

移动与嵌入式开发技术

开始使用 Android 4 开发 Android 移动游戏



Android 4 游戏编程入门经典

[美] Mario Zechner 著
Robert Green
曾繁贰 于建业 译
王炜



Apress®

清华大学出版社

移动与嵌入式开发技术

Android 4 游戏编程 入门经典

[美] Mario Zechner 著
Robert Green
曾繁贰
于建业 译
王 炜

清华大学出版社

北 京

Mario Zechner, Robert Green

Beginning Android 4 Games Development

EISBN: 978-1-4302-3987-1

Original English language edition published by Apress, 2855 Telegraph Avenue, #600, Berkeley, CA 94705 USA. Copyright © 2011 by Apress L.P. Simplified Chinese-Language edition copyright © 2012 by Tsinghua University Press. All rights reserved.

本书中文简体字版由 Apress 出版公司授权清华大学出版社出版。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。

北京市版权局著作权合同登记号 图字：01-2011-7430

本书封面贴有清华大学出版社防伪标签，无标签者不得销售。

版权所有，侵权必究。侵权举报电话：010-62782989 13701121933

图书在版编目(CIP)数据

Android 4 游戏编程入门经典 / (美) 策希纳(Zechner, M.), (美) 格林(Green, R.) 著; 曾繁斌, 于建业, 王炜 译. —北京: 清华大学出版社, 2012.11

书名原文: Beginning Android 4 Games Development
(移动与嵌入式开发技术)

ISBN 978-7-302-30105-9

I. A… II. ①策… ②格… ③曾… ④于… ⑤王… III. ①移动终端—应用程序—程序设计
IV. ①TN929.53

中国版本图书馆 CIP 数据核字(2012)第 217838 号

责任编辑: 王 军 于 平

装帧设计: 牛艳敏

责任校对: 邱晓玉

责任印制:

出版发行: 清华大学出版社

网 址: <http://www.tup.com.cn>, <http://www.wqbook.com>

地 址: 北京清华大学学研大厦 A 座 邮 编: 100084

社 总 机: 010-62770175 邮 购: 010-62786544

投稿与读者服务: 010-62776969, c-service@tup.tsinghua.edu.cn

质 量 反 馈: 010-62772015, zhiliang@tup.tsinghua.edu.cn

装 订 者:

经 销: 全国新华书店

开 本: 185mm×260mm 印 张: 34.25 字 数: 876 千字

版 次: 2012 年 11 月第 1 版 印 次: 2012 年 11 月第 1 次印刷

印 数: 1~3000

定 价: 69.00 元

产品编号:

译者序

最近我们在移动互联网听到最多的词应该就是 Android，无疑它已经成为移动操作系统的一颗新星。越来越好的用户体验和应用程序已经博得了广大手机用户的喜爱。现在 Android 的全球市场占有率已经超越了 iPhone，在中国更是达到了 70%。这就给我们的应用开发者带来了无限的机会，也是广大应用开发者大展身手的绝佳平台。

现在，国内学习 Android 开发的人越来越多，而在 Android 平台上游戏应用占据了很大的一个比例，这就能看出市场对移动游戏有着巨大的需求。所以，很多开发者(特别是初学者)都希望能拥有一本 Android 游戏开发方面的书籍。本书就是专为这些游戏开发的初学者量身打造的。无论你是一个 Android 游戏开发的初学者，还是一个从其他平台转向 Android 平台，准备进行 Android 游戏开发的技术人员，本书都能够给你带来一些不错的启发。

当我第一次看到该书的英文原作的时候，就喜欢上了这本书，准确地说是喜欢上了该书的作者。我为他的认真和一种分享的精神所感动。因为它不再是一种简单的理论知识，而是一个又一个具体的实例。它让你在学习 Android 游戏开发的过程中不再感觉到枯燥乏味，而是获得一次又一次的惊喜。

本书的一个最大亮点是，它教你如何从零开始学习 Android 游戏开发。只要你具有一些 Java 的基础知识和对游戏开发的激情，那么开发出一款让自己满意的游戏(甚至是带来满意的收入)已不再是遥不可及的梦想。

如果你是一个没有 Android 基础的初学者，那么请从头开始阅读。如果你是一个具有 Android 基础的开发者，那么你可以跳过前面几章直接进入游戏开发部分。书中将会带领你完成游戏的策划、设计、编码和发布等过程。同时也以逐步深入的方式，从 Android 游戏的基本架构，再到 2D 和 3D 游戏，有针对性地帮你解决 Android 游戏开发中需要注意的问题。如果你是一个 Open GL 编程的爱好者，那本书更是一个不可多得的开发教程。

我非常庆幸自己能够成为该书的一名译者，同时，也希望我们的翻译能够为你的游戏开发带来一些帮助。翻译是一项辛苦的劳动，在此，我要感谢那些给予我们支持和帮助的家人和朋友们！

本书由曾繁斌、于建业和王炜翻译，同时得到了中科院李洪刚博士和清华大学出版社李阳老师的大力协助，在此表示由衷的感谢。由于时间比较紧张，译者水平有限，若译文有不当之处，敬请读者批评指正。

译者

作者简介

Mario Zechner 白天是一名软件研发工程师，到了晚上就变身为热情高涨的游戏开发者，以 Badlogic Games 的名义发布游戏。他开发了基于 Android 平台的游戏 Newton，和基于 Windows、Linux 和 Mac OSX 的游戏 Quantum 以及其他众多游戏原型和小型游戏。目前，他正致力于一个名为 libgdx 的开源跨平台游戏开发解决方案。除了编写代码以外，他还积极撰写关于游戏开发的教程和文章，这些可以在网上特别是他的博客(<http://badlogicgames.com>)上免费查看。

Robert Green 是 Oregon 州 Portland 市的 Battery Powered Games 游戏工作室的创办人。他已经开发了 9 款 Android 游戏，包括 Deadly Chambers、Antigen、Wixel、Light Racer 和 Light Racer 3D。在全力开发和发布移动视频游戏以前，Robert 分别在 Minneapolis 和 Chicago 的软件公司中工作过，其中包括 IBM Interactive。Robert 目前的精力主要放在跨平台游戏开发和高性能移动游戏开发。Robert 经常在自己的博客 <http://www.rbgrn.net> 中更新关于游戏编程的知识。

前言

大家好，欢迎来到 Android 游戏开发的世界。你阅读本书是为了学习 Android 游戏开发，希望我们能帮你实现这一目标。

我们将一起探讨许多游戏开发的主题和思想，包括 Android 基础知识、音频与图形编程、少量的数学或物理知识以及让人头疼的 OpenGL ES 编程。基于以上内容，我们将开发三款不同的游戏，并且其中一款为 3D 游戏。

如果你知道自己在做什么，游戏编程就会变得很容易。因此本书不仅提供了你可以重用的代码片段，同时也会让你知道游戏开发的方向。了解游戏开发的基本原理是进行日益复杂的游戏开发的关键。这样，你不仅可以写出类似于本书开发的各种游戏，而且还有足够的能力从网上或书店里吸取知识，甚至开创出你自己的游戏开发新天地。

本书读者对象

本书主要面向 Android 游戏编程的初学者。你不需要具备任何游戏开发的经验，因为本书将介绍 Android 游戏开发的所有基础知识。但是你必须具备一点 Java 基础知识。如果你对此觉得有些生疏，建议在线阅读 Bruce Eckel 撰写的 *Thinking in Java* 一书(Prentice Hall, 2006)，该书是学习 Java 编程语言的优秀入门书籍。除此之外，你不需要具备其他知识，包括 Android 或 Eclipse。

该书同时也面向那些希望涉足 Android 的中级游戏开发人员。可能有些内容对他们来说已不是什么新知识，但仍有很多技巧和提示值得体会。如果说 Android 是一只奇怪的野兽，那么该书将是你向它吹响战斗号角的指南。

本书组织结构

本书将以循序渐进的方式逐步展示从基础到硬件加速等各种深度游戏开发的过程。在各章节中，我们创建了可重用的代码库，所以建议按顺序阅读本书各章。当然，有经验的读者可以跳过某些有把握的章节，但是要浏览一下跳过的章节中的程序清单，这样可以知道在后续章节中如何使用它们的类和接口。

本书的源代码

本书是完全独立的，包括运行本书的示例和游戏所需要的全部代码。但是，将本书代码复制到 Eclipse 开发环境时很容易出错。因为游戏应用程序不仅包含代码，它还需要其他一些无法从书中复制的资源。而且，复制本书的代码文本到 Eclipse 开发环境的过程中可能会引入一些错误。虽然 Robert(本书的技术编辑)和我做出了很大的努力来确保书中的程序清单少出错误，但我们知道仍有些地方还不尽如人意。

为了帮助你顺利学习，我们创建了一个 Google Code 项目，提供以下内容：

- 完整的源代码和资源遵循 GPL 3.0 版本，可通过该项目的 Subversion 存储库得到整个内容。
- 一个快速入门指南，让你知道如何以文本形式将项目导入到 Eclipse 开发环境，同时提供一个视频演示。
- 问题追踪系统，你可以提交任何发现的错误，包括书本身和书中附带的代码错误。一旦你向系统提交一个问题，我们就可以在 Subversion 存储库中修复。这样你就总能得到本书一份最新的、(希望)没有错误的源代码，当然其他读者也能从中受益。
- 一个讨论组，任何人都可以免费参加并讨论本书的内容，当然我也会参与讨论。

包含代码的每一章在 Subversion 存储库下都有一个对应的 Eclipse 项目。每个项目都是相对独立的，不依赖于其他项目，因为在本书进行的过程中，将逐渐改进其中的一些框架类。例如，第 5 章、第 6 章的代码就位于 ch06-mmom 项目中。

访问网址 <http://code.google.com/p/beginning-android-games> 可以找到 Google Code 项目。

目 录

第 1 章	Android, 后起之秀	1			
1.1	Android 简介	1			
1.2	版本分裂	3			
1.3	谷歌的角色	3			
1.3.1	Android 开源项目	3			
1.3.2	Android Market	4			
1.3.3	挑战赛、设备播种计划 和谷歌 I/O	4			
1.4	Android 的功能和体系结构	5			
1.4.1	内核	6			
1.4.2	运行库和 Dalvik 虚拟机	6			
1.4.3	系统库	7			
1.4.4	应用程序框架	8			
1.5	软件开发工具包	8			
1.6	开发人员社区	9			
1.7	设备, 设备, 设备	9			
1.7.1	硬件	9			
1.7.2	设备的范围	10			
1.8	所有设备之间的兼容性	15			
1.9	不同的手机游戏	15			
1.9.1	人手一台游戏机	16			
1.9.2	随时上网	16			
1.9.3	普通用户与游戏迷	17			
1.9.4	市场很大, 开发人员很少	17			
1.10	小结	18			
第 2 章	从 Android SDK 开始	19			
2.1	搭建开发环境	19			
2.1.1	安装 JDK	20			
2.1.2	安装 Android SDK	20			
2.1.3	安装 Eclipse	21			
2.1.4	安装 ADT Eclipse 插件	22			
2.1.5	Eclipse 快速浏览	23			
2.1.6	一些实用的 Eclipse 快捷键	24			
2.2	Android 环境下的 Hello World	25			
2.2.1	创建项目	25			
2.2.2	进一步分析项目	26			
2.2.3	编写应用程序代码	27			
2.3	运行和调试 Android 应用 程序	29			
2.3.1	连接设备	29			
2.3.2	创建一个 Android 虚拟 设备	29			
2.3.3	运行应用程序	30			
2.3.4	调试应用程序	32			
2.3.5	LogCat 和 DDMS	34			
2.3.6	使用 ADB	36			
2.4	小结	37			
第 3 章	游戏开发基础	39			
3.1	游戏类型	39			
3.1.1	休闲游戏	40			
3.1.2	益智游戏	41			
3.1.3	动作和街机游戏	42			
3.1.4	塔防游戏	44			
3.1.5	创新	45			
3.2	游戏设计: 笔比代码更强大	46			
3.2.1	游戏的核心机制	46			
3.2.2	一个故事和一种艺术风格	47			
3.2.3	画面和切换	48			
3.3	代码: 具体细节	52			
3.3.1	应用程序和窗口管理	52			
3.3.2	输入	53			
3.3.3	文件 I/O	56			
3.3.4	音频	57			
3.3.5	图形	60			

3.3.6 游戏框架	69	5.4.5 触摸处理程序	160
3.4 小结	75	5.4.6 AndroidInput: 优秀的 协调者	167
第 4 章 面向游戏开发人员的 Android	77	5.5 AndroidGraphics 和 AndroidPixmap	169
4.1 定义一个 Android 应用程序: 清单文件	77	5.5.1 处理不同屏幕大小和 分辨率的问题	169
4.1.1 <manifest>元素	78	5.5.2 AndroidPixmap: 人物的 像素	174
4.1.2 <application>元素	79	5.5.3 AndroidGraphics: 满足 绘图需求	174
4.1.3 <activity>元素	80	5.5.4 AndroidFastRenderView	178
4.1.4 <uses-permission>元素	82	5.6 AndroidGame: 合并所有 内容	180
4.1.5 <uses-feature>元素	83	5.7 小结	184
4.1.6 <uses-sdk>元素	84	第 6 章 Mr. Nom 入侵 Android	185
4.1.7 10 个简单步骤建立 Android 游戏项目	84	6.1 创建资源	185
4.1.8 市场过滤器	86	6.2 建立项目	187
4.1.9 定义游戏图标	87	6.3 MrNomGame: 主要活动	187
4.2 Android API 基础	87	6.3.1 资源: 便捷的资源存储	188
4.2.1 创建测试项目	88	6.3.2 设置: 跟踪用户的选项设置 和高分榜	189
4.2.2 活动的生命周期	91	6.3.3 LoadingScreen: 从磁盘获取 资源	191
4.2.3 处理输入设备	96	6.4 主菜单画面	192
4.2.4 文件处理	110	6.5 HelpScreen 类	195
4.2.5 音频编程	116	6.6 高分榜画面显示	197
4.2.6 播放音效	116	6.6.1 渲染数字	198
4.2.7 音乐流	119	6.6.2 画面的实现	199
4.2.8 基本图形编程	122	6.7 抽象	201
4.3 最佳实践	143	6.7.1 抽象 Mr. Nom 的世界: 模型、视图、控制器	201
4.4 小结	144	6.7.2 GameScreen 类	211
第 5 章 Android 游戏开发框架	145	6.8 小结	218
5.1 制定计划	145	第 7 章 OpenGL ES 介绍	219
5.2 AndroidFileIO 类	146	7.1 OpenGL ES 概述以及关注它的 原因	219
5.3 AndroidAudio、AndroidSound 和 AndroidMusic	147	7.1.1 编程模型: 一个比喻	220
5.4 AndroidInput 和 Accelerometer- Handler	152		
5.4.1 AccelerometerHandler: 手机 哪一面朝上	152		
5.4.2 CompassHandler	153		
5.4.3 Pool 类: 重用相当有用	154		
5.4.4 KeyboardHandler	156		

7.1.2	投影	221	7.12.2	Android 1.5 平台下 Hero 的 奇特案例	275
7.1.3	规范化设备空间和视口	223	7.12.3	使 OpenGL ES 渲染如此 慢的原因	275
7.1.4	矩阵	223	7.12.4	移除不必要的状态 改变	276
7.1.5	渲染管道	224	7.12.5	减小纹理大小意味着需要 获取更少的像素	278
7.2	开始之前	225	7.12.6	减少 OpenGL ES/JNI 方法的 调用	278
7.3	GLSurfaceView: 从 2008 年开始, 事情变得简单了	225	7.12.7	绑定顶点的概念	279
7.4	GLGame: 实现游戏接口	228	7.12.8	写在结束之前	282
7.5	绘制一个红色的三角形	235	7.13	小结	283
7.5.1	定义视口	235			
7.5.2	定义投影矩阵	235	第 8 章	2D 游戏编程技巧	285
7.5.3	指定三角形	238	8.1	写在开始	285
7.5.4	综合示例	241	8.2	向量	286
7.6	指定每个顶点的颜色	243	8.2.1	使用向量	286
7.7	纹理映射: 轻松地创建 壁纸	246	8.2.2	一点三角学的知识	288
7.7.1	纹理坐标	247	8.2.3	实现一个向量类	289
7.7.2	上传位图	248	8.2.4	一个简单的用法示例	292
7.7.3	纹理过滤	249	8.3	2D 物理定律浅析	296
7.7.4	释放纹理	250	8.3.1	牛顿和欧拉, 永远的 好朋友	296
7.7.5	有用的代码片段	251	8.3.2	力和质量	297
7.7.6	启用纹理	251	8.3.3	理论上的运动	298
7.7.7	综合示例	251	8.3.4	运动的实现	299
7.7.8	Texture 类	253	8.4	2D 碰撞检测和对象表示	302
7.8	索引顶点: 重用是有好处的	255	8.4.1	边界形状	303
7.8.1	代码整合	256	8.4.2	构造边界形状	304
7.8.2	Vertices 类	258	8.4.3	游戏对象的属性	306
7.9	半透明混合处理	260	8.4.4	宽阶段和窄阶段碰撞检测	307
7.10	更多图元: 点、线、条 和扇	263	8.4.5	一个详细的示例	313
7.11	2D 变换: 操作模型视图 矩阵	264	8.5	2D 照相机	324
7.11.1	世界空间和模型空间	264	8.5.1	Camera2D 类	327
7.11.2	再次讨论矩阵	265	8.5.2	示例	328
7.11.3	第一个使用平移的 示例	266	8.6	纹理图集	329
7.11.4	更多的变换	270	8.7	纹理区域、精灵和批处理: 隐藏 OpenGL ES	334
7.12	性能优化	273			
7.12.1	测量帧率	273			

8.7.1 TextureRegion 类	334	10.2.2 示例	406
8.7.2 SpriteBatcher 类	335	10.3 透视投影: 越近则越大	409
8.8 精灵动画	343	10.4 z-buffer: 化混乱为有序	411
8.8.1 Animation 类	344	10.4.1 完善上一个例子	412
8.8.2 示例	345	10.4.2 混合: 身后空无一物	413
8.9 小结	348	10.4.3 z-buffer 精度与 z-fighting	416
第 9 章 Super Jumper: 一个 2D OpenGL ES 游戏	351	10.5 定义 3D 网格	417
9.1 核心游戏机制	351	10.5.1 立方体: 3D 中的“Hello World”	417
9.2 背景故事和艺术风格	352	10.5.2 一个示例	419
9.3 画面和切换	352	10.6 矩阵和变换	422
9.4 定义游戏世界	353	10.6.1 矩阵堆栈	423
9.5 创建资源	355	10.6.2 用矩阵堆栈实现分层 系统	425
9.5.1 UI 元素	355	10.6.3 木箱太阳系的简单 实例	425
9.5.2 使用点阵字体处理文本	356	10.7 小结	433
9.5.3 游戏元素	358	第 11 章 3D 编程技巧	435
9.5.4 用于救援的纹理图集	359	11.1 准备工作	435
9.5.5 音乐与音效	360	11.2 3D 中的向量	436
9.6 实现 Super Jumper	361	11.3 OpenGL ES 中的光照	440
9.6.1 Assets 类	361	11.3.1 光照的工作机制	440
9.6.2 Settings 类	364	11.3.2 光源	441
9.6.3 主活动	366	11.3.3 材质	442
9.6.4 Font 类	367	11.3.4 OpenGL ES 中如何对光照 过程进行运算: 顶点 法线	442
9.6.5 GLScreen	369	11.3.5 实践	443
9.6.6 主菜单画面	369	11.3.6 关于 OpenGL ES 中光照 应用的一些建议	456
9.6.7 帮助画面	372	11.4 材质变换(Mipmapping)	456
9.6.8 高分画面	374	11.5 简单的照相机	460
9.6.9 模拟类	377	11.5.1 第一人称照相机或欧拉 照相机	460
9.6.10 游戏画面	390	11.5.2 一个欧拉照相机的示例	463
9.6.11 WorldRenderer 类	397	11.5.3 跟随照相机	468
9.7 是否需要优化	401	11.6 加载模块	470
9.8 小结	402	11.6.1 Wavefront OBJ 格式	470
第 10 章 OpenGL ES: 进入 3D 世界	403		
10.1 准备工作	403		
10.2 3D 中的顶点	404		
10.2.1 Vertices3: 存储 3D 空间 位置	404		

11.6.2	OBJ 加载器的实现	471	第 13 章	发布游戏	523
11.6.3	使用 OBJ 加载器	475	13.1	关于测试	523
11.6.4	关于加载模型的一些 建议	475	13.2	成为注册开发人员	524
11.7	3D 中的一些物理知识	476	13.3	给游戏的 APK 包签名	524
11.8	碰撞检测与 3D 中的对象 表达法	477	13.4	将游戏发布至 Market	527
11.8.1	3D 中的边界形状	477	13.4.1	上传资源	527
11.8.2	边界球重叠测试	477	13.4.2	产品详情	528
11.8.3	GameObject3D 与 Dynamic- GameObject3D	478	13.4.3	发布选项	528
11.9	小结	479	13.4.4	发布	529
			13.4.5	市场推广	529
第 12 章	Droid Invaders 游戏	481	13.5	开发人员控制台	529
12.1	游戏的核心机制	481	13.6	小结	530
12.2	游戏的故事背景与艺术 风格	483	第 14 章	进阶内容	531
12.3	屏幕与场景切换	483	14.1	社交网络	531
12.4	定义游戏世界	484	14.2	位置识别	531
12.5	创建资源	485	14.3	多玩家功能	532
12.5.1	用户界面的资源	485	14.4	OpenGL ES 2.0 以及更多 内容	532
12.5.2	游戏资源	486	14.5	框架及引擎	532
12.5.3	音效与音乐	488	14.6	网络资源	534
12.6	开始编写代码	488	14.7	结束语	534
12.7	Assets 类	489			
12.8	Settings 类	492			
12.9	主活动	493			
12.10	主菜单	494			
12.11	游戏设置画面	496			
12.12	模拟类	499			
12.12.1	Shield 类	499			
12.12.2	Shot 类	500			
12.12.3	Ship 类	500			
12.12.4	Invader 类	502			
12.12.5	World 类	505			
12.13	GameScreen 类	510			
12.14	WorldRender 类	516			
12.15	游戏优化	521			
12.16	小结	522			

第 1 章

Android，后起之秀

对于我们这批 20 世纪 90 年代初出生的孩子们来说，成长过程中一直伴随着任天堂的 Game Boy 游戏机和 Sega 的 Game Gear 游戏机。我们花了无数个小时帮助 Mario 拯救公主，获得俄罗斯方块的最高分，通过连线和朋友们在 Super RC Pro-Am 中进行比赛等。对游戏的热情促使我们想要创建自己的世界，并与朋友们进行分享。我们开始在 PC 上编程，但很快就发现这些游戏的小杰作不能在游戏机上使用。我们仍然是充满热情的程序员，但随着时间的推移，我们对玩视频游戏的兴趣在慢慢减退。而且，我们的游戏机最终坏掉了……

转眼到了 2011 年，智能手机已经成为这个时代的新的移动游戏平台，相比于传统的专用的手持游戏设备(Nintendo DS 或 PlayStation PSP)。这又引起了我们的兴趣，我们开始调查哪种移动平台适合我们的开发需求。Apple 公司的 iOS 系统看上去是个不错的游戏开发平台。不过，我们很快意识到该系统是封闭的，只有在经过苹果公司许可的情况下才能够和他人分享我们的作品，而且还需要一台 Mac 电脑来进行开发。所以，我们最终选择了 Android 平台。

我们马上爱上了 Android 平台，它的开发环境适用于所有的主流操作系统平台，没有任何限制。它还有一个很活跃的开发人员社区，在那里你可以寻求帮助以及获取全面的开发文档。任何人都可以免费使用和自由分享，同时如果你想盈利，可以在几分钟内把最新和最好的应用发布到一个全球性的电子市场上去，因为那里有数以百万计的用户。

接下来唯一要弄明白的事情是，如何利用 PC 游戏开发经验来开发 Android 平台的游戏。在后续章节中，我们将与你分享我们的开发经验，带你走进 Android 游戏开发世界。当然，这也许是个自私的计划，因为我们希望有更多的移动游戏出现。

让我们开始认识我们的新朋友吧——Android。

1.1 Android 简介

2005 年，谷歌收购了一家名叫 Android 的小型初创企业。那时 Android 首次被公众所关注，

同时也引发了大家对谷歌进军移动领域的猜测。直至 2008 年谷歌发布了 Android 1.0 版本，才使猜测烟消云散。Android 成为移动市场的一个新挑战者。从那时起，它就对已有平台，如 iOS(原来叫做 iPhone OS)和 BlackBerry 发起了挑战。Android 发展得相当好，每年的市场占有率都在提高。虽然移动技术的未来总有许多变化因素，但是有一个事实可以确定：Android 不会是昙花一现。

由于 Android 是开源的，因此手机制造商使用这一新平台的门槛很低。他们可以生产出各种价位的产品，通过修改 Android 系统的配置，以适应特定移动设备的需求。因此，Android 系统不仅适合于高端设备，也可以部署到低端设备上，正因为这样 Android 平台才有更广泛的受众面。

2007 年末开放手机联盟(OHA)的成立是 Android 取得成功的重要因素之一，该联盟包括宏达电(HTC)、高通(Qualcomm)、摩托罗拉(Motorola)和英伟达(NVIDIA)，他们共同致力于移动设备开放标准的制定。虽然 Android 的核心部分由谷歌负责开发，但其他的开放手机联盟成员也以不同的形式贡献了自己的一份力量。

Android 本身是一个移动操作系统和基于 Linux 内核 2.6 版本的平台，它可免费用于商业或者非商业用途。许多开放手机联盟的成员通过修改 Android 系统的用户界面来构建自定义的 Android 版本，以满足他们设备的需求，例如：宏达电的 Sense 和摩托罗拉的 MOTOBLUR。Android 的开源性也使得业余爱好者能够创建和发布他们自己的 Android 版本，这些常被称为 mod、固件或者 rom。截止本书创作时，一个最为大家所熟知的 rom 是由 Cyanogen 开发的，旨在为各种 Android 设备提供最新和最好的改进。

自 2008 年发布以来，Android 已经进行了 7 个版本的更新，所有版本代号均以甜品为名(Android 1.1 版本的除外，不过现在这个版本已经很少用了)。每个新版本都在原来 Android 平台的基础上增加了新功能，这些新功能或多或少给游戏开发人员带来一些启发。1.5 版本(Cupcake)开始支持在 Android 应用程序中包含本地库，而在以前的版本中只能使用纯 Java 编写应用程序。在我们更关注于程序性能的情况下，本地代码就能体现出它的优越性。1.6 版本(Donut)引入了对不同屏幕分辨率的支持，这对编写 Android 游戏有一些影响，所以本书中会有几次讨论该特性。2.0 版本(Eclair)增加了对多点触摸屏幕的支持，而 2.2 版本(Froyo)则向 Dalvik 虚拟机(VM)增加了即时编译(JIT)的功能，以提高 Java 程序在 Android 中的性能。JIT 能够大幅度提升 Android 平台下 Java 应用程序的性能，在最好的情况下，执行时间能够缩短到原来的 1/5。2.3 版本(Gingerbread)为 Dalvik 虚拟机增加了一个新的并发垃圾回收器。2011 年早期，Android 生产了一个针对平板电脑的版本，它的代号为 Honeycomb，版本号为 3.0。Honeycomb 中对应用程序接口(API)做了大量更改，比到目前为止的其他任何 Android 版本都多。到版本 3.1 时，Honeycomb 为拆分和管理高分辨率的大平板电脑屏幕提供了广泛的支持。它加入了更多与 PC 类似的功能，例如支持 USB 主设备和 USB 外设，包括键盘、鼠标和摇杆。这个版本唯一的问题是只面向平板电脑。Android 的小屏幕/智能手机版本仍然只能使用 2.3 版本。Android 4.0 发布了，情况有了改观。Android 4.0 的代号为 Ice Cream Sandwich(ICS)，它是将 Honeycomb(3.1)和 Gingerbread(2.3)合并后得到的一组在平板电脑和手机上都可以正常工作的公共功能。

ICS 对用户的热情是一种极大的激励，它对 Android 的用户界面做了很多改进，并且内置了浏览器、电子邮件客户端和照片服务等应用程序。ICS 为开发人员提供了许多便利，其中一

个就是合并了 Honeycomb UI API, 使手机也支持大屏功能。ICS 还合并了 Honeycomb 的 USB 外设支持, 使制造商也可以选择支持键盘和摇杆。ICS 添加了一些新的 API, 例如 Social API, 它为联系人、个人资料数据、状态更新和照片提供了一个集中的存储区。对于 Android 开发人员来说, 值得庆幸的是 ICS 的核心仍然保持了良好的向后兼容性, 这确保了巧妙构建的游戏可以在较老的版本(如 Cupcake 和 Eclair)上也可以很好地兼容。

1.2 版本分裂

Android 系统的快速更新也给那些制造商带来一些不便, 因为制造商们在选择开发自有手机界面的时候, 必须紧跟各种 Android 新版本的发布。这可能使一部手机没过几个月系统就已过时了, 而运营商和制造商又没有提供新 Android 版本的更新, 这就造成了各种 Android 版本的分裂。

版本的分裂有多方面的影响。对最终用户而言, 因为使用的 Android 版本较老, 这就意味着他们无法安装和使用一些新的应用程序的功能。而对于开发人员而言, 这意味着当他们开发应用的时候, 需要维护多个 Android 版本。早期 Android 版本的应用程序通常可以在新 Android 版本上正常运行, 反之则不然。新版本中添加的一些功能(例如, 多点触摸)当然无法在旧版本中使用, 所以开发人员不得不针对不同的版本开发不同的代码。

在 2011 年, 许多主要的 Android 设备制造商同意在 18 个月的设备生命期中支持最新的 Android OS。听起来时间好像不长, 但是对于减轻分裂, 这已经是迈出了一大步。这也意味着更多的手机会更快地支持 Android 的新功能(例如 Ice Cream Sandwich 中的新 API)。不过, 市场上还是会有很多手机运行旧版本的 Android。如果游戏开发商想获得高市场占有率, 就要使游戏至少能够在 6 个不同的 Android 版本, 超过 400 种手机上运行。

听起来似乎很可怕, 其实不必担心。通常情况下, 需要采取的应对措施并不多。很多时候, 我们完全可以忽视多种版本的存在, 而假装只有一个 Android 版本。作为游戏开发人员, 我们更多关注硬件的特性而忽视 API 的差异。这其实是分裂的另外一种形式, 其他平台(如 iOS)也有这个问题, 尽管没有 Android 那样明显。在本书中, 我将对可能会给你开发 Android 游戏造成阻碍的分裂问题进行讨论。

1.3 谷歌的角色

虽然官方表示 Android 是开放手机联盟的心血结晶, 但在实现 Android 和为 Android 提供必要的发展环境方面, 谷歌无疑付出了最多的努力。

1.3.1 Android 开源项目

谷歌的贡献可归结为“Android 开源项目”。绝大多数的代码遵循 Apache License 2 协议, 与其他开源许可协议(如 GNU General Public License, GPL)相比, 这种协议更加开放, 限制程

度更低。每个人都可以自由地使用这些源代码来构建自己的系统。然而，自称兼容 Android 的系统首先要通过 Android 的兼容性测试，这个过程可以确保应用程序能够与第三方的应用程序(像我们这样的开发人员开发的应用程序)基本兼容。兼容的系统可加入 Android 生态系统，其中包括 Android Market。

1.3.2 Android Market

Android Market 由 Google 于 2008 年 10 月对外公布，这是一个在线软件商店，用户可以从其中搜索并安装第三方应用程序。该电子市场目前只能通过移动设备中的市场应用程序来访问，不过在不久的将来这种情况就会发生改变，因为谷歌已经承诺将发布可以使用桌面浏览器访问的网上商店。

电子市场允许第三方开发人员发布免费或付费的应用。付费应用可在许多国家购买，集成的购买系统使用 Google Checkout 处理汇率。Android 电子市场还提供了针对不同国家手动为应用程序单独定价的选项。

在注册了谷歌账户以后，用户可以访问电子市场。目前只能通过信用卡来购买电子市场中的应用。购买者自购买应用时起的 15 分钟内，可退还应用并获得全额退款，而以前的退款时间长达 24 小时。所以，这让用户有些难以接受。

开发人员为了能在电子市场上发布应用，必须注册一个谷歌开发人员账户并一次性支付 25 美元的费用。开发人员完成注册之后，几分钟内便可以开始上传应用程序。

Android 电子市场没有审批流程，而是依靠一个许可的制度。在安装一个应用之前，用户需要确认使用应用程序所需的一组权限。这些权限处理电话服务、网络接入、安全数字卡(SD)的访问等问题，确定它们之后方可安装该应用。系统正常运行的前提是用户执行了正确的操作。而在 PC 上，特别是 Windows 系统上，这样一种模式并不能很好地实现；但在 Android 上，至少直到现在它运行地很好，只有非常少的应用程序因为恶意用户行为被剔除出电子市场。

开发人员如果要出售应用程序，还需要注册一个谷歌 Checkout 账户来管理收入资金。注册是免费的，并且所有的商业金融交易都是通过该账号来进行的。谷歌还有一个包含在应用程序内的购买系统，它与 Android 电子市场和谷歌 Checkout 集成在一起。开发人员可以使用一个单独的 API 来处理应用程序内完成的购买交易。

1.3.3 挑战赛、设备播种计划和谷歌 I/O

为了吸引更多的开发人员加入 Android 平台，谷歌开始举行一些挑战赛。2008 年举行了第一次挑战赛——Android 开发人员挑战赛(ADC)，此次比赛提供了丰厚的奖金。同样谷歌在次年又举行了 Android 开发人员挑战赛，并吸引了大量的开发人员参与。也许有了前两次的基础，Android 已经吸引了大量的开发人员，感觉没有进一步进行推广的必要，谷歌在 2010 年停止了举办挑战赛。

谷歌于 2010 年初还推出了设备播种计划，给那些在电子市场上有一个或多个应用的下载量超过 5000 次，而且得到 3.5 星以上用户评价的开发人员提供一部 Motorola Droid、Motorola Milestone 或 Nexus One 手机。该计划在开发社区中十分受欢迎，不过刚开始的时候人们还有些

怀疑, 因为许多人以为收到的电子邮件是一个精心设计的骗局。幸运的是, 该计划得到了确切的落实, 成千上万的设备被送往全球各个角落的开发人员手中。谷歌的这个举措给了第三方开发人员很大的甜头, 让他们继续为该平台进行开发, 并吸引更多的开发人员。

谷歌还为开发人员提供专门的 Android 开发机(ADP)。第一代 ADP 是 T-Mobile 的 G1 手机(也被称为 HTC Dream), 第二代 ADP 是 HTC 的 Magic 手机。同时, 谷歌也发布了自己的手机——Nexus One。虽然很多人认为该机是 ADP 2 的后续, 但是 Google 最终停止了向最终用户销售 Nexus One, 现在仅对合作伙伴和开发人员出售。在 2010 年底, 谷歌发布了第三代的 ADP, 该机就是运行 Android 2.3 系统的三星 Nexus S 手机。所有的 ADP 都可通过 Android 电子市场进行购买, 当然这需要你有一个开发人员账号。不过, Nexus S 手机可以通过谷歌的网站 www.google.com/phone 进行购买。

一年一度的谷歌 I/O 开发人员大会也是每个 Android 开发人员期待的盛会。在大会上, 谷歌将会发布一些最新或最出色的技术和项目, 而在近几年, Android 是其中最让人期待的部分。同时在大会期间, 谷歌还会举行多场关于 Android 主题的会议, 而这些内容将可以通过 YouTube 的 Google Developers 频道进行观看。在 2011 年度举办的谷歌 I/O 上, 三星和谷歌向所有经常与会的人员免费赠送了 Galaxy Tab 10.1 设备。这标志着谷歌开始努力在平板电脑市场分一杯羹。

1.4 Android 的功能和体系结构

Android 并不是一个为移动设备而发布的 Linux 版本。因为当你进行 Android 开发时, 并没有太大的可能会遇到 Linux 内核。对开发人员来说, Android 是一个平台, 它抽象了底层的 Linux 内核, 并使用 Java 语言进行应用开发。从上层应用方面来看, Android 有几个不错的功能:

- **应用框架**, 提供丰富的 API 以供开发各种应用程序; 同时它允许重用或替换系统或第三方应用程序。
- **Dalvik 虚拟机**, 负责在 Android 平台上运行应用程序。
- **图形库**, 可应用于 2D 和 3D 编程。
- **多媒体库**, 支持常见的音频、视频和图片格式。如 Ogg Vorbis、MP3、MPEG-4、H.264 和 PNG 等。同时还提供专门的 API 用于回放音效, 这在游戏开发中十分方便。
- **访问外设的 API**, 如照相机、全球定位系统(GPS)、罗盘、加速度传感器、触摸屏、轨迹球和键盘等。请注意, 并非所有的 Android 设备都有这些外设——这就是前面提到的“分裂”。

当然, Android 不只有上面所提到的这些功能, 不过这些是我们进行游戏开发最为重要的部分。

Android 的体系结构是由一系列的组件所构成的, 所有的组件都是基于它下面的组件, 图 1-1 展示了 Android 主要组件的一个概览。

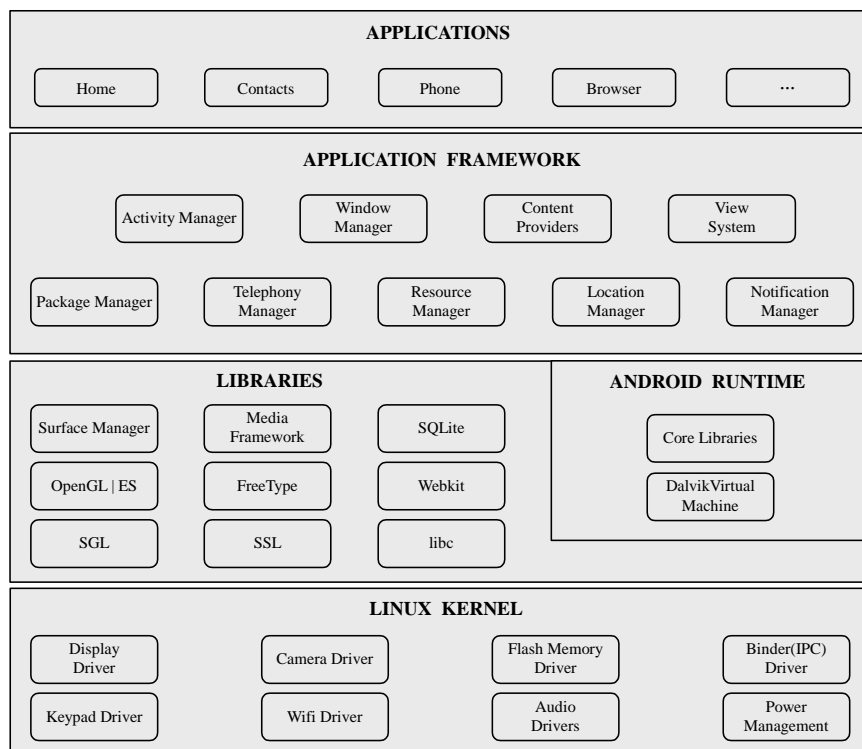


图 1-1 Android 系统体系结构概览

1.4.1 内核

从该体系结构的最底层我们可以看到，Linux 内核为硬件提供了基本的驱动程序。同时，内核还负责内存和进程的管理、网络操作等。

1.4.2 运行库和 Dalvik 虚拟机

Android 的运行库建立于内核之上，它主要负责各种 Android 应用程序的生成和运行。每一个 Android 应用程序都在 Dalvik 虚拟机的一个实例中运行，而这个实例驻留在一个由 Linux 内核管理的进程之中。

Dalvik 虚拟机运行 DEX 字节码格式的程序。通常，将普通 Java 的.class 文件转换为 DEX 格式需要一个由软件开发工具包提供的特殊工具，叫做 dx。DEX 格式相比于传统 Java 的.class 文件占用的内存更少，因为它进行了程度很大的压缩、使用了表并合并了多个.class 文件。

Dalvik 虚拟机与核心库相交交互，而核心库为上层的 Java 编程提供基本的类库。核心库与 Java SE 类库相兼容，它们都是来源于 Apache Harmony 的项目。这就意味着没有任何 Swing 或 Abstract Window Toolkit(AWT)可用，同样在 Java ME 环境中也不能使用。但是，如果你细心观察就会发现，还有很多用于 Java SE 的第三方库可应用于 Dalvik 虚拟机上。

在 Android 2.2(Froyo)之前，程序所有的字节码是经过解释的。不过 Froyo 引入了 JIT 编译器，可以在遇到字节码时就将其编译为机器码，这显著提升了需要执行大量计算的应用程序的性能。JIT 编译器可充分利用 CPU 专门为特殊计算设计的 CPU 功能，如 Floating Point Unit(FPU)。几乎每个 Android 新版本都改进了 JIT 编译器并提高了性能，这通常是以使用更多的内存为代

价实现的。不过, 这是一个可以扩展的解决方案, 因为新设备会包含越来越多的 RAM。

Dalvik 虚拟机同样集成了一个垃圾回收器(GC)。它使用标记-清除技术, 不是一种普通的垃圾回收器, 这给开发人员带来了一些难度。不过, 只要注意一些细节, 在每天进行游戏开发的时候, 垃圾回收器就不会造成问题。Android 2.3 版本提供了一个改进的并发垃圾回收器以提高开发的效率, 我们将在本书的后面部分探讨更多有关垃圾回收器的问题。

每一个应用程序都是一个运行于 Dalvik 虚拟机的实例, 有至少 16MB 的堆内存可用。较新的设备, 特别是平板电脑, 对使用的堆大小有更高的限制, 这是为了能够使用更高分辨率的图片。但是, 在游戏中仍然很容易用完所有内存。因此, 在使用图片或音频资源时要十分小心。

1.4.3 系统库

除了提供一些 Java SE 功能的核心库之外, 还有一套原生的 C/C++库(图 1-1 中的第 2 层), 可服务于应用框架层(图 1-1 中的第 3 层)。这些系统库多用于需要执行大量计算的任务, 如图形渲染、音频回放和数据库访问等任务, 当然这些不是 Dalvik 虚拟机的好选择。这些系统库通过应用框架层对外提供 API 接口, 我们就可利用它们来进行游戏开发。下面的库是我们常用的系统库:

- **Skia Graphics Library(Skia):** 这个 2D 图形渲染软件用于渲染 Android 应用程序的用户界面, 我们将用它来开发我们的第一个 2D 游戏。
- **嵌入式系统的 OpenGL(OpenGL ES):** 这是业界标准的硬件加速图形渲染库。OpenGL ES 1.0 和 OpenGL ES 1.1 在所有 Android 版本中都通过 Java 使用, 而增加了着色器支持的 OpenGL ES 2.0 则仅能应用于 Android 2.2(Froyo)版本以上的系统中。值得一提的是 Android 绑定的 OpenGL ES 2.0 并不完整, 它缺乏一些重要的方法; 而且, 模拟器和旧版本的设备仍然占据了主要的市场份额, 而它们仍然不支持 OpenGL ES 2.0。因此, 我们将更多地关注 OpenGL ES 1.0 和 1.1, 以尽可能地保持兼容, 这样也可以很自然地进入 Android 3D 编程的世界。
- **OpenCore:** 这是一个运用于音频和视频的多媒体回放和录制库。它支持多种格式, 如 Ogg Vorbis、MP3、H.264 和 MPEG-4 等。我们将会更关注音频部分, 虽然它不直接暴露在 Java 层, 但它却提供很好的服务接口。
- **FreeType:** 这是一个加载和渲染位图和向量字体的库, 其中最引人注目的是 TrueType 格式。FreeType 支持 Unicode 标准, 也包括从右到左呈现字体的阿拉伯语和其他类似的特殊语言。遗憾的是, FreeType 也并不完全适合在 Java 层, 因为目前它不支持阿拉伯语的排版等。正如 OpenCore 一样, FreeType 也不直接暴露在 Java 层, 但同样提供一些方便的服务接口。

这些系统库是构成我们游戏开发的基础, 并为我们完成了很多重要的功能。也正是因为有了它们, 我们才能用 Java 写出这样的游戏。

注意: 虽然 Dalvik 虚拟机的能力已足够我们进行各种开发, 但有些时候你可能需要更高的性能。例如, 在进行非常复杂的物理模拟或繁重的 3D 计算时, 我们通常会选择编写原生代码。本书没有介绍这方面的内容, 但已有很多开源的 Android 代码库为我们提供了很好的支持。具体可参见这个例子——<http://code.google.com/p/libgdx>。

1.4.4 应用程序框架

应用程序框架将系统库和运行库结合在一起，构成了 Android 的用户端。该框架管理着所有的应用程序，并为应用程序的运行提供了一个复杂的结构。开发人员通过一个 Java 的 API 类库来创建各种应用，这个类库可以处理 UI 编程、后台服务、通知、资源管理和外设访问等。借助于这些 API，Android 系统本身提供了一些核心应用程序，如邮件客户端等。

应用程序无论是 UI 还是后台服务，都可以与其他应用程序进行通信，这样使得一个应用程序可以重用其他应用所提供的组件。一个简单的例子就是，当一个应用程序需要拍摄一张照片的时候，通过查询系统中某个提供这种服务的应用程序的组件，然后就可以对图片进行其他操作；接着第一个应用程序就可以重用该组件(例如，一个内置的摄像头应用程序或照片库等)。这样就大大降低了开发的难度，而且开发人员也可以自定义 Android 的各种行为。

作为游戏开发人员，我们一般在这一框架内创建各种 UI 的应用程序。因此，我们将会关注应用程序的架构和生命周期以及与用户的交互。后台服务通常在游戏开发中作用不大，所以本书中没有深究其细节。

1.5 软件开发工具包

要开发 Android 应用程序，需要使用 Android 的软件开发工具包(software development kit, SDK)。SDK 包括一系列的综合性工具、文档和教程以及帮助你立即投入开发的示例程序，此外还包括创建 Android 应用程序所需要的 Java 库，所有这些构成了应用框架的 API 接口。目前主流的桌面操作系统都支持该 SDK 的开发环境。

SDK 有以下主要特征：

- 调试器，可在设备或模拟器上调试应用程序。
- 内存和性能探测，帮助查找内存泄露和识别执行速度慢的代码。
- 设备模拟器，基于 QEMU(一个开源的虚拟机，可用于模拟不同的硬件平台)，准确度高，但有时运行速度较慢。用于设备间通信的命令行工具。
- 构建脚本和工具，用于打包并部署应用程序。

SDK 可以集成到 Eclipse 中，Eclipse 是一个流行的、功能丰富的开源 Java 集成开发环境(integrated development environment, IDE)。通过 Android 开发工具(Android Development Tools, ADT)插件可进行集成，该插件为 Eclipse 增加了一些新功能来创建 Android 项目、执行项目、配置项目和在模拟器或设备上调试应用程序，同时还生成 Android 应用程序包以便发布到 Android 市场上。当然该 SDK 也可以集成到其他的 IDE 中例如 NetBeans，只不过官方不对此提供支持。

注意：第 2 章将介绍如何利用 SDK 和 Eclipse 来创建开发环境。

SDK 和 Eclipse 的 ADT 插件会不断地进行更新，它们将增加一些新的特性和功能。因此，及时更新它们很有帮助。

除了附带丰富的文档之外，Android SDK 还提供了大量的示例应用程序。同时你还可以到一个网站查看开发指南和应用程序框架的所有模块的 API 引用，其网址为 <http://developer.android>。

com/guide/index.html。

1.6 开发人员社区

Android 的成功部分归功于开发人员社区, 它通过网站吸引了世界各地的开发人员。最受开发人员追捧的 Android 开发人员组 <http://groups.google.com/group/android-developers>。当遇到看似无法解决的问题时, 这是一个发问和寻求帮助的首选地方。有各种各样的 Android 开发人员访问该组, 从系统程序员、应用程序开发人员到游戏开发人员, 有时还有谷歌的 Android 工程师帮你解答问题。该开发社区的注册是免费的, 所以强烈建议你现在就看看这个社区。它除了让你找到一个地方提问, 它还是一个搜索已解决问题和问题解决办法的好地方。所以, 当你提出问题之前, 最好先看看是否已经有了这个问题的答案。

每一个开发人员社区都有一个吉祥物图标。Linux 有企鹅图标, GNU 有 gnu 图标, 而 Mozilla 有着时髦的 Web 2.0 的狐狸图标。Android 也不例外, 它选择了一个绿色的小机器人作为它的图标, 如图 1-2 所示。

虽然它的颜色选择存在争议, 但是这个无名的小机器人已经在几个流行的 Android 游戏中出现了。尤其是出现在 Replica Island 中, 这是前谷歌开发人员 Chris Pruett 作为他的“20 percent”项目创建的一个免费的开源平台。“20 percent”代表谷歌的员工在他们自己选择的项目上所花费的一周中的一天。

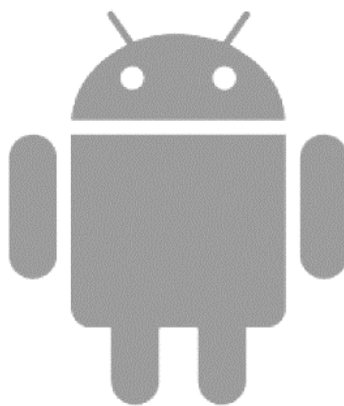


图 1-2 Android 的无名吉祥物图标

1.7 设备, 设备, 设备

Android 系统没有被锁定到一个单一的硬件生态系统中, 许多著名的手机制造商(如 HTC、摩托罗拉、三星和 LG)都加入了 Android 的时尚潮流中, 并开发了多种 Android 手机。除了手机, 也有一系列基于 Android 的平板电脑。当然一些关键的功能是所有设备所共有, 这使得游戏开发进程容易一点。

1.7.1 硬件

Android 对硬件没有最低的要求, 但是谷歌建议使用下面的最低硬件配置。基本上所有的 Android 设备都能满足这些配置, 而且多数设备甚至远远高于该配置:

- 128MB RAM: 这是一个最低的限度, 目前高端设备已经支持 1GB 的 RAM, 而且如果摩尔定律仍然适用, RAM 大小不断增加的趋势并不会很快停止。

- 256MB 的闪存：这是用于存储系统图像和应用程序所要求的最低配置。长期以来，缺乏足够的内存是 Android 用户最大的抱怨，因为第三方的应用程序只能安装到内存。不过 Froyo 版本的发布改变了这个局面。
- 微型 SD 存储卡：大多数设备会有几个 G 字节的 SD 存储卡，用户也可以更新到更大容量的 SD 卡。
- 16 位彩色 QVGA(Quarter Video Graphics Array)TFT(Thin Film Transistor)液晶触摸屏：在 Android 1.6 版本之前只支持 HVGA 触摸屏(480×320 像素)，之后开始支持更高和更低分辨率的屏幕。目前，高端手机设备支持 WVGA 屏幕(800×480、848×480 和 852×480 像素)，一些低端设备支持 QVGA(320×280 像素)屏幕。平板电脑的屏幕有不同的大小，通常是 1 280×800，而 Google TV 支持 HDTV 的 1 920×1 080 分辨率！虽然许多开发人员都倾向于认为每个设备都有触摸屏，但是实际上不是这样。Android 正在进军机顶盒和具有传统监视器的类似于 PC 的设备行业，它们都没有手机或平板电脑那样的触摸屏输入。
- 专用按键：这些按键用于导航功能。手机总是提供与标准的导航命令对应的按键，例如主页键和返回键，它们通常独立于屏幕上的触摸命令。Android 的硬件种类十分广泛，所以不要假定设备一定有某些按键。

当然，大多数 Android 设备的配置都远高于上述最低要求。几乎所有的手机都有 GPS、加速计和罗盘，许多还具有距离和光感传感器。这些外设给游戏开发商提供了更多的方法来让用户参与游戏的互动，后面我们将会看到一些例子。一些设备甚至还具有 QWERTY 键盘和轨迹球，后者常见于 HTC 的设备。同时摄像头也是多数设备所必备的，有些手机或平板电脑甚至有两个摄像头，一个位于后面，一个位于前面，用于视频聊天。

专用的图形处理单元(graphics processing units, GPUs)对游戏开发起到了关键的作用。最早运行 Android 的手机有一个与 OpenGL ES 1.0 兼容的 GPU，而现在设备的 GPU 已经能与老的 Xbox 或 PlayStation 2 相媲美，并且支持 OpenGL ES 2.0。如果没有图形处理器可用，系统一般会转而提供一个叫 PixelFlinger 的渲染软件来提供支持。许多廉价的手机都是依靠这个软件来进行图形渲染，对于低分辨率的手机它已经足够快了。

除了图形处理器，现在的 Android 设备也有专用的音频硬件。许多硬件平台通过特殊的电路用于在硬件上解码诸如 H.264 等不同的媒体格式，还有电话、无线网络和蓝牙等硬件组件用于通信连接。Android 设备上所有的这些硬件模块通常都集成在一个单一的系统芯片上(system on chip, SoC)，嵌入式硬件也采用这种系统设计方法。

1.7.2 设备的范围

最早是 G1 设备。开发人员热切地等待更多的设备出现，而几种差别不大的手机很快就问世了，它们被认为是“第一代手机”。多年以后，硬件功能越来越强大，现在手机、平板电脑和机顶盒的范围十分广泛，从具有 2.5" QVGA 屏、在 500MHz ARM CPU 上运行一个软件渲染器的设备一直到具有 1GHz 双核 CPU 和能够支持 HDTV 的非常强大的 GPU 的设备。

我们已经讨论过分段存储的问题，但作为开发人员不仅要应对分段存储问题，还需要处理这些变化范围很大的屏幕尺寸、能力和性能。最好的方法是理解最低硬件需求，并将其作为游

戏设计和性能测试的基准。

1. 最低可行目标设备

截止到 2011 年 10 月 3 日，只有不到 3% 的 Android 设备在运行低于 Android 2.1 的版本。这一点很重要，因为它意味着现在开始开发的游戏最低必须支持 API level 7(2.1)，因此在完成以后它可以被 97% 的 Android 设备使用。这并不是说您不能使用最新的功能！这么做当然可以，我们也将讲解具体的做法。您只是需要在设计游戏时提供某种回退机制，以便能够与低至 2.1 版本的系统兼容。当前的各版本使用情况可在 <http://developer.android.com/resources/dashboard/platform-versions.html> 查看，图 1-3 显示了在 2011 年年中收集的一张图。

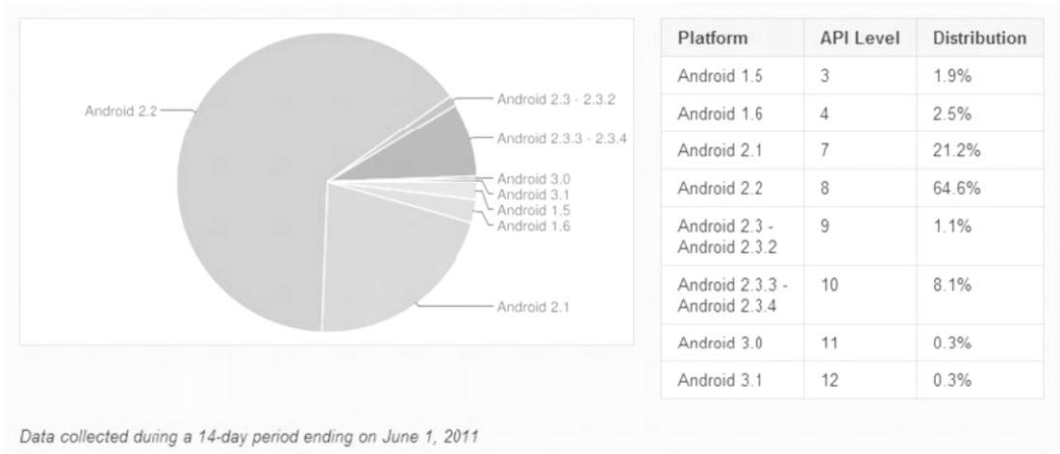


图 1-3 2011 年 10 月 3 日的 Android 版本分布情况

哪个基准设备可以作为最低目标设备？应该是第一个发布的 Android 2.1 手机：最初的摩托罗拉 Droid，如图 1-4 所示。虽然后来它被更新为 Android 2.2 系统，但是 Droid 仍然是一个广泛使用的设备，其 CPU 和 GPU 性能都有不错的表现。



图 1-4 摩托罗拉 Droid

最初的 Droid 在发布时被打造为第一个“第二代”设备，它是在第一批基于高通 MSM7201A 的模型(包括 G1、Hero、MyTouch、Eris 等)发布大约一年后发布的。Droid 是第一个屏幕分辨率超过 480×320 、并且有一个独立的 PowerVR GPU 的手机，也是第一个直接支持多点触摸的 Android 设备(不过它的双点触摸功能有一些问题，稍后会进行介绍)。

支持 Droid 意味着也支持具有以下特征的设备：

- CPU 速度位于 550MHz~1GHz 之间，硬件支持浮点运算
- 可编程的 GPU 支持 OpenGL ES 1.x 和 2.0
- WVGA 屏幕
- 支持多点触摸
- Android 系统采用 2.1 或 2.2+版本

Droid 是一个非常好的最低目标，因为它运行 Android 2.2，并且支持 OpenGL ES 2.0。它的屏幕分辨率与大多数屏幕尺寸为 854×480 的手机设备类似。如果某个游戏可以在 Droid 上运行良好，那么它很可能在 90% 的 Android 设备上运行良好。仍然会有一些老设备(甚至一些新设备)的屏幕尺寸为 480×320 ，所以考虑到这个尺寸并至少在这些设备上进行测试是很好的做法，但是从性能方面来看，要占领大部分 Android 市场，不太可能需要支持比 Droid 性能差很多的设备。

2. 前沿设备

Honeycomb 对平板电脑有不错的支持，而现在很明显，平板电脑正在成为一种首选的游戏平台。2011 年初的设备开始使用 NVIDIA Tegra 2 芯片，从那时起，手机和平板电脑开始使用快速的双核 CPU，甚至更加强大的 GPU 成为它们的必备品。由于产品变化十分快速，所以在写书的时候很难讨论什么是“现代的”。但是，在撰写本书时，主流设备普遍具有高速处理器、大量存储空间、大量 RAM、高分辨率和双手多点触摸支持，一些新模型甚至还有 3D 立体显示。

Android 设备中最常见的 GPU 是 Imagination Technologies 的 PowerVR 系列、Qualcomm 的集成了 Adreno GPU 的 Snapdragon 以及 NVIDIA 的 Tegra 系列。PowerVR 现在有了新版本：530、535 和 540。模型号相差不大，不过不要被迷惑了。与前代相比，540 异常快速，并且三星的 Galaxy S 系列和谷歌的 Nexus S 都使用了这个 GPU。而 530 是用于 Droid，535 用于其他几个模型中。高通的 GPU 可能是使用最广泛的 GPU，几乎每个 HTC 设备都在使用。Tegra GPU 面向的是平板电脑，但是在一些手机中也有应用。这 3 种相互竞争的芯片架构能力非常相似，都十分强大。

三星的 Galaxy Tab 10.1(如图 1-5 所示)是目前 Android 平板电脑事实上的标准，它支持以下功能：

- NVIDIA Tegra 2 双核 1GHz CPU/GPU
- 可编程 GPU，支持 OpenGL ES 1.x 和 2.0
- 1280×800 屏幕
- 支持多达 10 点的多点触摸
- Android Honeycomb 3.1

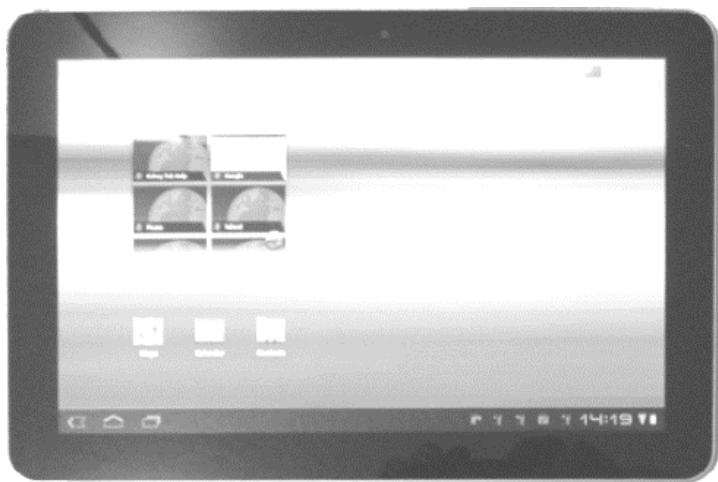


图 1-5 三星 Galaxy Tab 10.1

支持与 Galaxy Tab 10.1 类似的平板电脑对于维持越来越多的使用这种技术的用户十分重要。从技术上讲，支持它与支持其他任何设备没有两样。但是，谷歌和三星承诺在发布该设备后的 18 个月内对它使用最新版本的 Android，所以它很可能在第一批部署中获得最新的 Android OS 升级和功能。平板电脑大小的屏幕是另一个可能需要在设计阶段进行额外考虑的方面，后面将对此进行详细讨论。

3. 下一代设备

尽管设备制造商在极力掩盖他们下一代手机的神秘性，但我们仍可得到一些关于它们的信息。

所有未来设备的趋势是将拥有更多的内核、更大的内存、更好的 GPU 和更高的屏幕分辨率。不断有参数更高的芯片问世，而 Android 本身也在成熟，不只表现在性能越来越好，而且随着每一个后续版本的发布，其功能也在不断增加。硬件市场的竞争越来越激烈，而且没有减缓的迹象。

虽然 Android 最初是用于手机上的，但是它很快发展到可以用在不同类型的设备上，包括电子书阅读器、机顶盒、平板电脑、导航系统和插入到基座上即可当做 PC 使用的混合式 hybrid 手持设备。要创建可以在任何设备上运行的 Android 游戏，开发人员需要考虑到 Android 的本质，即可以嵌入到任何设备上运行的通用 OS。我们不能假定 Android 只是会用到现在的设备类型上。从 2008 年以来，Android 的增长十分迅速，涉及的设备十分广泛，所以很难判断它的极限在哪里。

不管将来是什么样子，Android 会继续流行一段时间了。

4. 游戏控制器

由于不同 Android 设备其输入方法也各异，因此很少有厂家生产专门的游戏控制器。因为 Android 系统没有提供这种游戏控制器的 API 接口，所以游戏开发商不得不单独使用由游戏控制器制造商提供的 SDK 进行单独集成。

例如，有一个游戏控制器 Zeemote JS1，它具有一个摇杆和一组按钮，如图 1-6 所示。



图 1-6 Zeemote JS1 游戏控制器

该控制器与设备通过蓝牙进行通信。游戏开发人员通过 Zeemote SDK 提供的单独 API 来集成对控制器的支持。目前已有很多 Android 游戏支持使用该控制器。

从理论上讲，用户也可以通过蓝牙将 Android 设备与任天堂的 Wii 控制器相连。目前，已经有一些这样的原型开发存在，只不过没有官方的 SDK 支持，这给集成工作带来了不少的难度。

如图 1-7 所示的 Game Gripper 是专门为摩托罗拉 Droid 和 Milestone 手机设计的颇有创意的配件。这是一个简单的橡胶配件，它位于手机 QWERTY 键盘之上，以替代硬件游戏控制板的功能。游戏开发人员只需要添加键盘控制的功能就能完成与该配件的通信，而不需要添加其他特别的软件库以用于与 Gripper 通信。毕竟，它只是一个橡胶配件。

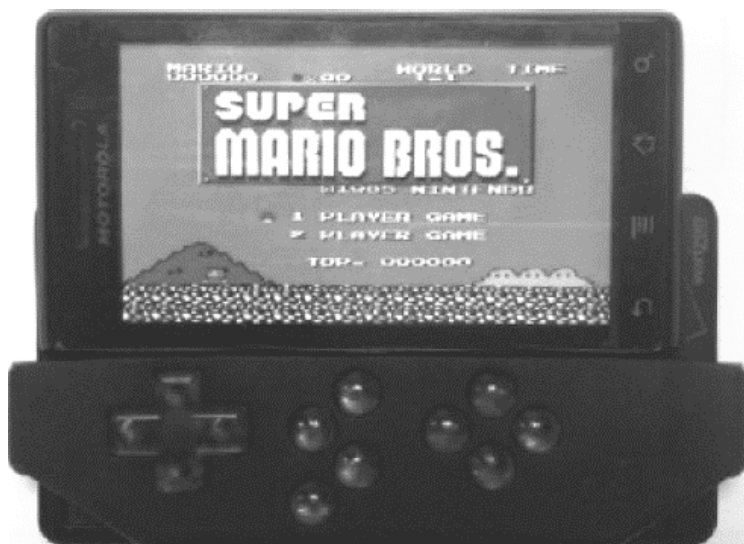


图 1-7 Gripper 游戏

游戏控制器也许对 Android 来说还有点陌生。不过，一些游戏已经集成对所选控制器的支持，并且得到了玩家的认同，所以还是应该考虑集成对这种外设的支持。

1.8 所有设备之间的兼容性

讨论了这么多关于手机、平板电脑、芯片集和外设等的内容,显然可以知道,支持 Android 设备市场与支持 PC 市场没有太大分别。屏幕大小的范围从很小的 320×240 一直到 1920×1080 (在 PC 监视器上可能更高)。在最低端的第一代设备上,仅使用了 500MHz ARM5 CPU 和功能十分有限且不具有太多内存的 GPU。而在最新的设备上有高带宽的多核 1-2GHz CPU、大规模并行 GPU 和大量内存。第一代手机的多点触摸系统很不可靠,不能检测到离散的触摸点。而新的平板电脑可以支持 10 个离散的触摸点,机顶盒则完全不支持任何触摸。那么开发人员该怎么办?

首先,这是需要弄清楚的。Android 本身有一个兼容性程序,决定了需要兼容 Android 的设备各个部分的最低要求。如果某个设备没有满足标准,则无法安装 Android Market 上相关的应用程序,这极大减小了开发人员的负担。在 <http://source.android.com/compatibility/overview.html> 上可以找到兼容性程序。

在兼容性站点上可以找到一个名为兼容性定义文档(Compatibility Definition Document, CDD)的文档,其中概述了 Android 兼容性程序。每次发布 Android 平台时,该文档都会更新,硬件制造商必须更新和重新测试它们的设备才能保持兼容。

CDD 的要求中与游戏开发人员有关的一些条目如下所示:

- 最低音频延迟(不固定)
- 最小屏幕尺寸(目前是 2.5")
- 最小屏幕密度(目前是 100dpi)
- 可接受的纵横比(目前是 4:3 到 16:9)
- 3D 图形加速(需要 OpenGL ES 1.0)
- 输入设备

即便不能理解上面列出的一些项目也不用担心,在本书后面的章节中将更详细地介绍这些主题。从这个列表中可以知道的是,可以设计出能够在大多数 Android 设备上运行的游戏。通过预先规划诸如游戏中的用户界面和一般视图等,使它们可以在不同的屏幕尺寸和纵横比下工作,并且理解您不只需要支持触摸功能,还需要支持键盘或其他输入方法,就可以成功地开发出兼容性非常好的游戏。对于不同的游戏,需要使用不同的方法才能在不同的硬件上实现良好的用户体验,所以没有一个可以解决这些问题的一劳永逸的办法。但是请放心:只要投入时间进行适当规划就可以得到良好的结果。

1.9 不同的手机游戏

在 iPhone 和 Android 进入游戏市场之前,游戏在市场上已经广泛存在。并且随着这种新型混合设备的出现,前景已经开始发生改变。游戏已不仅仅是孩子的专利,很多商业人士也在公共场合中用手机玩最流行的游戏。随手拿起报纸你就会发现,一些小的游戏开发商已在手机应用市场上获得巨大的财富,而已经功成名就的游戏开发商也很难跟得上手机设备的发展。我们作为游戏开发人员必须认识到这种改变并及时做出调整,接下来就看看这个新的生态系统带来了什么吧。

1.9.1 人手一台游戏机

智能手机无处不在，这也许是本章节中最关键的语句。从这一点出发就可以得出关于手机游戏的其他事实。

随着手机硬件价格的不断下降及新手机的计算能力不断提高，手机已经成为理想的游戏开发设备。现在，手机已经成为人们的一种必需品，它们的市场渗透力是巨大的。很多人把他们的老的、经典的手机换成新一代的智能手机，并且发现可以使用的应用程序多到不可思议。

以前，人们为了玩视频游戏，不得不在视频游戏系统和 PC 游戏机之间作出艰难的决定。现在，有一台智能手机就能免费获得游戏功能。不需要支付额外的费用(这里没有考虑你可能使用的数据计划)，新的游戏设备就能随时伴你左右。你只要从口袋或钱包里拿出它就可以开始玩游戏了，无须随身携带一个专用的系统，一切都在手机之中。

除了只携带一个设备就可以打电话、上网和玩游戏的特点以外，还有另外一个事实使得更多人可以在手机上玩游戏：即启动一个专门的电子市场应用程序，从中选择一个感兴趣的戏，然后就可以立即开始玩游戏了。这样，你就不会到商店买或通过 PC 下载游戏，结果却发现自己丢掉了把数据传输到手机上所需要的 USB 传输线。

目前，智能手机不断增长的处理能力对游戏开发商所能实现的功能也有一定的影响。即使中端的手机设备也已经能拥有与老式的 Xbox 和 PlayStation 2 同等的游戏体验。有了这么强大的硬件平台，我们也可以开始体验到更精美的、使用物理模拟的游戏，而这是一个有无限的创意空间的领域。

如前所述，伴随新设备而来的是新的输入方法。很多游戏已经开始利用多数 Android 设备都具有的 GPS 和罗盘等功能。另外，我们也看到加速计已成为游戏开发的主要特征了，而多点触摸也成为游戏体验的新方式。相比于经典的游戏控制台(Wii 除外)，这对游戏开发人员已是一个很大的改变。现在，手机大部分的功能都已被使用，但我们相信仍会有新的方式来使用这些功能。

1.9.2 随时上网

Android 手机通常还绑定很多其他的服务，除了打电话它还是一些流行的互联网网站带来了很大的流量。使用智能手机的用户可能一直都在给定时间连接着互联网(忽略硬件设计等问题导致的信号不好的情况)。

随时上网给手机游戏开启了一个新纪元。人们可以通过网络一起下棋、一起探索虚拟世界或在特定时间蓄意杀伤另一个城市的好朋友(社交游戏)。而这一切都可以发生在旅途中，在公交车上、在火车上或自己最心爱的公园的一角。

除了多人游戏功能，社交网络在移动游戏中也发挥着巨大的能量。游戏可以将你最新的得分分享到你的 Twitter 账户，或告知朋友你在你们两个人都喜欢的游戏中的最新成果。虽然快速增长的社交网络也存在于各种经典的游戏世界中(如，Xbox Live 或 PlayStation Network)，但是像 Facebook 和 Twitter 这种服务的普及率要高许多，所以用户也不必一次管理多个社交网络账号。

1.9.3 普通用户与游戏迷

智能手机的庞大用户群也意味着之前从没接触过 NES 控制器的人们会突然发现一个游戏世界。他们的头脑中对好游戏的看法与游戏爱好者的头脑中的看法相去甚远。

由于移动电话的特殊性, 普通用户更喜欢一些休闲的游戏, 这样当他们在车上或在快餐店排队的时候就能玩上几分钟。这样的游戏很像 PC 机上让人着迷的 flash 游戏, 在工作环境中玩 flash 游戏的人在感到背后有人盯着自己时会疯狂地按 Alt+Tab 键来切换程序。可以问问你自己: 每天你愿意花多少时间在手机上玩游戏? 你能想象自己在手机上“快速”地玩一局“文明”游戏?

肯定有人愿意为能够在手机上玩他们心爱的 Advanced Dungeons & Dragons 游戏等付出巨大的代价, 不过这种人很少, 从 iPhone 应用程序商店或 Android 市场的排行榜就可以看出来。最畅销的游戏一般都非常休闲并且对用户有很好的挑战性: 它会让你几分钟就可以玩一局, 同时又设计了不同的关卡能让你不断地玩下去。这种游戏一般会制作一个在线成绩系统, 让你可以炫耀自己的技术。但也许这是一个伪装成休闲游戏的大型游戏。为用户提供保存进度的简单方法, 就可以把一个庞大的 RPG 游戏作为益智游戏销售给他们。

1.9.4 市场很大, 开发人员很少

低门槛的特点吸引了众多业余爱好者和自由开发人员加入移动开发。在 Android 系统中, 这一门槛更低: 你只需要准备 SDK 并开始编写程序。你甚至不需要一台设备(但还是强烈建议至少有一部开发设备), 只要有模拟器就行。Android 的开放性在网络上也有很好的表现, 关于系统编程各个方面的信息都可在网上找到, 并且免费使用, 不必签署任何保密协议或等待官方的授权才能使用。

在撰写本书的时候, 市场上大部分成功的游戏都是由个人和小团队开发的, 而大公司还没有进军这个领域, 至少没有成功过。Gameloft 就是一个很好的例子。虽然在 iPhone 市场上它取得了不错的成绩, 但在 Android 市场上却不尽如人意, 以至于他们决定在自己的网站上出售游戏。Gameloft 可能对 Android 中没有数字版权管理方案(Digital Rights Management scheme, DRM)(现在 Android 上已经有了)感到不快, 因为这让他们的知名度又减少了一些。

Android 环境可以让闲暇的人进行大量的尝试和创新来寻求他们想要看到的“宝贝”, 包括新的游戏系统和游戏玩法。在经典的游戏平台如 PC 机或游戏机上, 这种尝试经常失败。不过, 在 Android 市场上有庞大的愿意尝试新想法的用户, 使他们注意到你的游戏会更加容易一些。

当然, 这并不是说你不需要为自己的游戏做宣传。你可以通过在各大博客或专业网站上做宣传, 因为很多 Android 的用户很热衷于去浏览此类网站了解最新和最好的游戏。

另一种让你的游戏增加曝光率的方法是获得 Android 市场的推荐, 这样你的应用程序就能在市场应用的首页列表上显示。许多开发人员报告说在获得推荐之后, 他们应用的下载量获得巨大的增长。如何获得推荐也许是一个谜。不过, 拥有一个好想法和以最漂亮的方法实现它是你最好的选择, 不管你是个人还是大的开发公司。

1.10 小结

Android 真是一个令人兴奋的东西。现在，你已经知道它的一些基本框架和开发环境。从开发的角度讲，它为我们提供了一个在软件和硬件上都非常有趣的系统，而且由于 **SDK** 是免费的，所以它的开发门槛很低。设备本身是一个强大的手持装备，并将使我们把视觉丰富的游戏展示给我们的用户。各种传感器的使用(如加速计)更是为我们创建游戏用户体验提供新方式；而且当我们完成游戏开发时，我们可以在几分钟之内部署给数百万的潜在玩家。听起来就让人兴奋，那就让我们开始动手编程吧！

第 2 章

从 Android SDK 开始

Android SDK 提供了大量的工具让我们可以立即开始编写应用程序。本章将介绍如何利用 Android 的 SDK 工具建立一个简单的应用程序，其主要步骤如下：

- (1) 搭建 Android 开发环境。
- (2) 在 Eclipse 中创建一个新项目并编写代码。
- (3) 在模拟器或设备上运行我们的应用程序。
- (4) 调试并配置应用程序。

接下来我们就先搭建开发环境。

2.1 搭建开发环境

Android SDK 非常灵活，可以将多个开发环境很好地集成起来。我们可以通过命令行工具来操作。但为了使用起来更方便，我们选择使用一个简单的可视化集成开发环境(integrated development environment, IDE)。

先下载并按顺序安装以下软件：

- (1) Java 开发工具包(JDK)的版本 5 或版本 6，建议使用版本 6。
- (2) Android 软件开发包(Android SDK)。
- (3) 针对使用 Java 的开发人员的 Eclipse，版本 3.4 或更新。
- (4) Eclipse 所需要的 Android 开发工具(ADT)插件。

下面开始各部分的安装。

注意：由于下载网址经常变更，因此无法提供下载链接地址。请使用熟悉的搜索引擎进行搜索并下载它们。

2.1.1 安装 JDK

请选择与操作系统的版本号对应的 **JDK** 进行下载。对于大多数系统来说它就是一个安装器或软件包，因此下载安装不会有什么障碍。安装完成之后，需要将环境变量 **JDK_HOME** 添加到你的系统中，其值为 **JDK** 的安装路径。另外还要将 **\$JDK_HOME/bin**(Windows 系统使用 **%JDK_HOME%\bin**) 添加到环境变量 **PATH** 中。

2.1.2 安装 Android SDK

对于三个主流桌面操作系统，**Android SDK** 都可使用，请选择一个适合自己平台的版本并下载下来。**SDK** 一般是 **ZIP** 或 **tar gzip** 文件的形式，把它解压到一个合适的文件夹中(例如，Windows 下的 **c:\android-sdk** 或 Linux 系统下的 **/opt/android-sdk**)。**SDK** 提供的几个命令行工具都位于 **tools/**文件夹下。接着再添加一个环境变量 **ANDROID_HOME** 到系统中，其值为 **SDK** 的安装路径，同时在 **PATH** 环境变量中增加字段 **\$ANDROID_HOME/tools**(Windows 系统使用 **%ANDROID_HOME%\tools**)，这样在需要时可以很容易地在 **shell** 中使用命令行。

在执行以上步骤后，你就完成了一个基本的安装，其中包含创建、编译和部署 **Android** 项目所需要的基本命令行工具。不过，这还不够进行应用程序的开发，还需要一个 **SDK** 和 **AVD** 管理器，这是用于安装 **SDK** 组件和创建模拟器使用的虚拟设备。该管理器是一个包管理器，很像 Linux 下的包管理工具，它允许你安装下面的组件：

- **Android 平台**：所有正式发布的 **Android** 其 **SDK** 都有一个平台组件，包括运行时库、模拟器使用的系统图像和版本适用的工具。
- **SDK 增件**：增件通常是外部的库和工具，并不是某个平台特有的。例如一些 **Google API**，它们可以将 **Google maps** 集成到你的应用程序中。
- **Windows 系统 USB 驱动程序**：这是在 **Windows** 系统下的物理设备上运行和调试应用程序所必需的。在 **Mac OS X** 和 **Linux** 系统中不需要特殊的驱动程序。
- **示例**：每个平台都有一个特定于平台的示例集，这是了解如何使用 **Android** 运行时库实现特定目标的很好的资源。
- **文档**：这是最新的 **Android** 框架 **API** 文档的本地副本。

作为开发人员，我们总希望把所有的组件都安装上以便能够使用它们的全部功能。为此，我们需要启动 **SDK** 和 **AVD** 管理器。在 **Windows** 系统中，我们启动位于 **SDK** 的根目录下面的可执行的 **SDK manager.exe** 文件。而在 **Linux** 或 **Mac OS X** 系统中，我们启动 **SDK** 的 **tools** 目录下的 **android** 脚本文件。

在第一次启动时，**SDK** 和 **AVD** 管理器将会连接到包服务器并获取可用软件包的列表。然后它会弹出一个如图 2-1 所示的对话框，从中可以安装各个软件包。只要选中 **Accept All** 单选按钮并单击 **Install** 按钮即可，之后你便可以去喝一杯咖啡，因为管理器需要花一些时间来完成安装。

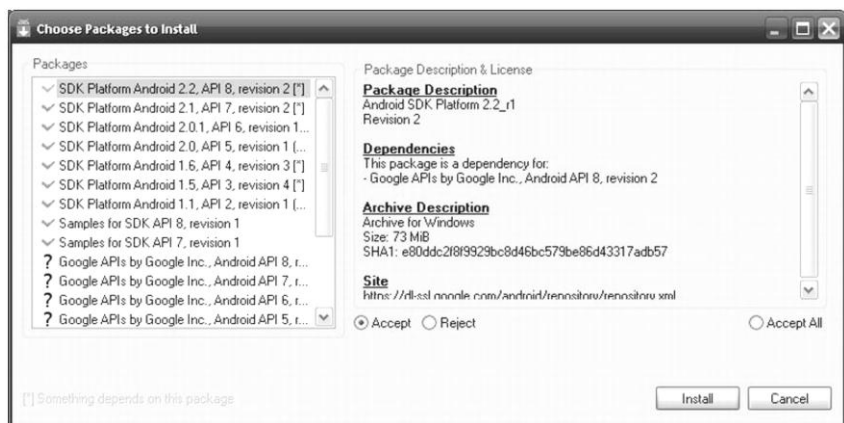


图 2-1 第一次启动 SDK 和 AVD 管理器

你可在任何时候使用 SDK 和 AVD 管理器来更新组件或安装新的组件。该管理器也可用于创建新的 AVD，之后你在模拟器上运行和调试应用程序的时候就会用到它们。

一旦该安装完成，就可以进入到搭建开发环境的下一步。

2.1.3 安装 Eclipse

Eclipse 有几个不同的版本。对 Android 开发人员而言，建议使用 Eclipse for Java Developers 3.6 版本。同 Android SDK 一样，Eclipse 也是一个 ZIP 或 tar gzip 文件包的形式，将其解压到一个文件夹即可。一旦软件包被解压缩后，可以在桌面建立一个快捷方式，指向 Eclipse 安装的根本目录下的可执行文件 eclipse。

当第一次启动 Eclipse 的时候，系统提示选择一个工作区目录，如图 2-2 的对话框所示。

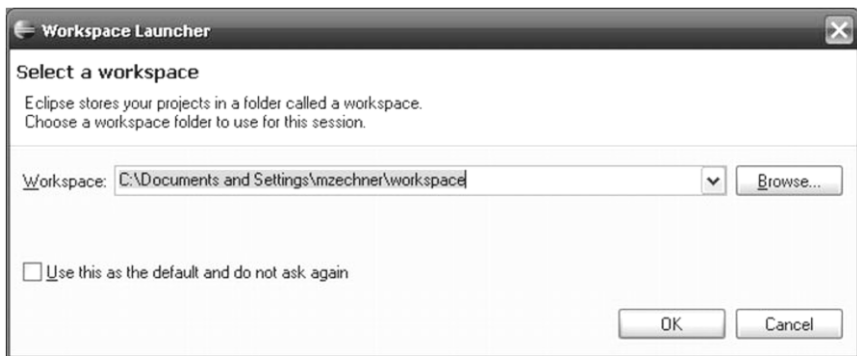


图 2-2 选择工作区

Eclipse 的工作区目录其实就是一个存放项目的文件夹。可以为所有项目建立一个工作区，也可以建立多个工作区，且在每个工作区中存储几个项目，这完全取决于你。本书的所有示例项目全放在一个工作区中，你可以在该对话框中指定该工作区。下面，我们首先建立一个空的工作区。

接下来 Eclipse 会出现一个欢迎界面，可以忽略并关闭它。然后便进入了默认的 Eclipse Java 视图，我们将会在后面更深入地学习它，现在只需要运行它就可以了。

2.1.4 安装 ADT Eclipse 插件

安装过程的最后一步就是安装 ADT Eclipse 插件。Eclipse 是基于插件体系结构的，用于通过很多第三方的插件来扩展其功能。ADT 插件可以将很多 Android SDK 工具和 Eclipse 的功能结合起来，这样我们就可以不使用所有的 Android SDK 的命令行工具，因为 ADT 插件已经将它们透明地集成到 Eclipse 工作流中了。

该插件可通过手动安装到 Eclipse 中，只需要将该插件的 ZIP 文件中的内容放到 Eclipse 的插件文件夹中即可；也可以通过 Eclipse 的插件集成管理工具来进行安装。这里我们使用第二种方法。

(1) 为安装新的插件，单击 Help | Install New Software...，这将会打开一个安装对话框。在该对话框中，你可以选择需要安装的插件的地址。首先必须添加包含所需 ADT 插件的插件存储库，然后单击 Add 按钮，看到如图 2-3 所示的对话框。

(2) 在第一个文本输入框中输入该插件存储库的名称，如“ADT 库”等。在第二个输入框中输入该插件的 URL 地址。对于 ADT 插件，其 URL 为 <https://dl-ssl.google.com/android/eclipse/>。注意该 URL 可能会随版本的变化而不同，请到 ADT 插件的网站确定最新的链接地址。

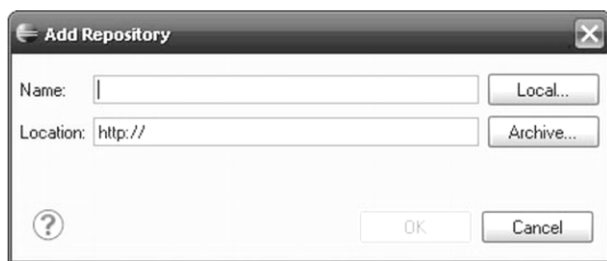


图 2-3 添加一个存储库

(3) 完成该对话框后，会返回到安装对话框。此时会列出存储库中所有的可用插件，选中 Developer Tools 复选框并单击 Next 按钮。

(4) Eclipse 将计算所有必要的依赖关系，并在一个新的对话框中列出那些将要安装的插件及其依赖关系，单击 Next 按钮确认即可。

(5) 接着会弹出另一个对话框，提示你是否同意安装该插件的许可协议。当然，你必须接受，并单击 Finish 按钮开始进行安装。

注意：在安装过程中，你可能会被要求确认安装没有签名的软件。不必担心，该插件不需要签名认证，同意并继续安装即可。

(6) 最后，Eclipse 会询问你是否重启以接受所有更新，可以选择重启，也可以选择只更新不重启。为了安全起见，一般选择重启 Eclipse。

完成所有的对话框操作后你会回到初始的 Eclipse 界面。这时会发现工具栏多了一些针对 Android 的新按钮，它可以直接在 Eclipse 中启动 SDK 和 AVD 管理器以及创建新的 Android 项目。图 2-4 展示了这些新的工具栏按钮。



图 2-4 ADT 工具栏按钮

位于左边的第一个按钮用于打开 SDK 和 AVD 管理器，

第二个按钮是创建新 Android 项目的快捷键，其他两个按钮分别用于创建新的单元测试项目和 Android 清单文件(本书中不会使用该功能)。

完成 ADT 插件安装的最后一步是需要告知插件 Android SDK 安装的位置。

- (1) 打开 Window | Preferences，并在打开的对话框的树型视图选择 Android 选项。
- (2) 单击右边的 Browse 按钮，选择 Android SDK 的安装根目录。
- (3) 单击 OK 按钮关闭对话框，接下来就可以创建第一个 Android 应用程序了。

2.1.5 Eclipse 快速浏览

Eclipse 是一个开源的 IDE，一般使用 Java 语言进行开发，但也可以使用其他语言进行开发。由于 Eclipse 的基于插件的架构，很多功能不断扩展，现在也可以开发纯 C/C++、Scala 或 Python 的项目。它的功能扩展好像永无止境，甚至插件可进行现有的 LaTeX 项目的开发——这与平常的代码开发只是略微相似。

Eclipse 的工作区可以有一个或多个项目同时存在。前面我们已经定义了一个工作区，所有新建的项目将存放于这个工作区目录下。同时，所有使用工作区时用于定义 Eclipse 界面的配置文件和其他信息文件都存放在该目录下。

Eclipse 的用户界面(UI)主要围绕以下两个概念：

- 视图：一种简单的 UI 组件，例如源代码编辑器、输出控制台或项目资源管理器。
- 透视图：一系列特殊的视图，用于特殊的开发任务，例如编辑和浏览源代码、调试、分析和使用版本控制存储库进行同步等。

Eclipse for Java Developers 附带了很多预定义透视图，其中我们比较感兴趣的是 Java 和 Debug 透视图。Java 透视图如图 2-5 所示，它带有一个位于左边的 Package Explorer、一个位于中间的源代码编辑视图(我们没打开源代码文件的时候，它是空的)、一个位于右边的 Task List 视图、一个 Outline 视图和一个选项卡视图，它还包括一些子视图，如 Problems 视图、Javadoc 视图和 Declaration 视图，如图 2-5 所示。

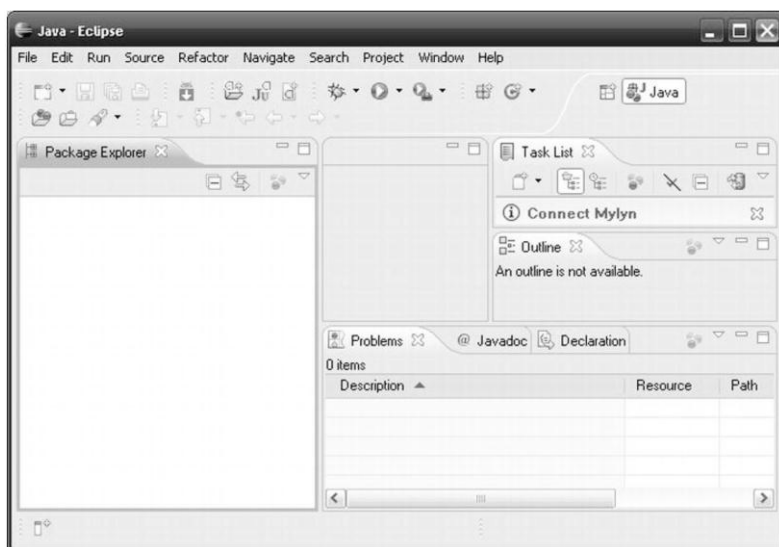


图 2-5 Eclipse 的 Java 透视图

可以通过拖放在透视图自由排列这些视图的位置，也可以重新定义其大小。除此之外，还可以添加或者删除一些视图。要添加视图，单击 **Window | Show View**，并从列表中选择要添加的视图，或者选择 **Other...** 来获得更多的可选视图。

如果想切换到另一个透视图，单击 **Window | Open Perspective** 并选择所要的透视图即可。一种快速切换已打开透视图的方法是单击 Eclipse 的左上角，在此你能看到已打开的透视图和当前激活的透视图。如图 2-5 所示，Java 透视图是一个唯一已打开并激活的透视图。一旦打开了其他的透视图，它们也会出现该位置。

图 2-5 中的工具栏也是一种视图。在不同的透视图下，这个工具栏可能会发生变化。在我们安装 ADT 插件时，一些按钮被也添加到工具栏中。一般情况下，新插件的安装都会带来一些新的视图和透视图。例如，在安装 ADT 插件时，我们看到在 Java Debug 透视图中就添加了 DDMS(Dalvik Debugging Monitor Server)透视图(专门用于调试和分析 Android 应用程序)，同时还添加了一些其他视图，如 LogCat 视图(用于显示附加的任何设备或模拟器的各种日志信息)。

一旦你熟悉了透视图和视图这两个概念，Eclipse 就变得容易多了。在下面章节编写 Android 游戏的时候，会探讨其中的一些视图和透视图。但我们无法将其全部都讲解一遍，因为实在是太多了。因此当你有需要的时候，建议参考 Eclipse 的帮助系统。

2.1.6 一些实用的 Eclipse 快捷键

每个新的 IDE 都需要花一些时间来学习并熟悉。使用了这么多年的 Eclipse 以后，我们发现下面列出的快捷键可以显著加快软件的开发过程。在表述这些快捷方式时使用了 Windows 术语，所以 Mac OS X 用户应该在合适的地方替换命令和选项：

- 当光标位于函数或字段中时，按 **Ctrl+Shift+G** 组合键将在工作区中搜索所有引用该函数或字段的地方。例如，如果想要查看某个函数是否被调用，只需要单击将光标移动到该函数中，然后按 **Ctrl+Shift+G** 组合键。
- 当光标位于要调入的函数时，按 **F3** 键将跟踪该调用，并显示声明和定义该函数的源代码。把这个快捷键和 **Ctrl+Shift+G** 组合键结合起来使用可以方便地在 Java 源代码中进行导航。
- 按 **Ctrl+空格键** 可以自动完成正在键入的函数或字段的名称。在键入前几个字符后即可按该快捷键。如果有几个可能匹配的名称，会在一个框中列出它们。
- **Ctrl+Z** 组合键执行撤消操作。
- **Ctrl+X** 组合键执行剪切操作。
- **Ctrl+C** 组合键执行复制操作。
- **Ctrl+V** 组合键执行粘贴操作。
- **Ctrl+F11** 键运行应用程序。
- **F11** 键调试应用程序。
- **Ctrl+Shift+O** 组合键组织当前 Java 源文件中的 **import** 语句。
- **Ctrl+Shift+F** 组合键设置当前源文件的格式。
- **Ctrl+Shift+T** 组合键跳转到任意 Java 类。
- **Ctrl+Shift+R** 组合键跳转到任意资源文件，即图像、文本文件等。

Eclipse 中还有许多十分有用的功能,但是掌握这些基本的键盘快捷键就可以显著加快游戏开发进度,并简化 Eclipse 的使用。Eclipse 也是高度可配置的。以上列出的键盘快捷键都可以在 Preferences 中被重新分配给其他键。

2.2 Android 环境下的 Hello World

完成开发环境的搭建后,我们就可以开始创建第一个 Android 项目了。ADT 插件的安装给我们提供了很多向导,这使创建 Android 项目变得容易多了。

2.2.1 创建项目

创建 Android 项目有两种方法。第一种是右击 Package Explorer 视图(如图 2-4 所示),并选择 New | Project...。在出现的新对话框中,选择 Android 条目下的 Android Project。可以看到在该对话框中有很多关于项目创建的其他选项。这是在 Eclipse 下一种很标准的创建项目的方法。在确认对话框之后,Android 项目向导就打开了。

另一个方法就更简单了,只需要单击按钮就可以创建 Android 项目了(如图 2-4 所示)。

当进入 Android 项目向导对话框后,需要做出一些设置:

(1) 首先,必须定义该项目的名称,如“hello world”。常用的约定是名称的所有字母都采用小写。

(2) 其次,必须指定创建目标。现在,选择 Android 1.5 作为创建目标,它是最低的要求,而且现在还不需要任何其他花哨的功能,如多点触摸等。

注意:在第 1 章中已经介绍了随着新的 Android 版本的发布,新的类也不断地加入 Android 框架 API 中。选择什么样的创建目标就意味着你能够在应用程序中使用哪个版本的 API,如果选择了 Android 3.1 版本,就可以使用最新的 API 功能。但这可能会带来一些危险,因为当应用程序在一台老版本系统(例如 Android 1.5)的设备上运行时,应用程序可能会因为使用了只有 3.1 版本才可用的 API 而崩溃。所以,需要在运行时检测设备支持的 SDK 版本,只有在设置上的 Android 版本号支持时才使用 3.1 版本的功能。听起来这似乎有些令人讨厌,但当你阅读第 5 章时,就会知道一个好的应用程序架构就能够容易启用或禁用这些与版本相关的功能而没有崩溃的风险。

(3) 接着必须为应用程序命名(例如,Hello World),为 Java 包命名(所有的代码文件都包含在该包内,如 com.helloworld)以及为活动命名。活动有些类似于桌面操作系统的窗口或对话框,在此我们将其命名为 HelloWorldActivity。

(4) Min SDK Version 字段用于指定应用程序运行所需要的最低 Android 版本,这个不是必需的,但最好还是指定它。SDK 版本的编号从 1(1.0)开始,并随版本升级不断增加。到 1.5 版本的时候,其值已为 3 了。记住,必须先指定一个创建目标,即使比最低 SDK 版本还高也没关系。这样系统允许应用程序运行在高 API 的设备里,但也可以部署到较老的 Android 版本(当然必须确保调用的 API 方法是该 SDK 所支持的)。

(5) 单击 Finish 按钮,创建你的第一个 Android 项目。

注意：设置 SDK 最低版本时，要记住该应用程序只能运行在一个与它相同 Android 版本或更高 Android 版本的设备上。当一个用户通过 Market 应用程序浏览 Android 电子市场时，他只能看到那些带有合适的最低版本的 SDK 的应用程序。

2.2.2 进一步分析项目

在 Package Explorer 中，现在可以看到一个名为“hello world”的项目。完全展开它你会看到如图 2-6 所示的界面，这是一般 Android 项目的结构。下面我们来进一步分析它。

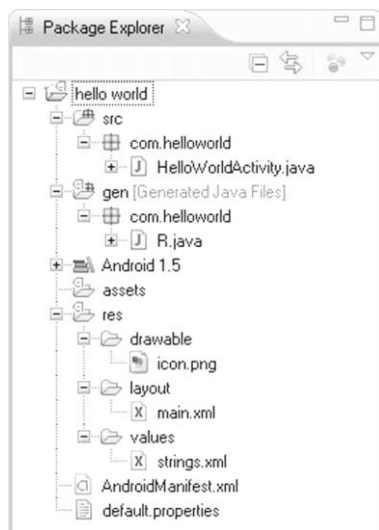


图 2-6 Hello World 项目结构

- **AndroidManifest.xml** 用于描述应用程序。它定义了应用程序中包含的活动和服务，支持的 Android 最低版本和用到的所有权限(例如，SD 卡权限和网络权限)。
- **default.properties** 存储创建系统的各项设置。我们不需要更改它，必要的时候 ADT 插件会帮我们修改它。
- **src/**包含所有的 Java 源代码文件。注意它的包名与你在 Android 项目向导中指定的一样。
- **gen/**包含 Android 创建系统生成的 Java 源文件。你不能修改它，必要时它会被自动更新。
- **assets/**用于存储各种应用程序中需要的文件(例如配置文件或音频文件等)，这些文件会打包在 Android 应用程序中。
- **res/**包含应用程序所需的各种资源文件，如 icons、用于国际化的字符串文件和用于界面布局的 XML 文件。它们同样打包于应用程序中。
- **Android 1.5** 告知我们建立了一个 Android 1.5 版本的项目，这是一个标准的 JAR 文件，保存 Android 1.5 的 API 类库。

在 Package Explorer 视图中，其实还隐藏了一个名为 **bin/**的目录，它存储编译生成的代码文件，这些代码文件将用于部署到设备或模拟器上。和 **gen/**文件夹一样，我们也无须关心它的内容。

通过右击 Package Explorer 视图中的文件夹，并选择 **New** 和想要创建的资源类型，我们可以轻松地将新的源文件、文件夹和其他资源添加到 Package Explorer 下的文件夹中，但现在先不添加。接下来我们对源代码做一些小小的修改。

2.2.3 编写应用程序代码

到现在为止我们还没有写过一行代码，下面就开始编写代码。Android 项目向导为我们创建了一个名为 `HelloWorldActivity` 的模板活动类，当在模拟器或设备上运行该应用程序的时候，它将会显示出来。双击 `Package Explorer` 视图中的该文件打开类的源代码，将模板程序替换为程序清单 2-1 所示的代码。

程序清单 2-1 `HelloWorldActivity.java`

```
package com.helloworld;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class HelloWorldActivity extends Activity
                                implements View.OnClickListener {

    Button button;
    int touchCount;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        button = new Button(this);
        button.setText("Touch me!");
        button.setOnClickListener(this);
        setContentView(button);
    }

    public void onClick(View v) {
        touchCount++;
        button.setText("Touched me " + touchCount + " time(s)");
    }
}
```

下面分析程序清单 2-1 以帮助你理解它的作用。具体的细节我们留到后面的章节再介绍，现在只需要对这段代码有一个大概的认识。

源代码文件以标准 Java 包的声明和几个 `imports` 语句开始，很多 Android 的框架类都包含在此 `android` 包中。

```
package com.helloworld;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
```

接下来，我们定义了一个继承于基类 `Activity` 的 `HelloWorldActivity` 类，`Activity` 由 Android 框架 API 提供。`Activity` 非常类似于经典桌面 UI 的窗口，它基本占据了整个屏幕(除了 Android UI 顶部的通知栏)。同时，它还实现了 `OnClickListener` 接口。如果你有使用其他 UI 工具包的经

```
public class HelloWorldActivity extends Activity
    implements View.OnClickListener {
```

我们在 Activity 中定义了两个成员，一个 Button 和一个用于记录按钮被单击次数的整数。

```
    Button button;
    int touchCount;
```

每一个 Activity 必须实现一个抽象 Activity.onCreate() 方法，当该活动首次启动时，Android 系统会调用该方法。它替代了创建类的实例时通常会使用的构造函数，并且强制调用 onCreate() 方法作为方法主体中的第一条语句。

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

然后创建一个 Button 并设置其初始文本。Button 是 Android 框架 API 提供的众多控件中的一种，它与 Android 中所谓的 View 的含义是相同的。注意，Button 是我们的 HelloWorldActivity 类的一个成员变量，后面我们需要引用它。

```
        button = new Button(this);
        button.setText("Touch me!");
```

接下来的一行主要是设置 Button 的 OnClickListener。OnClickListener 是一个回调接口，只有一个方法 OnClickListener.onClick()，当 Button 被按下的时候它就会被调用。为了侦听该按钮，需要让 HelloWorldActivity 实现该接口，并将其注册为 Button 的 OnClickListener。

```
        button.setOnClickListener(this);
```

onCreate() 方法的最后一行是将按钮设置为 Activity 的内容 View。View 是可嵌套的，Activity 的内容 View 就是整个层次结构的根。本例中，我们只是将 Button 设置为 View 并在 Activity 中显示出来。为了简单起见，我们就不详细讲解 Activity 在该内容 View 中的布局方式。

```
        setContentView(button);
    }
```

下一步就是实现 OnClickListener.onClick 方法，它包含于 Activity 之中。每当 Button 按下时，该方法就会被调用。在该方法中，我们让 touchCount 不断增加并将 Button 的文本设置为新的字符串。

```
    public void onClick(View v) {
        touchCount++;
        button.setText("Touched me" + touchCount + "times");
    }
```

最后，总结一下 Hello World 应用程序。我们先建立一个带有 Button 的 Activity，然后每当 Button 被按下时就让 Button 显示该次数(这可能不是一个能让人兴奋的应用程序，但它是其他好程序的基础)。

注意，我们从没有手动编译过项目。因为每当我们添加、修改或删除源文件或资源的时候，Eclipse 中的 ADT 插件都自动帮我们编译了。编译后生成了一个可部署到模拟器或设备的 APK

文件，该文件位于项目的 `bin/` 文件夹下。

以下我们将会使用该应用程序在模拟器或设备上运行和调试。

2.3 运行和调试 Android 应用程序

一旦写好一个应用程序的时候，就需要运行和调试，或找出存在的问题或陶醉于它的出色表现。一般我们有两种方法来操作：

- 将应用程序运行在一台通过 USB 与 PC 机相连的真实设备上。
- 将应用程序运行在 SDK 自带的模拟器上并进行测试。

上述两种方法都需要进行一些设置，然后才可以看到运行起来的应用程序。

2.3.1 连接设备

在连接设备以进行测试之前，我们必须确保设备已被操作系统所识别，在 Windows 系统中这包括安装合适的驱动程序，它一般在安装 SDK 的时候也被安装了。你只需要连上你的设备，然后在 SDK 的安装目录 `driver/` 文件夹下找到 Windows 系统的标准驱动程序即可。对于有些设备，你可能需要到制造商的网站上去下载驱动程序。许多设备可以使用 SDK 自带的 Android ADB 驱动程序，但这通常需把特定的设备硬件 ID 添加到 INF 文件中。在 Google 快速搜索设备的名称和“Windows ADB”通常可以返回连接那种设备所需要的信息。

对于 Linux 或 Mac OS X 系统，通常无须安装任何驱动程序，因为操作系统中已经提供了。不过在某些 Linux 版本中可能需要摆弄一下你的 USB 设备，通常是为 `udev` 创建一个新的规则文件。不同设备的做法也不同，上网搜索应该可以找到解决方法。

2.3.2 创建一个 Android 虚拟设备

Android SDK 带有一个可以运行 Android 虚拟设备(Android Virtual Devices, AVD)的模拟器。它由一个特定 Android 版本的系统图像、皮肤外观和各种属性组成。属性包括屏幕分辨率、SD 卡大小等。

为了创建 AVD，首先要启动 SDK 和 AVD 管理器。可以按前面的 SDK 安装步骤中介绍的方法来启动，也可以直接通过单击工具栏的 SDK 管理器按钮来启动。

(1) 选中左边列表中的 Virtual Devices，将出现目前可用的 AVD。除非已经使用过 SDK 管理器，否则列表应该为空。现在我们就修改它。

(2) 为了创建新 AVD，单击右边的 New... 按钮，弹出如图 2-7 所示的对话框。

(3) 可以为每一个模拟器指定一个名字，以便在以后引用它。同时指定一个 AVD 使用的 Android 系统版本。此外，还可以指定 AVD 的 SD 卡的大小和屏幕的大小等。而对于此处的 Hello World 项目来说，我们选择 Android 1.5 版本，其他选择默认即可。在实际的测试中，你可能需要创建多个 AVD 用以覆盖应用程序想要处理的多个 Android 系统版本和不同大小的屏幕。



图 2-7 用于创建 AVD 的对话框

注意：除非有多个 Android 系统版本和屏幕不同大小的设备，否则可以使用模拟器来测试不同的 Android 系统和不同的屏幕大小。

2.3.3 运行应用程序

现在设备和 AVD 已准备好，可以运行 Hello World 应用程序了。在 Eclipse 中，右击 Package Explorer 视图中的“hello world”项目，选择 Run As | Android Application(你也可以单击工具栏的 Run 按钮)。此时，Eclipse 会在后台为我们做下面的事情：

- (1) 如果自上次编译以来有任何修改，Eclipse 将项目编译成一个 APK 文件。
- (2) 如果还没有 Run 配置文件，为 Android 项目创建一个新的 Run 配置文件(稍后将介绍这个 Run 配置文件)
- (3) 通过启动或复用一個帶有合適版本的 Android 的已运行的模拟器实例或通过在已连接的设备上部署或运行一个应用程序来安装和运行应用程序(该设备的 Android 版本必须等同或高于创建 Android 项目时指定的最低 SDK 版本)。

如果已按前一节的说明创建了一个 Android 1.5 的 AVD，那么 Eclipse 的 ADT 插件就会启动一个运行该 AVD 的模拟器实例，并部署 Hello World APK 文件，最后运行该应用程序。运行输出如图 2-8 所示。



图 2-8 Hello World 应用程序

模拟器工作起来就跟真的设备一样，你可以通过鼠标跟它交互，就跟用手指操作手机一样。不过还是有一些差别如下：

- 模拟器只支持单点触摸，晃动鼠标就跟用手指触摸一样。
- 模拟器缺乏一些应用程序，例如 Android Market。
- 为改变设备的屏幕方向，你不需要转动监视器，只需要使用数字键盘的 7 键就可以，但这需要先按下 Num Lock 键来禁用它的数字功能。
- 模拟器运行非常慢，不要通过在模拟器上运行来评价你的应用程序的性能。
- 目前，模拟器只支持 OpenGL ES 1.0 和少数扩展功能，我们将在第 7 章中讨论 OpenGL ES。对于我们的目的，这就足够了，不过模拟器的 OpenGL ES 实现是存在 bug 的，经常会出现跟在真机上不同的结果。所以，请千万不要在模拟器上测试任何 OpenGL ES 的应用程序。

尽量多试用一下模拟器，熟悉它的功能。

注意：启动一个新的模拟器实例需要很长的时间(如果硬件配置不够高，可能需要多达 10 分钟)，所以在整个开发过程中尽量不要关闭模拟器，这样就不必反复重启。或者也可以在创建或编辑 AVD 时选中 Snapshot 选项，该选项用于保存和还原 VM 的一个快照，从而能够更快地启动。

有时在我们运行应用程序的时候，ADT 插件的自动选择模拟器/设备的功能也许是个障碍。例如，当我们想在一个指定的模拟器/设备上运行应用程序的时，可能存在多个连接的模拟器/设备。为了解决这个问题，我们可以关闭 Android 项目中 Run 配置的自动选择设备/模拟器的功能。那么什么是 Run 配置呢？

Run 配置提供了一种方法来告诉 Eclipse 应该如何去启动和运行你的应用程序。Run 配置允许你指定一些传入到应用程序的命令行参数和 VM 参数(该参数用于 Java SE 桌面系统应用程序中)等。Eclipse 和第三方插件为各种类型的项目提供不同的 Run 配置，ADT 插件为可用 Run 配

置集添加了一个 Android 应用程序 Run 配置。在本章前面第一次运行应用程序的时候，Eclipse 和 ADT 在后端已为我们创建了一个使用默认参数的 Android 应用 Run 配置。

如果想取得项目的 Run 配置，按如下操作即可：

- (1) 在 Package Explorer 视图中右击一个项目，并选择 Run As | Run Configurations。
- (2) 在左边的列表中选择“hello world”项目。
- (3) 在右边的对话框中，可以修改 Run 配置的名称，并更改其他选项卡(如 Android、Target 和 Commons)的设置。
- (4) 要从自动部署应用转成手动部署应用，请单击 Target 选项卡并选择 Manual。

再次运行应用程序的时候，需要选择一个兼容的模拟器或设备来运行应用程序，如图 2-9 的对话框所示。图中添加了一些不同目标的 AVD，并连接了两台设备。

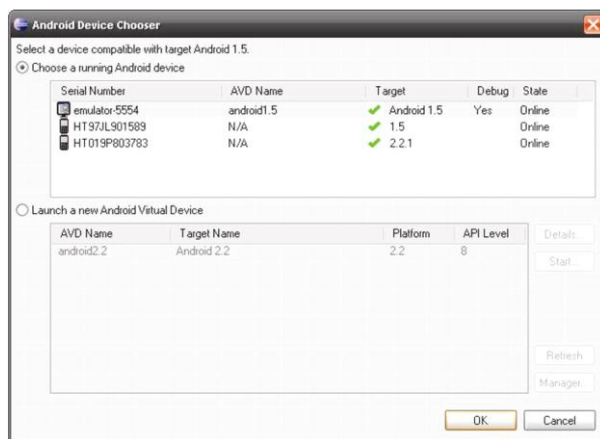


图 2-9 选择一个模拟器或设备来运行应用程序

该对话框展示了目前所有运行的模拟器和已经连接的设备，以及其他所有当前没有运行的 AVD。你可以选择任意一个模拟器或设备来运行你的应用程序。

2.3.4 调试应用程序

有时应用程序的行为可能会出现异常甚至崩溃。为了找出其中的原因，我们就需要来调试应用程序。

Eclipse 和 ADT 插件为 Android 应用程序提供了很强大的调试工具，我们可以在源代码中设置断点、查看变量或追踪堆栈等。

在调试应用程序之前，需要首先修改一下 AndroidManifest.xml 文件以进入调试模式。这就有点像鸡和蛋的问题，因为我们还没有仔细地看过 manifest 文件。不过，我们现在只需要知道 manifest 文件定义了很多应用程序的属性，其中一个就是该应用程序是否可以调试。该属性在 manifest 文件中的 <application> 标签中以 xml 属性的形式来设定。为了启用调试模式，我们把下面的代码加入 manifest 文件的 <application> 处。

```
android:debuggable="true"
```

开发应用程序时，该属性可以留着不变，但是当我们把应用程序部署上传到电子市场的时候要记得删除该属性。

现在，你的应用程序已经启动了调试模式，可以在模拟器或设备上进行调试。通常你是需要在要进行调试的代码段添加断点设置，以便能检查程序中特定位置处的程序状态。

要设置断点，只需要打开 Eclipse 中的源文件，并双击你想设置断点的行前面的灰色区域。我们演示一下，双击 HelloWorldActivity 文件的 23 行，每次单击按钮时调试器都会停下来。同时可以看到在源代码视图中，你双击的行位置出现了一个小圆点，如图 2-10 所示。如果你想删除该断点，只需要在源代码视图下在该位置重新双击即可。

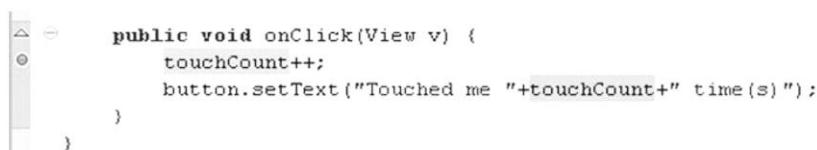


图 2-10 断点设置

启动调试跟前一节介绍的运行应用程序很相像。在 Package Explorer 视图中右击项目，并选择 Debug As | Android Application。这样系统就会为项目创建一个新的 Debug 配置，同运行应用程序的情况一样。同时，你还可以从快捷菜单中选择 Debug As | Debug Configuration，更改 Debug 配置的默认设置。

注意：除了在 Package Explorer 视图中使用项目的 Context 菜单以外，你可以通过 Run 菜单来运行和调试应用程序并访问配置。

首次启动调试进程时，Eclipse 会询问你是否切换到 Debug 透视图，当然我们是很乐意的。让我们来看看开始调试 Hello World 应用程序后进入该透视图的画面，如图 2-11 所示。

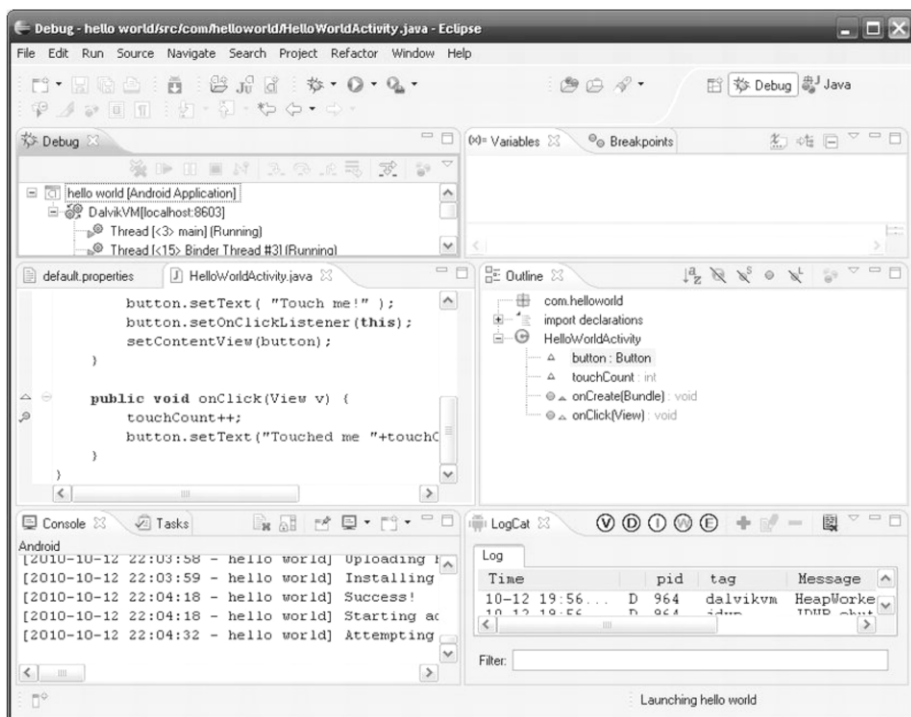


图 2-11 Debug 透视图

记得我们在介绍 Eclipse 时,说过它有很多种透视图的模式,而每种模式下都有很多不同任务的视图窗口。我们看到 Debug 透视图与 Java 透视图有很多的差别。

- 第一个注意到的新的视图是位于左上角的 Debug 视图。它显示了调试模式下所有当前运行着的应用和所有线程的堆栈跟踪。
- 源代码编辑视图位于 Debug 视图下面,跟在 Java 透视图模式下的一样。
- Console 视图通过 ADT 插件打印出各种信息,让我们知道发生了什么。
- LogCat 视图将成为我们开发中的有力工具。它会输出运行应用程序的模拟器/设备中的日志信息。这些信息来自系统组件、其他应用程序和我们自己的应用程序。该视图不仅能输出应用程序出错时的堆栈信息,还能输出在运行时自己的日志信息。第 2.3.5 节中我们将详细了解该视图。
- Outline 视图在 Debug 透视图中没有太大的作用。调试中更多的是关注一些断点和变量的信息和程序运行到哪里了。所以一般将它从 Debug 透视图中移除,为其他视图腾出空间。
- Variables 视图是一个对调试很重要的视图。当调试器遇到一个断点的时候,我们可查看或修改该段程序内的变量。
- Breakpoints 视图,展示了设置的所有断点的列表。

如果你感到好奇,可能已经单击了正在运行的应用程序的按钮以查看到调试器如何反应。它将会停止在 23 行,即我们设置断点的地方。同时也会看到 Variables 视图显示了作用于当前范围的各个变量,如活动本身(this)和方法的参数(v)等。也可以展开它们了解更多的信息。

Debug 视图向我们展现了当前堆栈中直到当前所在的方法的堆栈信息。注意,你可能有多个正在运行的线程,可以通过 Debug 视图随时暂停它们。

最后,注意断点处的那一行是高亮的,这表明你的程序刚好暂停运行到这个地方。

你可以按 F6 让调试器执行当前语句,按 F5 进入当前方法调用的方法内执行,或按 F8 继续正常运行程序。同样可以通过 Run 菜单来进行这些操作。除此之外还有很多其他的操作方法,我建议你多加尝试,看看哪种方法更适合你。

注意:好奇心是成功开发 Android 游戏的基础。所以,必须十分熟悉开发环境才能最大限度地利用它们。本书的篇幅决定了无法解释 Eclipse 的所有细节,所以建议你多进行尝试。

2.3.5 LogCat 和 DDMS

在 Eclipse 中安装 ADT 插件的时候,附带了很多新的视图和透视图。其中最常用的一个是 LogCat 视图——前面有过简单的介绍。

LogCat 是 Android 的事件日志系统,它允许系统组件和应用程序输出不同日志分级的日志信息。每一条日志由时间戳、日志分级、进程 ID、应用定义的标签和具体日志内容组成。

LogCat 视图通过一个连接的模拟器或设备来采集和显示这些日志信息,图 2-12 展示了一些该视图输出的示例。

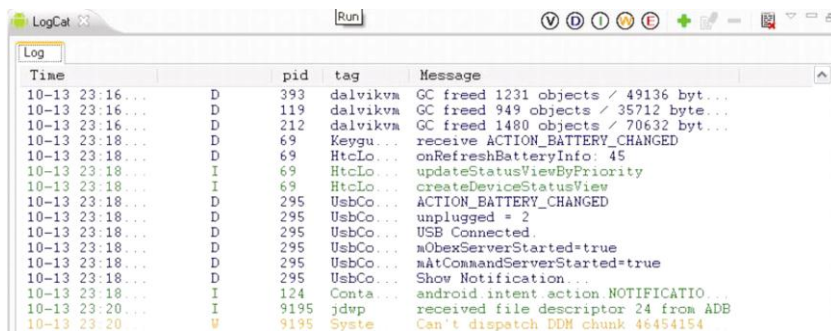


图 2-12 LogCat 视图

注意，在 LogCat 视图的右上角有一排按钮。

- 前 5 个按钮用于选择你想显示的日志分类。
- 绿色加号按钮用于定义一个基于标签的过滤器、进程 ID 和日志分级信息，在只想显示自己应用程序的输出时，日志分级很有用(可能对日志使用了特定的标签来进行记录)。
- 其他允许编辑过滤器、删除过滤器或清除当前输出内容的按钮。

如果当前连接了几个模拟器或设备，LogCat 视图只能输出其中一个的日志信息。如果你想获得更详细地控制和更多的查看选项，请切换到 DDMS 透视图模式。

DDMS(Dalvik Debugging Monitor Server)提供在所有连接的设备上运行的进程和 Dalvik VM 的大量深入信息。通过单击 Window | Open Perspective | Other | DDMS 可随时进入到 DDMS 透视图模式，如图 2-13 所示。

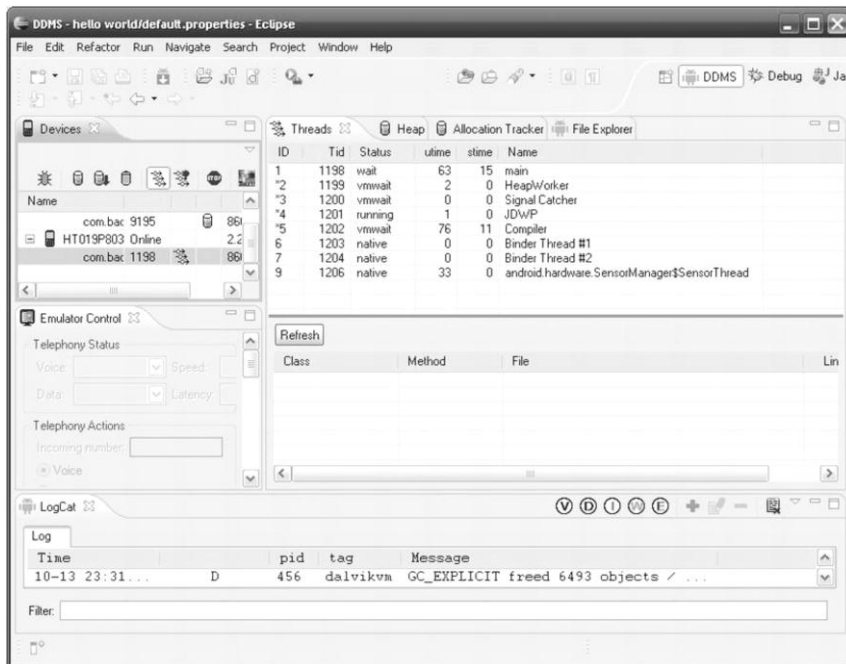


图 2-13 DDMS

通常情况下几个特定视图就能帮助我们完成示例应用程序。本例中，我们想得到所有进程的信息、它们的 VM 和线程、堆栈的当前状态和特定连接设备的 LogCat 信息等。

- **Devices** 视图显示了目前所有连接的模拟器和设备，还有在它们上面运行的所有进程。通过该视图的工具栏按钮可以进行各种操作，例如调试一个进程、记录堆栈和线程信息或进行屏幕截图等。
- **LogCat** 视图基本上和前面的透视图一样，不同之处在于它将在 **Devices** 视图中显示当前连接的设备的输出信息。
- **Emulator Control** 视图用于改变一个正在运行的模拟器实例的行为，例如你可以强行获取 GPS 坐标用于测试。
- **Threads** 视图显示了 **Devices** 视图下当前所选进程中正在运行的线程的信息。只有同时启用了线程跟踪，才会显示这些信息。通过单击 **Devices** 视图左边的第 5 个按钮可以启用线程跟踪。
- **Heap** 视图，图 2-13 中并没有显示出来，它可以显示一个设备上的堆栈信息。与线程信息一样，必须通过单击 **Devices** 视图左边第 2 个按钮来启用堆栈跟踪。
- **Allocation Tracker** 视图显示最近时刻哪些类实例得到资源分配，它提供一种很好地寻找内存泄漏的方法。
- **File Explorer** 视图允许你修改已连接模拟器或设备上的文件。就像在标准的操作系统资源管理器中一样，可以将文件拖放到该视图中。

DDMS 实际上是一个独立的工具，它通过 ADT 插件集成到 Eclipse 中。你可以在 \$ANDROID_HOME/tools 目录下将它作为一个独立应用程序来启动(在 Windows 下是 %ANDROID_HOME%/tools)。DDMS 不会直接连接到设备，而是通过 Android Debug Bridge(ADB)来连接，它是 SDK 中包含的另一个工具。下面来看一看 ADB 工具以丰富你对 Android 开发环境的一些知识。

2.3.6 使用 ADB

ADB 可用来管理连接的模拟器实例和设备，它由三个不同部分组成：

- 开发系统上运行的一个客户端，可以通过命令 `adb` 来启动命令行(如果按前面的说明设置环境变量时这将奏效)。当谈及 ADB 时，是指这个命令程序。
- 开发系统上运行的一个服务器，它是作为后端服务安装，负责 ADB 程序实例和连接的设备或模拟器实例之间的通信。
- 设备或模拟器上在后台运行的 ADB 守护进程，ADB 服务端通过连接它进行通信。

通常，我们通过 DDMS 透明地使用 ADB 并且忽略了它是一个命令行工具。但有些时候，使用它来操作一些任务却很方便，下面我们来快速浏览一下它的一些功能。

注意：通过 Android 开发人员网站 <http://developer.android.com> 查看 ADB 文档可看到可用命令的引用列表。

ADB 提供了一种很有用的任务来查询所有与 ADB 服务器(这里指开发机器)连接的模拟器或设备。为此，可以执行下面的命令行操作(注意 `>` 符号不是命令的一部分)。

```
>adb devices
```

屏幕上就会打印出所有的连接设备和模拟器信息以及各自的序列号，如下所示：


```
List of devices attached
HT97JL901589    device
HT019P803783    device
```

每台设备和模拟器的序列号是为标识在其上的特定序列命令。下面的命令行会将一个位于开发机器上的 APK 文件(myapp.apk)安装到一个序列号为 HT019P803783 的设备上。

```
> adb -s HT019P803783 install myapp.apk
```

-s 参数表示针对指定的设备执行该操作，可用于所有的 ADB 命令行操作。

我们可通过命令行操作将文件复制到已有的模拟器和设备上，同样也可将其中的文件复制出来。下面的命令就是将一个本地文件 myfile.txt 复制到一台序列号为 HT019P803783 的设备的 SD 卡里面。

```
> adb -s HT019P803783 push myfile.txt /sdcard/myfile.txt
```

要将文件 myfile.txt 从 SD 卡里面复制出来，可使用下面命令：

```
> abd pull /sdcard/myfile.txt myfile.txt
```

如果当前只有一台设备或模拟器连接到 ADB 服务器，那么我们可以忽略该序列号。因为 adb 工具会自动为你识别连接的设备或模拟器。

当然 ADB 工具还为我们提供很多功能。不过，大部分可通过 DDMS 的操作来替代这些命令行操作，所以我们通常会使用 DDMS。但对于一些要快速执行的任务，命令行工具更加理想。

2.4 小结

Android 的开发环境也许有时会有一点让人感到复杂。不过幸运的是，通过选择部分你所需要的功能子集来开始，然后再加上本章后面部分给出的这些信息就应该可以开始一些基本的编程了。

本章最重要的是教会你如何将各个部分整合起来。JDK 和 Android SDK 为所有的 Android 开发提供了基础。它们提供编译、部署和将应用程序运行到模拟器和设备上的工具。为了加速开发进程，我们在 Eclipse 中使用了 ADT 插件，它为我们提供了很多功能，否则我们就要使用 JDK 和 SDK 的命令行工具来完成。另外，Eclipse 本身也有几个概念要明确：工作区——用于管理项目，视图——提供特殊的功能，如源代码编辑和 LogCat 输出等；透视图——将多个视图功能集合起来，例如调试透视图；Run 和 Debug 配置——用于设置应用程序运行和调试的一些参数等。

掌握这些技术的最好办法就是不断尝试。本书还提供了一系列的示例项目，让你更快地熟悉 Android 开发，最后还是要靠你自己的努力才能够提高。

知道了这些信息后，就可以开始游戏开发了。