

import android.provider.BaseColumns;

public interface Constants extends BaseColumns {
    public static final String TABLE_NAME = "events";

    // Columns in the Events database
    public static final String TIME = "time";
```

```

    public static final String TITLE = "title";
}

```

每个事件都将作为events表中的一行进行存储。每行都包含一个_id、time和title列。_id是主键，在扩展的BaseColumns接口中声明。time和title分别用作时间戳和事件标题。

9.3.1 使用 SQLiteOpenHelper

接下来创建一个名为EventsData的帮助器类来表示数据库本身。这个类扩展自Android的SQLiteOpenHelper类，它负责管理数据库的创建和版本。你需要做的就是提供一个构造函数并重写两个方法。

Events1/src/org/example/events/EventsData.java

```

Line 1 package org.example.events;
-
- import static android.provider.BaseColumns._ID;
- import static org.example.events.Constants.TABLE_NAME;
5 import static org.example.events.Constants.TIME;
- import static org.example.events.Constants.TITLE;
- import android.content.Context;
- import android.database.sqlite.SQLiteDatabase;
- import android.database.sqlite.SQLiteOpenHelper;
10
- public class EventsData extends SQLiteOpenHelper {
-     private static final String DATABASE_NAME = "events.db";
-     private static final int DATABASE_VERSION = 1;
-
15     /** Create a helper object for the Events database */
-     public EventsData(Context ctx) {
-         super(ctx, DATABASE_NAME, null, DATABASE_VERSION);
-     }
-
20     @Override
-     public void onCreate(SQLiteDatabase db) {
-         db.execSQL("CREATE TABLE " + TABLE_NAME + " (" + _ID
-             + " INTEGER PRIMARY KEY AUTOINCREMENT, " + TIME
-             + " INTEGER, " + TITLE + " TEXT NOT NULL);");
25     }
-
-     @Override
-     public void onUpgrade(SQLiteDatabase db, int oldVersion,
-         int newVersion) {
30         db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
-         onCreate(db);
-     }
- }

```



小乔爱问……

为什么Constants是一个接口

它是Java中的一项内容。我不知道如何向你介绍它，但我也不同意在每次使用一个常量时都重复这个类名。例如，我只希望输入TIME，而不是Constants.TIME。通常，在Java中实现这一目的的方式是通过接口。类可以继承自Constants接口，然后在引用任何字段时省去接口名。如果看一下BaseColumns接口，就会看到Android编程人员使用了同样的技巧。

但是从Java 5开始有一种更好的方法：静态导入。我将在EventsData和本章中的其他类中使用该方法。因为Constants是一个接口，所以可以根据你的喜好通过旧方法或新方法使用它。

糟糕的是，在编写本书时，Eclipse对静态导入的支持还不完善，所以如果你在程序中使用静态导入，Eclipse可能不会自动插入导入语句。对于Eclipse用户而言有一个小技巧：在包语句后面输入一个通配符静态导入（例如，import static org.example.events.Constants.*），使所有语句能够顺利编译。然后可以使用Source > Organize Imports来展开通配符并对导入语句排序。希望在以后的Eclipse版本中这项功能会更直观。

构造函数从第16行开始。DATABASE_NAME是将要使用的数据库的实际文件名（events.db），而DATABASE_VERSION只是我们输入的一个数字。如果这是一个实际的程序，可以在需要对数据库设计进行重大变更时（例如，添加一个新列）增加版本号。

首次尝试访问数据库时，SQLiteOpenHelper将注意到该数据库不存在，并调用onCreate()方法来创建它。在第21行重写了该方法并运行了一条create table SQL语句。这将创建events表和包含该表的events.db数据库文件。

当Android检测到你在引用一个旧数据库时（根据版本号判断），它将调用onUpgrade()方法（第28行）。在这个示例中，我们只是删除了旧表，但是你可以根据需要执行更灵活的操作。例如，可以运行一个alter table SQL命令，将一列添加到现有数据库中。

9.3.2 定义主程序

在Events程序中做的第一次尝试是使用本地SQLite数据库来存储事件，并将这些事件显示为TextView中的一个字符串。按如下方法定义布局文件（layout/main.xml）：

```
Events1/res/layout/main.xml
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView
        android:id="@+id/text"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />
</ScrollView>
```

这段代码使用一个虚构的text ID（代码中的R.id.text）来声明TextView，并使用一个ScrollView对其进行封装，以防太多的事件占满了屏幕。可以在图9-2中查看其外观。



图9-2 该程序的第一个版本在TextView中显示数据库记录

主程序是Events活动中的onCreate()方法。

以下是其中的基本内容：

```
Events1/src/org/example/events/Events.java
package org.example.events;

import static android.provider.BaseColumns._ID;
import static org.example.events.Constants.TABLE_NAME;
import static org.example.events.Constants.TIME;
```

```

import static org.example.events.Constants.TITLE;
import android.app.Activity;
import android.content.ContentValues;
import android.database.Cursor;
10 import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.widget.TextView;

public class Events extends Activity {
15     private EventsData events;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
20         setContentView(R.layout.main);
        events = new EventsData(this);
        try {
            addEvent("Hello, Android!");
            Cursor cursor = getEvents();
25             showEvents(cursor);
        } finally {
            events.close();
        }
    }
30 }

```

在onCreate()内容中的第20行设置了此视图的布局。然后在第21行创建了EventsData类的一个实例并启动了一个try块。如果提前看一下第27行，就会看到我们在finally块中关闭了数据库。所以即使中间发生了错误，数据库仍然会被关闭。

如果没有任何事件，那么events表就没什么引人注意的，所以在第23行上调用了addEvent()方法来向表中添加一个事件。每次运行此程序时，都会获得一个新事件。可以根据需要添加菜单、手势或击键来生成其他事件，但是我将这个练习留给读者完成。

在第24行调用getEvents()方法来获取事件列表，最后在第25行调用showEvents()方法来显示用户列表。

是不是很简单？现在来定义刚才使用的这些新方法。

9.3.3 添加一行

addEvent()方法使用所提供的字符串作为事件标题，向数据库中添加一条新记录。


```
Events1/src/org/example/events/Events.java
```

```
private void addEvent(String string) {
    // Insert a new record into the Events data source.
    // You would do something similar for delete and update.
    SQLiteDatabase db = events.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(TIME, System.currentTimeMillis());
    values.put(TITLE, string);
    db.insertOrThrow(TABLE_NAME, null, values);
}
```

因为需要修改该数据，所以调用`getWritableDatabase()`来获取`events`数据库的一个读/写句柄。该数据库句柄已被缓存，所以可任意次数地调用此方法。

接下来使用当前时间和事件标题填充一个`ContentValues`对象，并将该对象传递给`insertOrThrow()`方法，以执行实际的`insert` SQL语句。此时无需传入记录ID，因为SQLite将生成一个ID并在方法调用中返回该ID。

顾名思义，如果`insertOrThrow()`执行失败，它可以抛出一个`SQLException`类型的异常。无需使用一个`throws`关键字来声明该异常，因为它是一个`RuntimeException`，而不是一个检查异常。但是，如果你喜欢，仍然可以在一个`try/catch`块中处理它，就像处理任何其他异常一样。如果未处理它并且存在一个错误，程序将终止，并将一条回溯信息转储到Android日志中。

默认情况下，只要执行插入，数据库就会立即更新。如果出于某种原因，需要批量或延迟修改，可以访问SQLite网站以了解更多相关的细节。

9.3.4 运行一个查询

`getEvents()`方法执行数据库查询，获得所需的事件列表：

```
Events1/src/org/example/events/Events.java
```

```
private static String[] FROM = { _ID, TIME, TITLE, };
private static String ORDER_BY = TIME + " DESC";
private Cursor getEvents() {
    // Perform a managed query. The Activity will handle closing
    // and re-querying the cursor when needed.
    SQLiteDatabase db = events.getReadableDatabase();
    Cursor cursor = db.query(TABLE_NAME, FROM, null, null, null,
        null, ORDER_BY);
    startManagingCursor(cursor);
    return cursor;
}
```

无需因查询而修改数据库，所以调用`getReadableDatabase()`来获得一个只读

句柄。然后调用`query()`方法来执行实际的`select` SQL语句，`FROM`是想要使用的列构成的数组，`ORDER_BY`告诉SQLite按照从新到旧的顺序返回查询结果。

尽管在这个示例中未使用参数，但是`query()`方法可以使用参数来指定一个`where`子句、一个`group by`子句和一个`having`子句。实际上，`query()`只是为编程人员提供了一种便捷方法。如果你喜欢，可以在一个字符串中自行构建`select`语句，并使用`rawQuery()`方法执行它。无论采用哪种方法，返回值都是一个表示查询结果集的`Cursor`对象。

`Cursor`类似于一个Java `Iterator`或一个JDBC `ResultSet`。在它之上调用方法可获得关于当前行的信息，然后调用另一个方法移到下一行。稍后在显示结果时会看到它的使用方法。

最后一步是调用`startManagingCursor()`，它告诉活动应根据该活动的生命周期来管理光标的生命周期。例如，当活动被暂停时，它将自动停用光标，然后在活动重启时重新查询该光标。当活动终止时，所有托管的光标都将关闭。

9.3.5 显示查询结果

需要定义的最后一个是`showEvents()`。此函数接受一个`Cursor`作为输入并格式化输出，以便用户能够理解输出的内容。

```

Events1/src/org/example/events/Events.java
Line 1 private void showEvents(Cursor cursor) {
-     // Stuff them all into a big string
-     StringBuilder builder = new StringBuilder(
-         "Saved events:\n");
5     while (cursor.moveToNext()) {
-         // Could use getColumnIndexOrThrow() to get indexes
-         long id = cursor.getLong(0);
-         long time = cursor.getLong(1);
-         String title = cursor.getString(2);
10     builder.append(id).append(": ");
-         builder.append(time).append(": ");
-         builder.append(title).append("\n");
-     }
-     // Display on the screen
15     TextView text = (TextView) findViewById(R.id.text);
-     text.setText(builder);
- }

```

在这个版本的Events应用程序中，将创建一个很大的字符串（参见第3行）来保

存所有事件条目，各个条目通过换行符来分隔。这不是推荐的方法，但是现在可以使用它。

第5行调用`Cursor.moveToNext()`方法前进到数据集中的下一行。第一次获得一个`Cursor`时，它位于第一条记录之前，所以调用`moveToNext()`可找到第一条记录。继续运行循环，直到`moveToNext()`返回`false`，这表示没有其他行了。

在循环内部（第7行），调用`getLong()`和`getString()`从所需的列中获取数据，然后将这些值追加到字符串后面（第10行）。`Cursor`上还有另一个方法`getColumnIndexOrThrow()`，可以使用它来获取列索引号（传递给`getLong()`和`getString()`的值0、1和2）。但是，这个方法有点慢，所以如果你需要使用它，应该在循环外部调用它并自己记住索引。

处理完所有列之后，在`layout/main.xml`中查找`TextView`，并向其中填充前面提到的较大字符串（第15行）。

如果现在就运行该示例，看到的结果应该与图9-2类似。恭喜你完成了第一个Android数据库程序！但是，该程序还有很大的改进空间。

如果列表中有数千个或者甚至数百万个事件会怎样？程序的运行速度将会变得非常慢，并且在尝试构建一个字符串以保存所有这些事件时可能会耗尽所有内存。如果希望让用户选择一个事件并对该事件执行一些操作，又该怎么办呢？所有信息都在一个字符串中是不可能实现此目的的。幸运的是，Android提供了一种更好的方法：数据绑定。

4 数据绑定

有了数据绑定，只需几行代码就可将模型（数据）连接到视图。为了演示数据绑定，我们将修改Events示例程序，使用一个已绑定到数据库查询结果的`ListView`。首先，需要让`Event`类扩展`ListActivity`，而不是扩展`Activity`。

```
Eventsv2/src/org/example/events/Events.java
```

```
public class Events extends ListActivity {
    // ...
}
```

接下来需要更改事件在Events中的显示方式。`showEvents()`方法如下：

```
Eventsv2/src/org/example/events/Events.java
private static int[] TO = { R.id.rowid, R.id.time, R.id.title, };
private void showEvents(Cursor cursor) {
    // Set up data binding
    SimpleCursorAdapter adapter = new SimpleCursorAdapter(this,
        R.layout.item, cursor, FROM, TO);
    setListAdapter(adapter);
}
```

注意，这段代码比以前短得多（现在只有2行，而以前有10行）。第一行为Cursor创建一个SimpleCursorAdapter，第二行告诉ListActivity使用新的适配器。该适配器充当着一种媒介，连接视图与其数据源。

如果回想一下，就会发现我们早在7.4节的Translate示例程序（参见Translate.setAdapters()）内就首次使用了适配器。在那个示例中使用了ArrayAdapter，因为数据源是在XML中定义的一个数组。对于现在的示例，我们使用SimpleCursorAdapter，因为数据源是一个来自于数据库查询的Cursor对象。

SimpleCursorAdapter的构造函数接受下面5个参数。

- context：对当前Activity的引用。
- layout：一个资源，它定义一个列表条目的视图。
- cursor：数据集光标。
- from：一组列名称，数据来源于这些列。
- to：视图列表，这是数据的目的地。

列表条目的布局在layout/item.xml中定义。注意在TO数组中引用的行ID、时间和标题视图的定义。

```
Eventsv2/res/layout/item.xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal"
    android:padding="10sp">
    <TextView
        android:id="@+id/rowid"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <TextView
        android:id="@+id/rowidcolon"
        android:layout_width="wrap_content"
```

```

        android:layout_height="wrap_content"
        android:text=": "
        android:layout_toRightOf="@id/rowid" />
<TextView
    android:id="@+id/time"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_toRightOf="@id/rowidcolon" />
<TextView
    android:id="@+id/timecolon"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text=": "
    android:layout_toRightOf="@id/time" />
<TextView
    android:id="@+id/title"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:ellipsize="end"
    android:singleLine="true"
    android:textStyle="italic"
    android:layout_toRightOf="@id/timecolon" />
</RelativeLayout>

```

这看起来比实际的要更复杂。我们所做的只是将ID、时间和标题放到一行上，并将各个字段用冒号隔开。我添加了一些装饰内容和格式，使其看起来更漂亮。

最后，需要在layout/main.xml中更改活动本身的布局。以下是新版本main文件：

Eventsv2/res/layout/main.xml

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <!-- Note built-in ids for 'list' and 'empty' -->
    <ListView
        android:id="@android:id/list"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
    <TextView
        android:id="@android:id/empty"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/empty" />
</LinearLayout>

```

由于该活动扩展了ListActivity，所以Android在布局文件中查找两个特殊ID。如果列表中有内容，那么将显示android:id/list视图，否则将显示android:

id/empty视图。因此,如果列表中没有条目,也不会显示空白屏幕,用户将看到消息“No events!”

最终结果请参见图9-3。作为练习,读者可以思考一下,如果有一个真正的列表,那么如何增强该应用程序。例如,当用户选择某个事件时,你可以打开一个详细信息查看器,将事件通过邮件发送给技术支持人员,或者可能从数据库中删除选定的事件及其下面的所有事件。



图9-3 这个程序版本使用了一个ListActivity和数据绑定

这个示例仍然存在一些问题。其他应用程序都不能向事件数据库添加内容,甚至都无法查看这些事件!对于这一点,需要使用Android中的ContentProvider。

9.5 使用 ContentProvider

在Android安全模型(参见2.5节中的内容)中,一个应用程序编写的文件无法被其他任何应用程序所读写。每个程序都有自己的Linux用户ID和数据目录(/data/data/包名),以及其受保护的内存空间。Android程序可通过下面两种方式进行彼此间的通信。

- IPC (Inter-Process Communication, 进程间通信): 一个进程使用AIDL (Android Interface Definition Language, 接口定义语言) 和IBinder接口声明一个任意的API。调用该API时,将在进程间安全且有效地对参数进行编组。

这项先进技术用于对后台Service线程进行远程过程调用。

- **ContentProvider**: 进程在系统中将它们本身注册为某些数据类型的提供者。请求该信息时, Android就会通过一个固定的API调用这些进程, 以它们认为适合的方式查询或修改内容。在Events示例程序中使用的就是此项技术。

ContentProvider管理的任何信息部分都通过一个URI来寻址, 这个URI类似于以下形式:

```
content://authority/path/id
```

其中:

- **content://**是标准要求的前缀;
- **authority**是提供者的名称, 建议你使用完全限定包名称, 避免出现名称冲突;
- **path**是提供者内部的一个虚拟目录, 用于标识被请求的数据类型;
- **id**是被请求的特定记录的主键, 要请求获得具有特定类型的所有记录, 可以省略此参数以及后面的斜杠。

Android已经内置提供了几个提供者, 包括:^①

- **content://browser**;
- **content://contacts**;
- **content://media**;
- **content://settings**。

为了演示如何使用ContentProvider, 将Events示例程序改为使用ContentProvider。对于Events提供者, 以下都是有效的URI:

```
content://org.example.events/events/3 -- single event with _id=3
content://org.example.events/events -- all events
```

首先需要向Constants.java添加两个常量:

^① 这里没有使用字符串, 而使用了已归档的常量, 如Browser.BOOKMARKS_URI。注意, 访问一些提供者需要在manifest文件中请求获得其他权限。


```
Eventsv3/src/org/example/events/Constants.java
```

```
public static final String AUTHORITY = "org.example.events";
public static final Uri CONTENT_URI = Uri.parse("content://"
    + AUTHORITY + "/" + TABLE_NAME);
```

无需更改布局文件(main.xml和item.xml),因此接下来可以对Events类稍作修改。

9.5.1 更改主程序

主程序(Events.onCreate()方法)实际上稍微简单一些,因为不需要跟踪数据库对象:

```
Eventsv3/src/org/example/events/Events.java
```

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    addEvent("Hello, Android!");
    Cursor cursor = getEvents();
    showEvents(cursor);
}
```

不需要try/finally块,并且可以删除对EventData的引用。

9.5.2 添加一行

addEvent()方法中有两行发生了变化。以下是该方法的新版本:

```
Eventsv3/src/org/example/events/Events.java
```

```
private void addEvent(String string) {
    // Insert a new record into the Events data source.
    // You would do something similar for delete and update.
    ContentValues values = new ContentValues();
    values.put(TIME, System.currentTimeMillis());
    values.put(TITLE, string);
    getContentResolver().insert(CONTENT_URI, values);
}
```

没有了getWritableDatabase()的调用,对insertOrThrow()的调用替换为getContentResolver().insert()。我们使用了一个内容URI,而不是用数据库句柄。

9.5.3 运行一个查询

使用ContentProvider时，getEvents()方法也简化了：

```
Eventsv3/src/org/example/events/Events.java
private Cursor getEvents() {
    // Perform a managed query. The Activity will handle closing
    // and re-querying the cursor when needed.
    return managedQuery(CONTENT_URI, FROM, null, null, ORDER_BY);
}
```

此处使用Activity.managedQuery()方法，将内容URI、感兴趣的列表和应该使用的排序顺序传递给该方法。

通过删除对数据库的所有引用，可将Events客户端从Events数据提供者中分离出来。现在客户端更简单，但必须实现以前从未实现过的一个新部分。

9.6 实现 ContentProvider

ContentProvider是一个类似于Activity的高级对象，需要向系统进行声明。因此，实现ContentProvider的第一步是将其添加到AndroidManifest.xml文件中的<activity>标签之前（作为<application>的子标签）：

```
Eventsv3/AndroidManifest.xml
<provider android:name="EventsProvider"
    android:authorities="org.example.events" />
```

android:name是类名，android:authorities是在内容URI中使用的字符串。

接下来创建EventsProvider类，该类必须扩展ContentProvider。以下是基本内容：

```
Eventsv3/src/org/example/events/EventsProvider.java
package org.example.events;

import static android.provider.BaseColumns._ID;
import static org.example.events.Constants.AUTHORITY;
import static org.example.events.Constants.CONTENT_URI;
import static org.example.events.Constants.TABLE_NAME;
import android.content.ContentProvider;
import android.content.ContentUris;
import android.content.ContentValues;
import android.content.UriMatcher;
import android.database.Cursor;
```

```

import android.database.sqlite.SQLiteDatabase;
import android.net.Uri;
import android.text.TextUtils;

public class EventsProvider extends ContentProvider {
    private static final int EVENTS = 1;
    private static final int EVENTS_ID = 2;

    /** The MIME type of a directory of events */
    private static final String CONTENT_TYPE
        = "vnd.android.cursor.dir/vnd.example.event";

    /** The MIME type of a single event */
    private static final String CONTENT_ITEM_TYPE
        = "vnd.android.cursor.item/vnd.example.event";
    private EventsData events;
    private UriMatcher uriMatcher;
    // ...
}

```

根据约定，使用 MIME^① 类型中的 `vnd.example` 而不是 `org.example`。EventsProvider 处理下面两种类型的数据。

- EVENTS (MIME 类型 CONTENT_TYPE): 一个事件目录或列表。
- EVENTS_ID (MIME 类型 CONTENT_ITEM_TYPE): 一个事件。

在 URI 方面，区别在于第一种类型不会指定 ID，但是第二种类型会指定。使用 Android 的 UriMatcher 类解析 URI 并告诉我们客户端指定了哪一个 ID。重用本章前面介绍的 EventsData 类，以管理提供者内的实际数据库。

由于篇幅的关系，我们打算在这里展示该类的剩余部分，但是可以从本书网站上下载完整的代码。Events 示例程序的所有 3 个版本都可以在源代码 .zip 文件中找到。

Events 示例程序的最后一个版本与前一个版本在外表方面极其相似（参见图 9-3）。但是在内部，你现在有了一个可被系统中其他应用程序使用的事件存储框架，即使是其他开发人员编写的应用程序也可以使用它。

9.7 快速阅读指南

本章介绍了如何在 Android SQL 数据库中存储数据。如果你想使用 SQL 实现更

① MIME (Multipurpose Internet Mail Extensions, 多用途因特网邮件扩展) 是一种因特网标准，用于描述任何内容种类的类型。

多功能，则需要了解更多有关语句和表达式的内容，而不仅仅是此处介绍的内容。由Jonathan Gennick编写的*SQL Pocket Guide*[Gen06]或由Mike Owens编写的*The Definitive Guide to SQLite*[Owe06]等都是不错的著作，但是要记住，不同数据库之间使用的SQL语法和功能会存在一些差异。

在Android上存储数据的另一个选择是db4o^①。这个库比SQLite更大并且使用不同的许可（GNU公共许可），但它是免费的，并且更容易使用，特别适合不了解SQL的读者。

本章介绍的SimpleCursorAdapter可以进行自定义，以显示除文本之外的更多信息。例如，可以显示评定的星级、密集的小图形（sparkline）或基于Cursor中数据的其他视图。在SimpleCursorAdapter文档中查找ViewBinder，以了解更多的信息^②。

接下来是完全不同的内容……下一章将介绍如何使用OpenGL实现3D图形。

① 参见网页<http://www.db4o.com/android>。

② 参见网页<http://androidappdoes.appspot.com/reference/android/widget/SimpleCursorAdapter.html>。

10

利用OpenGL实现3D图形

—— 二维图形对于大部分程序来说已经足够，但有时需要二维图形所无法实现的额外深度感、交互感或真实感。针对这些情形，Android提供了一个基于OpenGL ES标准的三维图形库。本章将介绍3D概念，并构建一个可使用OpenGL的示例程序。

10.1 理解 3D 图形

我们生活的世界是三维的，但我们也常以二维方式来观察它。在我们看电视或查看书中的图片时，3D图像就变成平面的了，或者说被投射到了2D表面（电视面板或书页）上。

下面做一个简单的实验：闭上一只眼睛，然后向窗外看。会看到什么呢？阳光照到窗外的物体上并反射回来，然后穿过窗户，照射到人眼上，所以我们能够看到物体。在图形术语中，窗外的场景被投射到窗户（或视区 [viewport]）上。如果将窗户更换为高质量的照片，那么只要你不移动，看到的景象就是一样的。

根据眼睛与窗户之间的距离和窗户的大小，可以看到的外面世界多少是不同的。这称为视野。如果从眼睛到窗户的4个角各连一条线并延长出去，将会获得如图10-1所示的金字塔。这称为视景体 (view frustum)（在拉丁文中表示一个“碎片”）。从性能方面考虑，视景体通常限制在近处和远处的两个剪切平面内。你可以看到视景体内的所有物体，但看不到其外部的物体。

在3D计算机图形领域，计算机屏幕作为视区。你的工作是让用户将其窗口想象成玻璃后面的另一个世界。OpenGL图形库就是用于完成此任务的API。

10.2 OpenGL 简介

OpenGL^①由Silicon Graphics于1992年开发。它为编程人员提供了统一接口，以便充分利用任何制造商提供的硬件。OpenGL的核心实现了视区和光照等我们所熟知的概念，并试图向开发人员隐藏大部分硬件层。

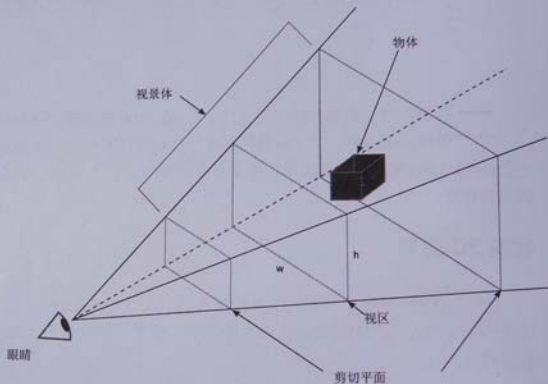


图10-1 查看一个三维场景

由于OpenGL专为工作站设计，它太大了，无法安装到移动设备上。所以，Android实现了OpenGL的一个子集，称为OpenGL ES（OpenGL for Embedded Systems，嵌入式系统OpenGL）^②。这项标准由Khronos Group创建，后者是由英特尔、AMD、Nvidia、诺基亚、三星和索尼等公司组成的行业联盟。这个库（经过细微改动之后）现在可应用于很多主流移动平台上，包括Android、Symbian和iPhone。

每种语言都有其自己的OpenGL ES语言绑定，Java语言也不例外。Java的语言绑定是由JSR（Java Specification Request，Java规范申请）239^③定义的。Android尽

① 参见网页<http://www.opengl.org>。

② 参见网页<http://www.khronos.org/opengles>。

③ 参见网页<http://jcp.org/en/jsr/detail?id=239>。

可能准确地实现了此标准，所以你可以参阅各种介绍JSR 239和OpenGL ES的书籍和文档，了解对其所有类和方法的详细介绍。

现在看一下如何在Android中创建简单的OpenGL程序。

谢谢你，John Carmack

实践证明，OpenGL非常成功，但是它曾经差一点就一命呜呼了。1995年，微软公司推出了一项名为Direct3D的竞争技术。由于微软公司处于主导的市场地位和巨大的研发投资，Direct3D大有后来居上并成为行业事实标准的趋势。但是id Software的联合创始人John Carmack拒绝妥协。他推出了广为流行的Doom和Quake游戏，几乎凭一己之力迫使硬件制造商不断更新他们的PC OpenGL设备驱动程序。现在的Linux、Mac OS X和移动设备用户都应该感激John和id Software推动了OpenGL标准的长足发展。

10.3 构建一个 OpenGL 程序

首先按照1.2节中操作步骤创建一个新的“Hello, Android”项目，但是这次在New Android Project对话框中提供以下参数：

```
Project name: OpenGL
Package name: org.example.opengl
Activity name: OpenGL
Application name: OpenGL
```



小乔爱问……

是否每部手机都拥有3D功能

可以说是也可以说不是。一些运行Android的低端设备可能没有实际的3D硬件。但是这些设备仍然具有OpenGL编程接口。所有3D功能都用软件的方式进行模拟。编写的程序仍然能够运行，但运行速度比硬件加速设备要慢很多。为此，为用户提供某些选项来关闭某些绘制起来很费时，但对程序而言又不是必需的细节和特效是一个不错的想法。有了这种方式，如果用户在较慢的设备上运行该程序，他们可以禁用一些视觉效果来获得更高的性能。

这将创建 OpenGL.java 来包含主要活动。编辑该程序，并将它改为引用一个名为 GLView 的自定义视图，如下所示（记住，可以在 <http://pragprog.com/titles/eband> 上找到该代码的最新版本，或者如果正在阅读 PDF，只需单击代码前的文件名即可）。

```
OpenGL/src/org/example/opengl/OpenGL.java
```

```
package org.example.opengl;

import android.app.Activity;
import android.os.Bundle;

public class OpenGL extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new GLView(this));
    }
}
```

不需要布局资源（res/layout/main.xml），所以可将其删除。现在定义我们的自定义视图类：

```
OpenGL/src/org/example/opengl/GLView.java
```

```
package org.example.opengl;

import android.content.Context;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
class GLView extends SurfaceView implements SurfaceHolder.Callback {
    GLView(Context context) {
        super(context);

        // Install a SurfaceHolder.Callback so we get notified when
        // the underlying surface is created and destroyed
        getHolder().addCallback(this);

        // Use hardware acceleration if available
        getHolder().setType(SurfaceHolder.SURFACE_TYPE_GPU);
    }

    public void surfaceCreated(SurfaceHolder holder) {
    }

    public void surfaceDestroyed(SurfaceHolder holder) {
    }

    public void surfaceChanged(SurfaceHolder holder, int format,
        int w, int h) {
        // TODO: handle window size changes
    }
}
```

这个例子中有几个需要你了解的新类。

- **SurfaceView**: 一种用于3D图形的特殊视图类型。可以扩展此视图，然后将其用于任何使用OpenGL的视图。
- **Surface**: 一个绘图区域，与Canvas类似（参见4.1节），但它是通过3D硬件（如果存在）实现的。当OpenGL活动在前台运行时将创建Surface，当活动退出或者其他活动变为前台运行时将销毁Surface。
- **SurfaceHolder**: 该类的一个实例始终存在，即使是它所保持的Surface被销毁时也是如此。
- **SurfaceHolder.Callback**: 一个SurfaceView会实现此接口，当创建、销毁视图的Surface或调整其大小时，OpenGL将通知这个SurfaceView。

下一节将介绍如何用实色填充屏幕。

10.4 管理线程

在4.2节已经看到，Android 2D库在需要重新绘制屏幕的部分区域时调用视图的onDraw()方法。OpenGL不会这样做。

在OpenGL中，你可创建自己专用于绘制显示屏幕的线程。现在创建一个这样的线程：

```
OpenGL/src/org/example/opengl/GLView.java
private GLThread glThread;

public void surfaceCreated(SurfaceHolder holder) {
    // The Surface has been created so start our drawing thread
    glThread = new GLThread(this);
    glThread.start();
}

public void surfaceDestroyed(SurfaceHolder holder) {
    // Stop our drawing thread. The Surface will be destroyed
    // when we return
    glThread.requestExitAndWait();
    glThread = null;
}
```

在GLView类中，为glThread线程声明一个变量，这个变量的类型是自定义的

GLThread类。创建OpenGL表面时，将启动一个新线程来处理其绘制操作，当表面将被销毁时，将终止该线程。

现在看一下如何定义GLThread类。首先看一下基本内容：

OpenGL/src/org/example/opengl/GLThread.java

```
package org.example.opengl;

import javax.microedition.khronos.egl.EGL10;
import javax.microedition.khronos.egl.EGL11;
import javax.microedition.khronos.egl.EGLConfig;
import javax.microedition.khronos.egl.EGLContext;
import javax.microedition.khronos.egl.EGLDisplay;
import javax.microedition.khronos.egl.EGLSurface;
import javax.microedition.khronos.opengles.GL10;

import android.app.Activity;
import android.content.Context;
import android.opengl.GLU;

class GLThread extends Thread {
    private final GLView view;
    private boolean done = false;

    GLThread(GLView view) {
        this.view = view;
    }
    @Override
    public void run() {
        // Initialize OpenGL...
        // Loop until asked to quit
        while (!done) {
            // Draw a single frame here...
        }
    }

    public void requestExitAndWait() {
        // Tell the thread to quit
        done = true;
        try {
            join();
        } catch (InterruptedException ex) {
            // Ignore
        }
    }
}
```

这是一个普通的Java线程编码，没有什么特别之处。在销毁表面时将调用requestExitAndWait()方法。我们只是设置一个标记，然后等待线程注意到该标记并关闭自己，而不会尝试销毁线程本身（这在Java中不安全）。这个操作的时间应该不超过几分之一秒。

现在编写run()方法:

OpenGL/src/org/example/opengl/GLThread.java

```

line 1 @Override
public void run() {
    // Initialize OpenGL...
    EGL10 egl = (EGL10) EGLContext.getEGL();
    5 EGLDisplay display = egl.eglGetDisplay(EGL10.EGL_DEFAULT_DISPLAY);

    int[] version = new int[2];
    egl.eglInitialize(display, version);

    10 int[] configSpec = { EGL10.EGL_RED_SIZE, 5,
        EGL10.EGL_GREEN_SIZE, 6, EGL10.EGL_BLUE_SIZE, 5,
        EGL10.EGL_DEPTH_SIZE, 16, EGL10.EGL_NONE };

    EGLConfig[] configs = new EGLConfig[1];
    15 int[] numConfig = new int[1];
    egl.eglChooseConfig(display, configSpec, configs, 1,
        numConfig);
    EGLConfig config = configs[0];

    20 EGLContext glc = egl.eglCreateContext(display, config,
        EGL10.EGL_NO_CONTEXT, null);
    EGLSurface surface = egl.eglCreateWindowSurface(display,
        config, view.getHolder(), null);
    25 egl.eglMakeCurrent(display, surface, surface, glc);

    GL10 gl = (GL10) (glc.getGL());
    init(gl);

    30 // Loop until asked to quit
    while (!done) {
        // Draw a single frame here...
        drawFrame(gl);
        egl.eglSwapBuffers(display, surface);

    35 // Error handling
        if (egl.eglGetError() == EGL11.EGL_CONTEXT_LOST) {
            Context c = view.getContext();
            if (c instanceof Activity) {
                ((Activity) c).finish();
            }
        }
    }

    45 // Free OpenGL resources
    egl.eglMakeCurrent(display, EGL10.EGL_NO_SURFACE,
        EGL10.EGL_NO_SURFACE, EGL10.EGL_NO_CONTEXT);
    egl.eglDestroySurface(display, surface);
    egl.eglDestroyContext(display, glc);
    50 egl.eglTerminate(display);
}

```

这段代码做的第一件事，就是获取EGL10对象的一个句柄，如第4行所示。接下来获得一个Display句柄（第5行）并初始化它（第8行），然后要求OpenGL找到一个与特定的颜色深度需求匹配的配置（第10行）。此处要求一个16位的颜色深度，其中5位为红色，6位为绿色，5位为蓝色，你自己的应用程序可能具有其他需求^①。

找到此配置后，可以创建OpenGL上下文（如第20行所示），创建一个表面（如第23行所示），以及设置显示、表面和上下文的当前值（如第25行所示）。

第27行使用getGLC()函数返回实际的OpenGL接口。我们将其转换为GL10，以便调用OpenGL ES 1.0方法。可能你已经注意到，尽管Android具有一些来自1.1版和OpenGL扩展的方法，但并没有完全实现这些方法，所以暂时应该避免使用它们。除此之外，GL10提供了可能需要的一切内容。

如何选择1.x版本

OpenGL ES 1.0基于完整的OpenGL 1.3版，ES 1.1基于OpenGL 1.5。JSR 239有两个版本：原始的1.0版本和一个维护发行版1.0.1。还有一些OpenGL ES扩展我没有提及。Android实现JSR 239 1.0.1，OpenGL ES实现1.0。预计以后Android将会支持OpenGL ES 1.1和更高版本。

不要被所有这些1.x版本弄得不知所措。这些标准已经演化了好多年。现在，它们已经非常稳定且很容易实现。

在第28行中调用init()方法来初始化OpenGL选项之后，开始运行主循环。线程将在这里不断循环，持续绘制每一帧，直到done标记变为true为止。

OpenGL对象会消耗大量的资源，因此在run()方法的末尾（从第46行开始）显式地销毁这些对象，以释放资源。

现在看一下init()方法：

OpenGL/src/org/example/opengl/GLThread.java

```
Line 1 private void init(GL10 gl) {
-      // Define the view frustrum
-      gl.glViewport(0, 0, view.getWidth(), view.getHeight());
-      gl.glMatrixMode(GL10.GL_PROJECTION);
5      gl.glLoadIdentity();
-      float ratio = (float) view.getWidth() / view.getHeight();
```

① 在0.9_beta版中，初始化OpenGL的Android调用有点复杂。希望在未来的版本中，谷歌公司会提供一些帮助函数来再次简化它，但是现在只需注意这一点并复制这些代码即可。

```

11 GLU.gluPerspective(g1, 45.0f, ratio, 1, 100f);
12
13 // Set up any other options we need
14 g1.glEnable(GL10.GL_DEPTH_TEST);
15 g1.glDepthFunc(GL10.GL_LEQUAL);
16 g1.glEnableClientState(GL10.GL_VERTEX_ARRAY);
17 // Optional: disable dither to boost performance
18 // g1.glDisable(GL10.GL_DITHER);
19 }

```

`init()`方法负责设置我们的视景体和一些OpenGL选项。注意，第7行调用了`GLU.gluPerspective()`帮助器函数。最后两个参数是眼睛与近处和远处剪切平面之间的距离（参见图10-1）。

在第10行设置了两个OpenGL选项。OpenGL拥有十多个选项，你可以使用`glEnable()`和`glDisable()`方法来启用或禁用它们。最常用的选项见表10-1。

表10-1 OpenGL的常用选项

选 项	描 述
<code>GL_BLEND</code>	混合传入的颜色值和颜色缓冲区中已有的值
<code>GL_CULL_FACE</code>	根据窗口坐标中多边形的转动方向（顺时针或逆时针）来忽略多边形的某些面。这是消除多边形后面内容绘制的一种简便方法
<code>GL_DEPTH_TEST</code>	进行深度比较，并更新深度缓冲区。如果某些像素距离已经绘制的像素很远，则这些像素将被忽略
<code>GL_LIGHTi</code>	计算对象的亮度和颜色时将光线数 <i>i</i> 包括在内
<code>GL_LIGHTING</code>	打开照明和材料计算
<code>GL_LINE_SMOOTH</code>	绘制平滑的线（没有锯齿的线）
<code>GL_MULTISAMPLE</code>	针对反锯齿和其他效果执行多次采样
<code>GL_POINT_SMOOTH</code>	绘制无锯齿的点
<code>GL_TEXTURE_2D</code>	使用纹理绘制表面

所有这些选项在默认情况下都是关闭的，但`GL_DITHER`和`GL_MULTISAMPLE`除外。注意，启用每一个选项都会对性能产生影响。

现在该绘制一些内容了。主循环中的每次迭代都会调用`drawFrame()`方法。

OpenGL/src/org/example/opengl/GlThread.java

```

private void drawFrame(GL10 g1) {
    // Clear the screen to black
    g1.glClear(GL10.GL_COLOR_BUFFER_BIT
        | GL10.GL_DEPTH_BUFFER_BIT);

    // Position model so we can see it

```

```
gl.glMatrixMode(GL10.GL_MODELVIEW);  
gl.glLoadIdentity();  
gl.glTranslatef(0, 0, -3.0f);  
  
// Other drawing commands go here...  
}
```

现在所做的只是将屏幕设为黑色。要清空颜色和深度缓冲区。始终记住这两项都要清除，否则留下的上一帧深度信息会导致一些非常奇怪的结果。

如果现在运行程序，将得到如图10-2所示的效果。你也许认为在循环中反复绘制同一个黑色屏幕比较愚蠢，不错。但是在稍后讨论动画时需要使用循环，所以现在先忍耐一下。



图10-2 获得一个黑色屏幕比较烦人

要想继续绘制一些更加有趣的内容。首先需要准确定义要绘制的内容（模型）。

10.5 构建一个模型

根据要绘制对象的复杂性，通常需要使用图形设计工具创建它们，然后将其导入到程序中。在这个示例中，仅在代码中定义一个简单模型：一个立方体。

```

1 package org.example.opengl;
2
3 import java.nio.ByteBuffer;
4 import java.nio.ByteOrder;
5 import java.nio.IntBuffer;
6
7 import javax.microedition.khronos.opengles.GL10;
8
9 import android.content.Context;
10 import android.graphics.Bitmap;
11 import android.graphics.BitmapFactory;
12
13 class GLCube {
14     private final IntBuffer mVertexBuffer;
15     public GLCube() {
16         int one = 65536;
17         int half = one / 2;
18         int vertices[] = {
19             // FRONT
20             -half, -half, half, half, -half, half,
21             -half, half, half, half, half, half,
22             // BACK
23             -half, -half, -half, -half, half, -half,
24             half, -half, -half, half, half, -half,
25             // LEFT
26             -half, -half, half, -half, half, half,
27             -half, -half, -half, -half, half, -half,
28             // RIGHT
29             half, -half, -half, half, half, -half,
30             half, -half, half, half, half, half,
31             // TOP
32             -half, half, half, half, half, half,
33             -half, half, -half, half, half, -half,
34             // BOTTOM
35             -half, -half, half, -half, -half, -half,
36             -half, -half, half, half, -half, -half, };
37         // Buffers to be passed to gl*Pointer() functions must be
38         // direct, i.e., they must be placed on the native heap
39         // where the garbage collector cannot move them.
40         //
41         // Buffers with multi-byte data types (e.g., short, int,
42         // float) must have their byte order set to native order
43         ByteBuffer vbb = ByteBuffer.allocateDirect(vertices.length * 4);
44         vbb.order(ByteOrder.nativeOrder());
45         mVertexBuffer = vbb.asIntBuffer();
46         mVertexBuffer.put(vertices);
47         mVertexBuffer.position(0);
48     }
49
50     public void draw(GL10 gl) {
51         gl.glVertexPointer(3, GL10.GL_FIXED, 0, mVertexBuffer);
52
53         gl.glColor4f(1, 1, 1, 1);

```

```

-         gl.glNormal3f(0, 0, 1);
56      gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 0, 4);
-         gl.glNormal3f(0, 0, -1);
-         gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 4, 4);
-
-         gl.glColor4f(1, 1, 1, 1);
60      gl.glNormal3f(-1, 0, 0);
-         gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 8, 4);
-         gl.glNormal3f(1, 0, 0);
-         gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 12, 4);
-
66      gl.glColor4f(1, 1, 1, 1);
-         gl.glNormal3f(0, 1, 0);
-         gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 16, 4);
-         gl.glNormal3f(0, -1, 0);
-         gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 20, 4);
70  }
-  }

```

定点与浮点

OpenGL ES 为其所有方法都提供了定点（整数）和浮点接口。定点方法以字母 **x** 结束，浮点方法以字母 **f** 结束。例如，可以使用 `glColor4x()` 和 `glColor4f()` 来设置某种颜色的 4 个组成部分。

定点数的范围是 2^{32} ，即 65 536。因此，定点数 32 768 等于 0.5f。另一种表示方式是，整数部分使用 4 字节 `int` 的最高两个有效字节，而小数部分使用最低两个有效字节。这与本机 Android 2D 库使用整数的方式有很大差别，所以使用时应加倍小心。

在这种简单的示例中，使用定点还是使用浮点算法并不重要，因此我根据需要交替使用它们。但是请记住，一些 Android 设备没有浮点硬件，因此定点算法可能会快得多。建议首先使用浮点编写代码，因为这样更容易编程，然后如果需要，在以后可使用定点优化较慢的部分。

第 18 行的 `vertices` 数组在定点模型坐标（参见上文“定点与浮点”）中定义立方体的每个角。立方体的每一面都是一个正方形，由两个三角形组成。我们使用一个常见的 OpenGL 绘图模式：三角形条带（`triangle strip`）。在这个模式下，我指定两个起点，然后每个后续的点与前两个起点一起定义一个三角形。这是生成大量几何形状来消耗图形硬件的一种快速方法。

注意，每个点拥有 3 个坐标（`x`、`y` 和 `z`）。`x` 和 `y` 轴分别指向右侧和上侧，`z` 轴指向屏幕外朝着眼睛的方向。

在绘图方法中（第50行），使用在构造函数中创建的顶点缓冲区并绘制6个不同方向的三角形（用于立方体的6个面）。在实际的程序中，可能需要将这些调用组合到一个或两个条带中，因为执行的OpenGL调用越少，程序运行速度就越快。

现在让我们在GLThread中使用新类：

```
OpenGL/src/org/example/opengl/GLThread.java
private final GLCube cube = new GLCube();
private void drawFrame(GL10 gl) {
    // ... Draw the model
    cube.draw(gl);
}
```

现在如果运行该程序，将会看到如图10-3所示的结果。这个效果可比只有黑色好多了。

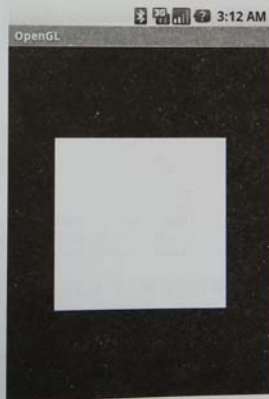


图10-3 绘制一个没有阴影的立方体

10.6 光线、相机……

现实生活中有很多光源，如太阳、汽车前灯、火炬或炙热的火山熔岩。OpenGL支持在场景中最多定义8种光源。照明包括两部分：光线和被照射的对象。下面首

先看看光线。

所有3D图形库都支持下面3种照明类型。

- **环境光**：一种普通的光线，光线会照亮整个场景，即使是对对象背对着光线也可以。拥有少量环境光很重要，这样可以呈现某些细节，即使是阴影中的细节也可以。
- **散射光**：柔和的方向性光线，例如荧光板上发出的光线就是这种散射光。场景中的大部分光线通常来源于散射光源。
- **镜面高光**：耀眼的光线，通常来源于明亮的点光源。与有光泽的材料结合使用时，这种光会带来高光（很明亮的样子）效果，增加场景的真实感。

一个光源可以提供所有3种类型的光线。这些值被代入到一个照明方程中，以确定屏幕上每个像素的颜色和亮度。

照明在GLThread.init()方法中定义：

OpenGL/src/org/example/opengl/GLThread.java

```
// Define the lighting
float lightAmbient[] = new float[] { 0.2f, 0.2f, 0.2f, 1 };
float lightDiffuse[] = new float[] { 1, 1, 1, 1 };
float[] lightPos = new float[] { 1, 1, 1, 1 };

gl.glEnable(GL10.GL_LIGHTING);
gl.glEnable(GL10.GL_LIGHT0);
gl.glLightfv(GL10.GL_LIGHT0, GL10.GL_AMBIENT, lightAmbient, 0);
gl.glLightfv(GL10.GL_LIGHT0, GL10.GL_DIFFUSE, lightDiffuse, 0);
gl.glLightfv(GL10.GL_LIGHT0, GL10.GL_POSITION, lightPos, 0);
```

在这段代码中，我们在位置(1, 1, 1)定义一个光源。它发出的是白色的全方向光，由明亮的散射光和较暗的环境光组成。在本示例中没有使用镜面高光。

接下来需要告诉OpenGL立方体的制作材料。不同材料的光线反射情况也不同，如金属、塑料或纸张。为了在OpenGL模拟这种特征，在init()方法中添加以下代码来定义材料对以下3种类型的光有何反应：环境光、散射光和镜面高光。

OpenGL/src/org/example/opengl/GLThread.java

```
// What is the cube made of?
float matAmbient[] = new float[] { 1, 1, 1, 1 };
float matDiffuse[] = new float[] { 1, 1, 1, 1 };
gl.glMaterialfv(GL10.GL_FRONT_AND_BACK, GL10.GL_AMBIENT,
    matAmbient, 0);
gl.glMaterialfv(GL10.GL_FRONT_AND_BACK, GL10.GL_DIFFUSE,
    matDiffuse, 0);
```

对象看起来会变得不太明亮，就像用纸做的一样（参见图10-4）。立方体的右上角靠近光源，所以显得更亮。

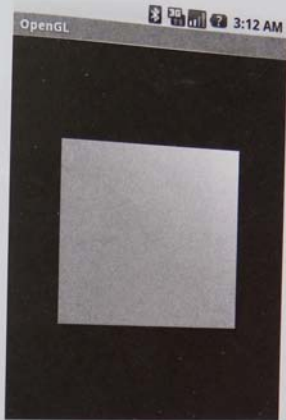


图10-4 照亮场景

10.7 动作

到目前为止，立方体还是静止的，没有任何移动。这样不太有趣，所以下面让它动起来。为此，需要对GLThread中的init()和drawFrame()方法进行一些修改。

OpenGL/src/org/example/opengl/GLThread.java

```
private long startTime;
private void init(GL10 gl) {
    startTime = System.currentTimeMillis();
}
private void drawFrame(GL10 gl) {
    // ... Set rotation angle based on the time
    long elapsed = System.currentTimeMillis() - startTime;
    gl.glRotatef(elapsed * (30f / 1000f), 0, 1, 0);
    gl.glRotatef(elapsed * (15f / 1000f), 1, 0, 0);
}
```

这段代码在主循环中每迭代一次就会旋转一下立方体。具体地说，就是每隔1

秒立方体就会围绕x轴旋转 30° ，围绕y轴旋转 15° 。最终得到的是一个很漂亮、动作很平滑且不断旋转的立方体（参见图10-5）。



图10-5 旋转立方体

基于时间的动画

此示例的第一个版本会不断跟踪当前的旋转角度，并且每循环一次就让旋转角度递增。你能找出理由来说明这并不是一个好想法吗？

由于Android能够在各种不同的设备上运行，所以无法预测设备绘制一个帧需要多长时间。极可能要花半秒的时间，也可能只需 $1/100$ 秒。如果在每一帧中将对象移动固定的距离，那么在速度较慢的设备上对象会移动得更慢，在速度较快的设备上对象会移动得更快。通过将移动量与所花的时间绑定，就可以在任何设备上预测设备的移动情况。速度较快的硬件能够更流畅地绘制动画，但对象从A移到B所花的时间是相同的。

10.8 应用纹理

尽管场景变得越来越有趣了，但是没有人会将其与现实生活混淆。每种对象都

有纹理，例如砖墙的粗糙表面或花园小路上的砂石。你是否拥有贴面的桌子？木料贴面就是一种木材纹理的照片，它可粘在廉价材料（例如塑料或碎料板）的表面。

下面将使用一幅图片对立方体进行同样的处理。遗憾的是，完成此操作的代码很长。如果暂时无法理解所有代码，也不要担心。

OpenGL/src/org/example/opengl/GLCube.java

```

Line 1  private final IntBuffer mTextureBuffer;

-
-
-   public GLCube() {
-       int texCoords[] = {
10         // FRONT
-         0, one, one, one, 0, 0, one, 0,
-         // BACK
-         one, one, one, 0, 0, one, 0, 0,
-         // LEFT
10         one, one, one, 0, 0, one, 0, 0,
-         // RIGHT
-         one, one, one, 0, 0, one, 0, 0,
-         // TOP
-         one, 0, 0, 0, one, one, 0, one,
15         // BOTTOM
-         0, 0, 0, one, one, 0, one, one, };
-
-         // ...
-         ByteBuffer tbb = ByteBuffer.allocateDirect(texCoords.length * 4);
-         tbb.order(ByteOrder.nativeOrder());
20         mTextureBuffer = tbb.asIntBuffer();
-         mTextureBuffer.put(texCoords);
-         mTextureBuffer.position(0);
-     }

25     static void loadTexture(GL10 gl, Context context, int resource) {
-         Bitmap bmp = BitmapFactory.decodeResource(
-             context.getResources(), resource);

-         ByteBuffer bb = extract(bmp);

30         load(gl, bb, bmp.getWidth(), bmp.getHeight());
-     }

-     private static ByteBuffer extract(Bitmap bmp) {
35         ByteBuffer bb = ByteBuffer.allocateDirect(bmp.getHeight()
-             * bmp.getWidth() * 4);
-         bb.order(ByteOrder.BIG_ENDIAN);
-         IntBuffer ib = bb.asIntBuffer();

-         // Convert ARGB -> RGBA
40         for (int y = bmp.getHeight() - 1; y > -1; y--) {
-             for (int x = 0; x < bmp.getWidth(); x++) {
-                 int pix = bmp.getPixel(x, bmp.getHeight() - y - 1);
-                 // int alpha = ((pix >> 24) & 0xFF);
-                 // int red = ((pix >> 16) & 0xFF);
45

```

```

        int green = ((pix >> 8) & 0xFF);
        int blue = ((pix) & 0xFF);

        // Make up alpha for interesting effect
        ib.put(red << 24 | green << 16 | blue << 8
            | ((red + blue + green) / 3));
    }
    bb.position(0);
    return bb;
}

private static void load(GL10 gl, ByteBuffer bb,
    int width, int height) {
    // Get a new texture name
    int[] tmp_tex = new int[1];
    gl.glGenTextures(1, tmp_tex, 0);
    int tex = tmp_tex[0];

    // Load it up
    gl.glBindTexture(GL10.GL_TEXTURE_2D, tex);
    gl.glTexImage2D(GL10.GL_TEXTURE_2D, 0, GL10.GL_RGBA,
        width, height, 0, GL10.GL_RGBA,
        GL10.GL_UNSIGNED_BYTE, bb);
    gl.glTexParameterx(GL10.GL_TEXTURE_2D,
        GL10.GL_TEXTURE_MIN_FILTER, GL10.GL_LINEAR);
    gl.glTexParameterx(GL10.GL_TEXTURE_2D,
        GL10.GL_TEXTURE_MAG_FILTER, GL10.GL_LINEAR);
}
}

```

代码很复杂的主要原因是需要将纹理从Android格式（一个便携式网络图片文件）转换为OpenGL可以理解的格式（按照红、绿、蓝和alpha值的顺序打包的整数）。`extract()`方法（第34行）接受一个Android的Bitmap，我们使用`BitmapFactor.decodeResource()`对其解码并将其复制到Java NIO ByteBuffer中，然后进行颜色转换。`Load()`方法（第57行）将此缓冲区的内容传递给OpenGL，后者将其加载到一个纹理上，并准备进行绘图。

接下来需要告诉OpenGL使用纹理坐标。将以下代码行添加到`draw()`方法中：

```
OpenGL/src/org/example/opengl/GLCube.java
```

```
gl.glTexCoordPointer(2, GL10.GL_FIXED, 0, mTextureBuffer);
```

最后需要在`GLThread.init()`中调用`loadTexture()`方法：

```
OpenGL/src/org/example/opengl/GLThread.java
```

```
gl.glEnableClientState(GL10.GL_TEXTURE_COORD_ARRAY);
gl.glEnable(GL10.GL_TEXTURE_2D);
```

```
GLCube.loadTexture(gl, view.getContext(), R.drawable.android);
```

这段代码启用了纹理和纹理坐标，然后调用loadTexture()方法，向其传递Activity上下文和资源ID，使其能够加载纹理图像。

R.drawable.android是一个缩小到128×128像素的PNG文件，我将其复制为res/drawable/ android.png。在本书附带的可下载代码包中找到它。注意，数字128不会在代码中的任何位置出现，所以可使用更大或更小的图像代替它。但是，我们需要让图像足够小，因为extract()方法非常慢。希望在未来的Android版本中，谷歌公司会提供一种更易用、运行速度更快的方式来完成此操作。

可以在图10-6中看到现在的工作进度。



图10-6 应用一种纹理

10.9 透明效果

为了让工作更有趣，下面将立方体的一部分设置为透明的。将以下内容添加到GLThread.init()中：

```
OpenGL/src/org/example/opengl/GLThread.java
boolean SEE_THRU = true;
```

```
// ...
if (SEE_THRU) {
    glDisable(GL10.GL_DEPTH_TEST);
    glEnable(GL10.GL_BLEND);
    glBlendFunc(GL10.GL_SRC_ALPHA, GL10.GL_ONE);
}
```

这段代码关闭了深度测试，因为我们希望看到模糊的对象以及前景中的清晰对象。这段代码还打开了一种混合模式，根据对象的alpha通道值来确定对象的不透明性。在GLCube中，将alpha通道定义为纹理中红色、绿色和蓝色值的平均值，这是计算颜色透明度时使用的一种很粗略的方式^①。其实际效果是立方体的后面部分将透过前面较暗的部分显现出来。要查看最终的效果，参见图10-7。



图10-7 最终版本：一个透明的立方体

读者可以自行实现这种透明效果的打开和关闭。试用不同的混合模式来获得出色的效果。

① 一些纯粹主义者对这类数学方法不屑一顾，但是这些技巧在3D图形课程中得到了广泛应用。大多数物理效果的精确实现都需要高昂的代价，因此通常采用估计值和一些快捷方法。无论如何，为什么要将自己局限在物理世界中呢？

10.10 快速阅读指南

本章介绍了如何使用Android的3D图形库。由于Android使用了行业标准的OpenGL ES API，所以要想学习更多相关的知识，在本书之外找到各类参考资料也很容易。特别是，我建议你了解一下JSR 239 API规范的Java文档^①。

接下来就要看你的了！你拥有了所需的各种工具，去大胆实现自己的创意吧！

^① 参见网页<http://java.sun.com/javame/reference/apis/jsr239>。

Part 4

第四部分

附 录

本 部 分 内 容

- 附录A Java与Android语言及其API
- 附录B 参考书目

Java与Android语言及其API

Android程序的大部分内容都是使用Java语言编写的,并且使用Java 5 Standard Edition (SE) 库API。之所以说是“大部分内容”,是因为其中存在一些差异。本附录主要介绍常规的Java与在Android中所看到的Java内容有何差异。如果你对其他平台上的Java开发很精通,那么应该仔细了解一下需要“忘记”哪些技巧。

A.1 语言子集

Android使用标准Java编译器将源代码编译为常规的字节码,然后将这些字节码转换为Dalvik指令。因此,Android支持完整的Java语言,而不只是Java的一个子集。与此相比,GWT (Google Web Toolkit, Google网络工具集)拥有自己的Java到JavaScript转换器。通过使用内置的编译器和字节码,甚至不需要拥有在应用程序中所使用的库的源代码。

A.1.1 语言级别

Android支持与Java Standard Edition 5或更早版本兼容的代码。Java 6和7的类格式和功能不再受支持,但是可以在未来的版本中添加对它们的支持。

A.1.2 内置的类型

所有Java内置的类型(包括byte、char、short、int、long、float、double、Object、String和数组)都受支持。但是,在目前移动设备中常用的硬件上,浮点功能是通过模拟来实现的。这意味着它在软件中执行,而不是在硬件中执行,其结果是运行速度比平常情况下慢得多。简单的实数操作需要花1毫秒来完成。尽管偶尔使用没什么影响,但是最好避免在注重性能的代码中使用float或double。

A.1.3 多线程和同步

多线程通过时间分片 (time slicing) 的方式来支持的: 为每个线程分配若干毫秒来运行, 然后执行上下文交换让另一个线程运行。尽管Android能够支持任意数量的线程, 但通常只应该使用一个或两个。一个线程专用于处理主用户界面 (如果有), 另一个线程用于比较耗时的操作, 如计算或网络I/O。

Dalvik VM实现synchronized关键字, 以及Object.wait()、Object.notify()和Object.notifyAll()等与同步相关的库方法。它还支持将java.util.concurrent包用于更复杂的算法。像在任何Java程序中一样, 使用这些方法可以避免多个线程彼此干扰。

A.1.4 反射

尽管Android平台支持Java反射, 但作为一条一般的规则, 应该避免使用它。原因很简单: 性能, 因为反射比较慢。可以考虑使用编译时工具和预处理器等替代方法。

A.1.5 终结

Dalvik VM支持在垃圾收集期间终结对象, 就像常规的Java VM一样。但是, 大部分Java专家都建议不要依赖于终结器, 因为无法预测终结器何时 (或是否) 会运行。可以使用显式的close()或terminate()方法来代替终结器。Android是专为资源有限的硬件设计的, 所以应尽快释放不再需要的所有资源, 这一点很重要。

A.2 标准库子集

Android支持Java Standard Edition 5.0库的一个相对较大的子集。该库中的一些内容未包含在这个子集中, 因为它们没有什么用处 (例如打印), 其他内容被排除的原因是已有专门针对Android的更好的API (例如用户界面)。

A.2.1 支持的包

Android支持以下标准包。可以查阅Java 2 Platform Standard Edition 5.0 API文

档^①，获取关于如何使用它们的信息。

- `java.awt.font`: 针对Unicode和字体的一些常量。
- `java.io`: 文件和流I/O。
- `java.lang` (`java.lang.management`除外): 语言和异常支持。
- `java.math`: 大数字、舍入、精度。
- `java.net`: 网络I/O、URL、套接字。
- `java.nio`: 文件和通道I/O。
- `java.security`: 授权、证书、公钥。
- `java.sql`: 数据库接口。
- `java.text`: 格式化、自然语言、整理。
- `java.util` (包括`java.util.concurrent`): 列表、映射、集、数组、集合。
- `javax.crypto`: 密码、公钥。
- `javax.microedition.khronos`: OpenGL图形 (来自Java Micro Edition)。
- `javax.net`: 套接字工厂、SSL。
- `javax.security` (`javax.security.auth.kerberos`、`javax.security.auth.spi`和`javax.security.sasl`除外)。
- `javax.sql` (`javax.sql.rowset`除外): 更多数据库接口。
- `javax.xml.parsers`: XML解析。
- `org.w3c.dom` (不是子包): DOM节点和元素。
- `org.xml.sax`: 针对XML的简单API。

注意，尽管常规Java SQL数据库API (JDBC) 也包含在内，但不要使用它们访问本地SQLite数据库，而要使用`android.database` API (参见第9章)。

A.2.2 不支持的包

以下包也是Java 2 Platform Standard Edition中的一部分，但Android不支持它们。

- `java.applet`;
- `java.awt`;
- `java.beans`;

^① 参见网页<http://java.sun.com/j2se/1.5.0/docs/api>。

- java.lang.management;
- java.rmi;
- javax.accessibility;
- javax.activity;
- javax.imageio;
- javax.management;
- javax.naming;
- javax.print;
- javax.rmi;
- javax.security.auth.kerberos;
- javax.security.auth.spi;
- javax.security.sasl;
- javax.sound;
- javax.swing;
- javax.transaction;
- javax.xml (javax.xml.parsers除外);
- org.ietf*;
- org.omg*;
- org.w3c.dom* (子包)。

A.3 第三方库

除了上面列出的标准库，Android SDK还附带了很多第三方库，以方便Android的使用。

- org.apache.http: HTTP身份验证、cookie、方法和协议。
- org.json: JavaScript对象表示法 (JavaScript Object Notation)。
- org.xml.sax: XML解析。
- org.xmlpull.v1: XML解析。

参考书目

B

- [Bur05] Ed Burnette. *Eclipse IDE Pocket Guide*. O'Reilly & Associates, Inc, Sebastopol, CA, 2005.
- [Gen06] Jonathan Gennick. *SQL Pocket Guide*. O'Reilly Media, Inc., Sebastopol, CA, second edition, 2006.
- [Goe06] Brian Goetz. *Java Concurrency in Practice*. Addison-Wesley, Reading, MA, 2006.
- [Owe06] Mike Owens. *The Definitive Guide to SQLite*. Apress, Berkeley, CA, 2006.

“这本书极其出色，不仅文笔流畅、浅显易懂，内容也妙趣横生。本书既恰到好处地讲解了Android独有的特性，同时也突出了高质量编程的原则。”

——Anthony Stevens, PocketJourney创始人兼CTO, Google Android竞赛前20强

“Ed Burnette的这本书虽然篇幅不长，但内容丰富，保持了Pragmatic系列图书的一贯风格。仅凭2D和3D图形方面的内容，本书就非常值得所有Android开发人员拥有。”

——Mark Murphy, CommonsWare创始人

Hello, Android

Introducing Google's Mobile Development Platform

Android基础教程

Android是Google推出的基于Linux和Java技术的开源移动开发平台。自2007年问世以来，得到了全球众多厂商和运营商的支持，迅速成为智能手机主流操作系统。

本书是一部关于Android开发的基础教程，采用Pragmatic系列图书一贯的由浅入深、循序渐进的方式讲解了Android程序设计的核心概念和技术。本书不仅结合数独游戏开发案例形象生动地讲解了Android生命周期、用户界面、2D图形、多媒体以及简单的数据存储等基础知识，而且还深入探讨了外部通信、基于位置的服务、内置SQLite数据库以及强大的3D图形等高级主题。此外，每章最后都提供“快速阅读指南”，通过它你无需按照顺序阅读，即可迅速找到需要的信息，高效地完成工作。



Ed Burnette 资深软件技术专家，拥有20多年软件开发经验。他是SAS高级计算机实验室的联合创始人和高级研究员，也是www.planetandroid.com网站的创办人和ZDNet的专栏作家。除本书外，他还出版了Google Web Toolkit: Taking the Pain out of Ajax和Eclipse IDE Pocket Guide等著作。

The Pragmatic Programmers

本书相关信息请访问：图灵网站 <http://www.turingbook.com>

读者/作者热线：(010)51095186

反馈/投稿/推荐信箱：contact@turingbook.com

分类建议 计算机/移动开发

人民邮电出版社网址：www.ptpress.com.cn



ISBN 978-7-115-21536-9



ISBN 978-7-115-21536-9

定价：39.00元

更多资源请访问我的新浪博客 <http://blog.sina.com.cn/ckook>