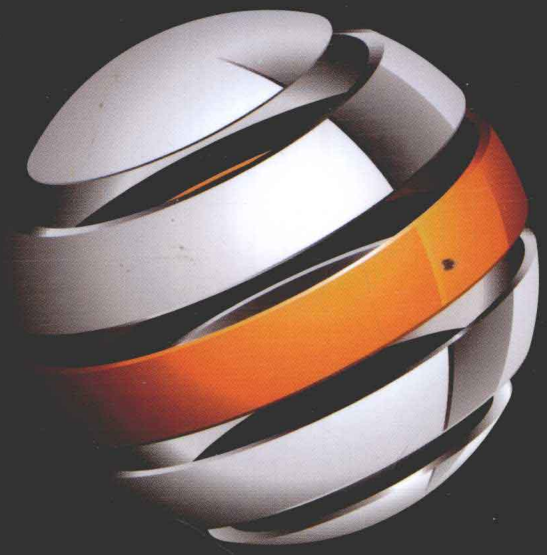
TURING 图灵程序设计丛书 移动开发系列

Apress®

- 将本地PC游戏轻松移植到Android的秘技
- 精彩炫酷游戏示例引人入胜
- 简明易读，一学就会



Pro Android Games

精通Android游戏开发

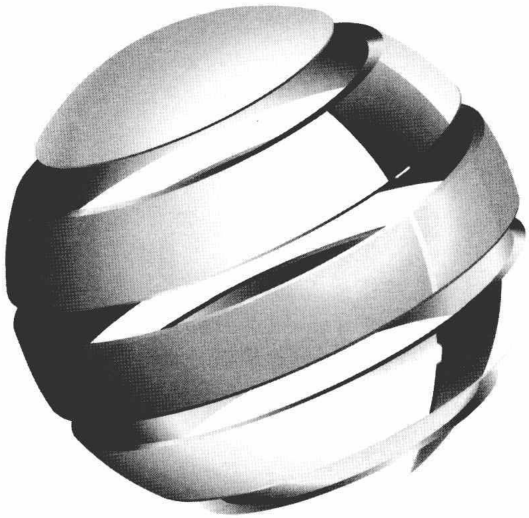
[美] Vladimir Silva 著

王恒 苏金国 等译

人民邮电出版社

POSTS & TELECOM PRESS

TURING 图灵程序设计丛书 移动开发系列



Pro Android Games

精通Android游戏开发

人民邮电出版社
北京

图书在版编目（CIP）数据

精通Android游戏开发 /（美）席尔瓦（Silva,V.）著；王恒等译. -- 北京：人民邮电出版社，2011.2
（图灵程序设计丛书）
书名原文：Pro Android Games
ISBN 978-7-115-24698-1

Ⅰ. ①精… Ⅱ. ①席… ②王… Ⅲ. ①移动通信－携带电话机－应用程序－程序设计②游戏－软件开发 Ⅳ. ①TN929.53②TP311.5

中国版本图书馆CIP数据核字(2010)第258313号

内 容 提 要

本书讨论如何将PC上的3D游戏移植到Android平台。作者从必备的技能和软件工具入手，逐步介绍如何从头构建纯Java游戏，如何混合使用OpenGL 3D图形和JNI，并以真实的PC游戏Wolfenstein 3D和Doom为例，介绍如何融合Java的优雅设计和C的强大功能，使混合游戏达到最佳性能。

本书适合熟悉Android平台的开发人员阅读。

图灵程序设计丛书

精通Android游戏开发

◆ 著 [美] Vladimir Silva
译 王 恒 苏金国 等
责任编辑 傅志红
执行编辑 谢灵芝

◆ 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
中国铁道出版社印刷厂印刷

◆ 开本：800×1000 1/16
印张：16.25
字数：384千字 2011年2月第1版
印数：1－3 500册 2011年2月北京第1次印刷

著作权合同登记号 图字：01-2010-1732号

ISBN 978-7-115-24698-1

定价：45.00元

读者服务热线：(010)51095186 印装质量热线：(010)67129223
反盗版热线：(010)67171154

前　言

本书将帮助你为 Android 平台创建最棒的游戏。现在讨论这个主题的书有很多，不过本书会从一个独特的视角，展示如何不费吹灰之力就将本地 PC 游戏轻松地移植到 Android 平台上。为此，书中采用了我们耳熟能详的真实例子，而且每一章都提供了大量源代码。要记住，深入学习本书之前，首先要有 Java 和 ANSI C 的坚实基础。我会尽我所能用清晰、简单的方式，结合图形和示例代码来解释这些最为复杂的概念。每一章提供的源代码都可以帮助你深入理解概念，作为移动游戏开发人员，你还可以充分利用这些源代码来节省开发时间。

需要什么软件

为了能够充分利用本书，需要以下工具。

已正确安装 Java SDK 的 Windows 或 Linux PC

我想这是显而易见的，因为大多数 Android 开发都用 Java 完成。要注意，我所说的是 Java SDK，而不是 JRE。由于后面各章会用到 JNI 头文件和命令行工具，所以 SDK 是必不可少的。

Eclipse IDE 和 Android SDK

Eclipse 是 Android 开发领域事实上的标准 IDE。我使用了 Eclipse Galileo 来创建本书所有示例的工作区，当然，使用 Eclipse Ganymede 也完全可以。

是否需要 IDE?

尽管我们选了 Eclipse Galileo 来创建代码工作区，不过你完全可以使用自己喜欢的其他 IDE。当然，这需要一些额外的设置。从 http://www.eclipse.org/可以得到 Eclipse Galileo。

对于如何用其他 IDE（如 IntelliJ 或某个基本编辑器）安装 Android SDK，有关说明请参见 http://developer.android.com/guide/developing/other-ide.html。

所谓正确安装 Android SDK，意味着两点。

- 必须为 Eclipse 安装 Android SDK 插件。

- 从 IDE 主菜单点击 Help （帮助）→Install New Software（安装新软件）。
- 点击 Add 按钮增加一个新站点，并输入名字 Android SDK 和位置 https://dl-ssl.google.com /android/eclipse/，然后点击 OK 按钮。
- 从 Available Software（可用软件）对话框选择 Android SDK，并按照向导的简易安装指令进行安装。

❑ 必须安装 Android SDK。可以从上面的 Android 站点下载。要记住，Eclipse 必须知道 Android SDK 的位置。从 IDE 主菜单点击 Window（窗口）→Preferences（首选项）。在左边的导航菜单选择 Android ，并输入 SDK 的位置（见图 0-1）。我使用的是 SDK 1.5，因为写本书时这是可用的最新版本。不过，本书的代码已经在 SDK 1.6 和 2.0 中做了测试（有关细节见“Android SDK 兼容性”一节）。

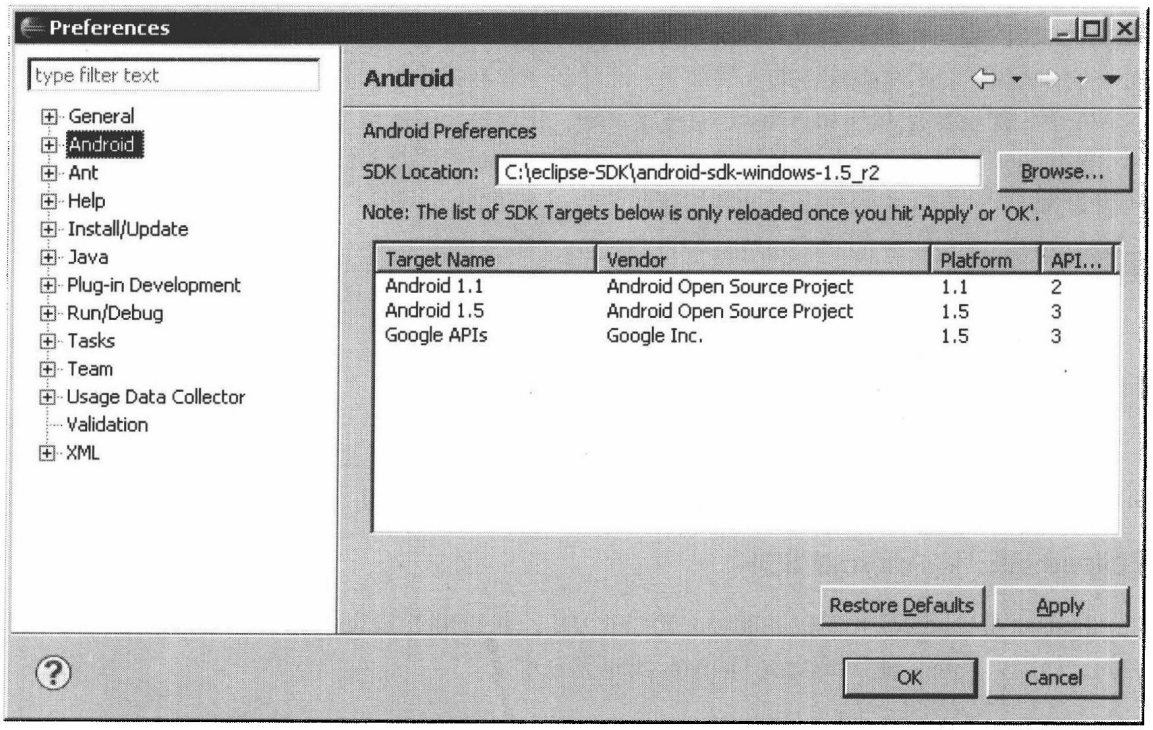


图 0-1 Eclipse Galileo 中的 Android SDK 配置对话框

面向 ARM 处理器的 GNU C 编译器

本书的混合型游戏要求必须安装 GNU C 编译器。这些游戏结合了 Java 代码和一个本地核心库，因此需要一个面向移动 ARM 处理器的 C 编译器。本书中使用的编译器是 Code Sourcery GNU G++ Toolkit（在第 1 章会详细介绍）。不过，并不是只能使用这个编译器。实际上，任何能够生成 ARM 代码的 C 编译器都可以使用，包括 Android NDK （Native Development Kit）。

各章源代码

这是可选的，不过学习本书时，各章源代码对于你理解书中的概念会很有帮助。我会尽可能简洁地介绍各章内容。不过，有些游戏（特别是 Wolf3D 和 Doom）采用了用 C 语言编写的庞大核心引擎（Doom 的引擎就有 10 万行代码），而且注释很少，因此很难读懂。总而言之，最重要的是你会看到如何不费劲地将这两种绝妙的语言（Java 和 C）轻松地结合在一起。登录出版商网站 http://www.apress.com 可以下载本书随附的源代码[①]，这些代码均使用 Eclipse Galileo 构建。

本书独特之处

我认为有一点非常重要，那就是要让读者理解本书的写作目标与众不同。尽管 Java 是 Android 的主要开发语言，但 Google 意识到，作为一个游戏平台，Android 要想取得成功，非常需要混合 Java/C 开发，正因如此，他们发布了 NDK。Google 逐渐认识到，为其他移动平台（如 iPhone）编写的本地游戏非常之多，Android 需要支持 C 开发才能迎头赶上。PC 游戏已经有几十年的发展历程（主要用 C 编写），只需使用一个简单的 ARM C 编译器，就可以把几千个 PC 游戏移植到 Android 平台。正是这一点让本书独具一格。既然可以采用一种优雅的方式简单地结合这两种语言，从而大量节省时间和金钱，为什么还要费力劳神地把 10 万行复杂的 C 代码转换成 Java 代码呢？利用这本书，你将掌握如何有效地结合这两种语言。这就是我的目标，也正是这一点，才让本书从众多相关图书中脱颖而出。另一方面，本书还包括介绍纯 Java 游戏的章节，通过合理均衡地分配各部分内容，希望能同时满足 Java 追随者和 C 爱好者的需要。

Android SDK 兼容性

作为一个开发人员，你可能想明确本书中代码的 SDK 兼容性。这是一个重要的问题，因为 Android SDK 的版本频繁更新。写本书时，Google 刚刚发布了 Android SDK 2.0 版本。书中的代码已经用以下版本的 Android SDK 做过测试：

- SDK 2.0
- SDK 1.6
- SDK 1.5
- SDK 1.0

总之，本书中的代码在 1.0 到 2.0 的所有 SDK 版本中都可以顺利运行，这也是我一直以来的目标。

SDK 1.6 版的变化

对于游戏开发来说，这个平台的 1.5 版到 1.6 版几乎未做任何改变。要了解 1.6 版本中 API 有哪些变化，详见 http://developer.android.com/sdk/RELEASENOTES.html。

① 本书代码也可以从图灵网站 http://www.turingbook.com 免费注册下载。——编者注

本书中对纯 Java 游戏和混合游戏的内容做了合理的划分，安排如下：

第 1 章

作为第一步，这一章首先建立一个 Linux 系统，完成混合游戏编译，包括获取 Android 源文件、提取设备系统库、建立定制编译 toolchain 和定制编译脚本。由于本书后面将要使用 Eclipse IDE，这一章还会介绍设置这个开发环境的有关细节。

第 2 章

这一章中，通过在一个本地库上构建简单的 Java 应用，你将了解如何采用一种优雅的方式结合 Java 和 C 代码。你将学习关于 Java 本地接口（Java Native Interface，JNI）的一些让人欢欣鼓舞的概念，还会了解结合 Java 和 C 时所用的 API，包括如何加载本地库、如何使用本地关键字、如何生成 JNI 头文件，以及方法签名、Java 数组与 C 数组、调用 Java 方法、产品编译和打包等内容。

第 3 章

从这一章开始，我们来研究纯 Java 游戏，首先从一个名为 Space Blaster 的真实游戏入手。这个游戏的目标是利用手指或键盘让一个飞船飞越星空。从这一章中，你会了解如何构建基于 XML 的定制线性布局、如何使用抽象类和定时器任务来模拟简单的游戏循环、如何在一个非 UI 线程中使视图无效，以及如何从项目资源加载 sprite 和声音，另外还会学习一些绘制技术，如 sprite 动画、简单对象绘制以及使用 Paint 对象设置样式和颜色。

第 4 章

第 4 章以经典街机游戏 Asteroids 为例，继续讨论纯 Java 游戏。你将学习在 Android 画布上绘制多边形 sprite 的所有知识。这个技术稍有些难度，因为 Android API 缺乏对多边形的支持。这一章依赖于 Java 语言的高度可移植性，将多边形代码从 J2SE API 移植到 Android API，从而最终创建 Asteroids。你可能已经注意到，这是一个主要基于多边形的游戏。此外，这一章还谈到另外一些有意思的主题，包括游戏生命周期中的步骤（即初始化、绘制和更新物理特性）、响应按键和触控事件，以及在设备模拟器上测试。

第 5 章

这一章讨论的是利用 OpenGL 绘制 3D 图形。这里会展示我无意中得到的一个绝妙技巧，利用这个技巧可以在 Java 和 C 中混合 OpenGL API 调用。Google 提供了一个 3D 立方体示例来展示如何在纯 Java 和混合模式中使用 OpenGL，我们就通过这个立方体示例说明这个概念。这个技巧将为 Android 的 3D 开发开拓一个新的领域，从而可以将大量 3D PC 游戏移植到这个平台，并且大大节省开发成本和时间。

第 6 章

第 6 章和第 7 章可以视为一个系列，这也是本书中我个人最喜欢的部分。这一章将把 Wolfenstein 3D 引入 Android 平台——这个游戏堪称所有 PC 上的 3D 射击游戏的“教父”。谁能想得到，这样一个复杂的 PC 游戏居然可以轻而易举地移植到 Android 移动设备上！这一章会介绍 Java 和 C 如何和谐共处，并介绍其他一些主题，如基本游戏体系结构（展示 Java 和 C 组件如何结合），声音、音乐、按键和触控事件的资源处理器，如何使用 JNI 将图形信息、视频缓冲区和声音/音乐请求级联返回到 Java，以及如何进行编译和测试。

第 7 章

作为下一步，第 7 章利用 PC 上的里程碑式游戏 Doom 进一步展示有关概念。Doom 是迄今为止所创建的最伟大的 3D 游戏，这一点绝对无可争议，它为 3D 图形游戏开辟了一个新领域。这一章的最终目标并不是描述这个游戏本身，而是希望你能从中了解到：将类似 Doom 这么复杂的 PC 游戏移植到 Android 平台是何等容易。不相信吗？可以给你提供一个证据：Doom 包含 10 万余行 C 代码，但是只需增加不到 200 行 JNI API 调用以及构建移动 UI 所需的 Java 代码，就可以把它引入到 Android！这一章将说明，完全不必把 10 万行 C 代码全部转换为 Java 代码，只需要采用巧妙的方式将这两种强大的语言完美地混合在一个“优雅”的应用中。想想看这将节省多少开发时间和成本！一定要好好读一读这一章。

目　录

第1章 欢迎进入Android游戏世界

欢迎进入 Android 游戏世界！本书的目标就是帮助你构建最出色的 Android 游戏。通过各章的学习，你将了解如何创建两类游戏：一类是纯 Java 游戏，另一类是可能最有意思的混合游戏，其中会结合 Java 的优雅设计与 C 的强大功能来达到最佳性能。正是因为 Java 和 C 的巧妙结合，使得本书中介绍的游戏独一无二，因为 Google 本身并不支持这种开发。不过，你可能会认为："何必要那么麻烦地开发混合游戏呢？"毕竟，Java 已经提供了构建各种游戏所需要的全部 API。这一点当然不假。不过，还有成千上万用 C 编写的游戏可以转向 Android，只需编译 C 内核，并使用 Java 本地接口（Java Native Interface，JNI）包装一个 Java。GUI（Graphical User Interface，图形化用户界面）就能轻松做到。在本书中，你将学习如何将两个原本面向 PC 的一流 3D 射击游戏 Wolfenstein 3D 和 Doom 引入这个平台。

总而言之，我的目标就是提供构建 Android 游戏的最新秘诀，这不仅包括已经公布的技巧，还包括一些尚未公开的独家秘技。另外，如果你计划将一个 PC 游戏移植到 Android 平台，本书也非常有帮助，能提供一些极有价值的经验。不过，在正式学习之前，要想最充分地利用本书，还需要了解一些内容。

1.1 必备技能

本书面向的读者应当是有丰富经验的游戏开发人员，不仅要精通 Java，还要擅长 C。这是因为性能对于游戏开发至关重要。尽管 Java 可以提供优雅的面向对象功能，但只有 C 能够保证提升游戏开发所需的性能。结合 Java 和 C 就能将它们的精华集于一身，同时展现二者最出彩的一面。本书假设你已经熟悉 Android，而且通晓 Linux 和 shell 脚本。

1.1.1 扎实的 Android 基础

阅读本书的前提是你已经非常了解 Android 开发的基础知识，例如，要知道什么是活动、视图和布局。请看以下代码段，如果你能看清楚这个代码要做些什么，就说明你已经具备了必要的基础。

```
public class MainActivity extends Activity
{
```

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

这个代码段定义了控制应用生命周期的主要活动或类。onCreate 方法会在应用启动时调用一次，它的任务是为应用设置内容布局或 GUI。

另外对于如何使用 XML 创建 GUI 也要有基本的了解。请看下面的代码段，你能说出它的作用吗？

```
<?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">

<ImageView android:id="@+id/doom_iv"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/doom"
    android:focusableInTouchMode="true" android:focusable="true"/>

 <ImageButton android:id="@+id/btn_upleft"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:src="@drawable/img1" />
</RelativeLayout>
```

这段代码定义了一个相对布局。在相对布局中，部件（widget）会相对于彼此放置（有时会重叠）。在这里，将有一个图像视图填充整个屏幕。这个视图会显示名为 doom.png 的文件作为背景，这个文件存储在项目的 res/drawable 文件夹中。另外这个图像视图还会接收按键和触控事件。在屏幕的左下方会显示一个 ID 为 btn_upleft 的图像按钮，它重叠在图像视图上。

需要一个 Android 教程吗？

与 Android 开发有关的概念实在太多，我们不太可能记住有关活动、视图和布局的所有细节。要想快速得到这些信息，可以求助于一个便捷资源，这就是 Android 教程，下面是网址：

http://developer.android.com/

还有为 Android 开发人员提供的“终极”指南，其中包括最新发布版本、下载包、SDK Quick Start、版本说明、本地开发工具和早期版本，这个指南可以在以下地址找到：

http://developer.android.com/sdk/1.6_r1/index.html

在本书中（特别是讨论本地代码的章节），我大量使用了 Android SDK 命令工具来完成系统

管理任务。因此，你还要清楚地了解这些工具，特别是 Android 调试桥（Android Debug Bridge，ADB）。应当知道如何完成以下工作。

- 创建一个 Android 虚拟设备（Android Virtual Device，AVD），它封装了一个针对设备配置的相应设置，如固件版本和 SD 卡路径，创建起来实际上很简单，可以从 IDE 使用 AVD Manager（可以点击工具条中的黑色电话图标来访问）完成。
- 创建一个 SD 卡文件。后面几章中的一些游戏包含很大的文件（5MB 以上）。为了节省空间，代码会把所有游戏文件保存在设备 SD 卡中，你要知道如何创建 SD 卡文件。例如，要在主目录中创建一个 100MB 的 SD 卡文件（名为 sdcard.iso），可以使用以下命令：

```
$ mksdcard 100M $HOME/sdcard.iso
```

- 连接到模拟器。做到了这一点才能完成各种各样的系统管理，如库提取（library extraction）。要打开设备的一个 shell，可以使用以下命令：

```
$ adb shell
```

- 向模拟器上传文件和从模拟器获取文件。这些任务对于在设备中存储游戏文件和从设备提取游戏文件很有帮助。可以使用以下命令：

```
$ adb push <LOCAL_FILE> <DEVICE_FILE>
$ adb pull <DEVICE_FILE> <LOCAL_FILE>
```

说明 运行有关命令来创建SD卡文件、连接到模拟器或者上传和提取文件之前，一定要确保将SDK_HOME/tools目录增加到系统变量PATH中。

1.1.2 Linux 和 Shell 脚本的基本知识

讨论混合游戏的几章中将会使用 Ubuntu Linux，所以你要重拾起那些陈旧的 Unix 技巧。

你应当知道基本的 shell 命令，如罗列文件、安装软件组件（可能很麻烦，这取决于你的 Linux 发布版本），以及基本的系统管理命令。

本书里有一些非常简单的 shell 脚本。对 bash shell 有基本的了解就很有帮助。

提示 如果需要复习Linux和shell脚本的知识，可以查阅Ashley J.S Mills编写的教程：http://supportweb.cs.bham.ac.uk/documentation/tutorials/docsystem/build/tutorials/unixscripting/unixscripting.html。

1.2 需要哪些软件工具

本章首先解释如何建立适当的环境来编译混合（C 和 Java）游戏，这包括 IDE（Eclipse）和

Android SDK，这是构建所有基本 Android 应用的必备工具。如果想将 Java 优雅的面向对象特性与 C 的强大功能相结合来得到最佳性能，这些环境信息将非常重要，这也是后面几章构建 Doom 和 Wolfenstein 3D 的必要条件。

假设你的桌面计算机上已经安装有以下软件。

- VMware Player 或 Workstation：这是运行 Linux 虚拟机（VM）的必要软件。VMware 可以从 VMware 下载站点（http://www.vmware.com/products/player/）免费获取。
- Ubuntu Linux VMware 工具：这是真正的 Linux 操作系统，所有开发都将在这里完成。如果还没有安装，可以从 VMware Virtual Appliance Marketplace（http://www.vmware.com/appliances/）免费下载。注意这个工具可能很大（600 MB 以上）。
- Eclipse：这是用来创建项目的 IDE。版本 3.3（Europa）、3.4（Ganymende）或 3.5（Galileo）都可以使用。
- 已正确配置的 Android SDK。写这本书时，SDK 的最新版本是 1.6。我们将使用 Android 调试桥来连接设备。首先把 SDK 解压缩到指定的文件夹，确保将命令行工具添加到系统的 PATH。可以编辑主目录中的文件.bashrc，修改 PATH 环境变量：PATH=[PATH_TO_SDK]/tools:$PATH。现在应该能够从命令行启动和连接模拟器了。可以做一个简单的测试，打开一个终端窗口，并键入 adb。你会看到屏幕上显示出这个工具的帮助文本。
- Java JDK 5.0 或更高版本：这是运行 Eclipse 和 Android SDK 的必要条件。

首先为 Ubuntu VM 安装必要的软件。登录后就可以开始建立环境了。

1.3　建立环境

要建立构建 Android 游戏的环境，需要在 Linux 桌面计算机上安装 3 个软件组件。

- Android 源文件：包含全部 Android 源代码，以及用来构建定制共享库的 C/C++/JNI 头文件。
- Android 本地库：包括 C 运行库、Math、XML、声音和其他库。
- 面向 ARM 处理器的 GNU C/C++工具链：这个工具链提供了构建本地库所需的一个 C/C++ 编译器和链接器，还包括其他一些有用的工具，如调试工具和性能分析工具，帮助完成调试过程。

还要写两个定制 shell 脚本，在编译混合游戏时可以免去很多麻烦。

1.3.1　获得 Android 源文件

为了存储 Android 源文件，Google 使用了一个软件版本控制工具，名为 Gittool（可以从 http://git-scm.com/得到）。根据你的 Ubuntu 版本，可能需要安装一些必要的包才能使用这个工具。为了确保已经安装所有必要的包，可以打开一个控制台，键入以下命令（整个命令都在同一行上）：

```
$ sudo apt-get install git-core gnupg
  sun-java5-jdk flex bison gperf
  libsdl-dev libesd0-dev libwxgtk2.6-dev
  build-essential zip curl libncurses5-dev zlib1g-dev
```

这个命令会安装运行 Git 所需的包。执行这个命令需要有系统管理员（sysadmin）访问权限。

提示 关于获取Android源文件的更多细节，请参考Android Open Source Project（Android开源项目，http://source.android.com/download）。

为了便于使用 Git，还需要安装和配置一个名为 repo 的工具（由 Google 提供）。在主目录创建一个名为 bin 的文件夹，并下载 repo：

```
$ cd ~
$ mkdir bin
$ curl http://android.git.kernel.org/repo >~/bin/repo
$ chmod a+x ~/bin/repo
```

用以下命令将这个 bin 文件夹添加到搜索路径：

```
$ export PATH=$HOME/bin:$PATH
```

如果希望每次登录系统时都能使用这个工具，可以把这个命令添加到$HOME/.bashrc 文件中。

接下来，在主目录中创建一个名为 mydroid 的文件夹来存储 Android 源文件，然后切换到这个文件夹：

```
$ mkdir mydroid
$ cd mydroid
```

最后，下载代码：

```
$ repo init -u git://android.git.kernel.org/platform/manifest.git
$ repo sync
```

提示 repo init命令会下载主分支。要下载其他分支，可以使用`repo init -u git://android.git.kernel.org/platform/manifest.git -b [BRANCH_NAME]`。

现在可以坐下来喝杯咖啡，静静地等待。下载可能最多需要 1 个小时，具体取决于你的网络速度。

一旦 Android 源文件下载完成，文件夹树应该如图 1-1 所示。其中最重要的文件夹名为 bionic。bionic 是支持 ARM 和 x86 指令集的 C 库，专门在 Android 设备上运行。bionic 一半是 BSD 一半是 Linux，它的源代码混合了 BSD C 库和用于处理线程、进程、信号等的 Linux 指定位。构建共享库所用的大多数 C 头文件都将由这个文件夹提供。

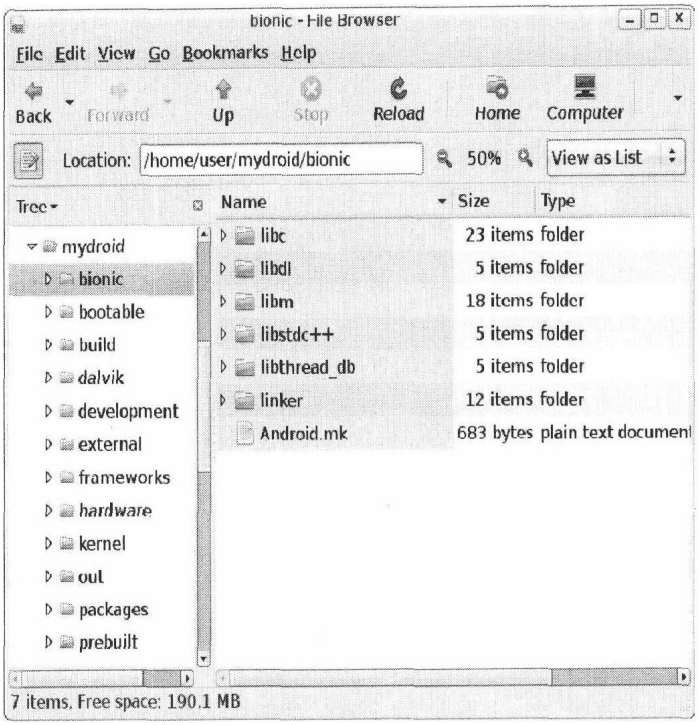


图 1-1　Android 源文件树

1.3.2　提取本地 Android 库

上一节中的头文件库提供了编译代码所需的必要文件。不过，除此以外，还需要系统映像共享库来完成链接过程。也就是说，需要把*.so 文件存储在设备的/system/lib 中。例如，通过使用模拟器 shell，可以查看设备文件系统和一些系统库。使用 Android 调试桥连接到设备，然后运行 df 命令来检查设备文件系统，如下所示：

```
user@ubuntu:~$ adb shell
# df
```

```
/dev: 47284K total, 0K used, 47284K available (block size 4096)
/sqlite_stmt_journals: 4096K total, 0K used, 4096K available (block size 4096)
/system: 65536K total, 43496K used, 22040K available (block size 4096)
/data: 65536K total, 43004K used, 22532K available (block size 4096)
/cache: 65536K total, 1156K used, 64380K available (block size 4096)
/sdcard: 40309K total, 34114K used, 6195K available (block size 512)
```

```
# ls -l /system/lib
```

```
-rw-r--r-- root      root          9076 2008-11-20 00:10 libdl.so
```

```
-rw-r--r-- root     root       227480 2008-11-20 00:10 libc.so
-rw-r--r-- root     root        13368 2008-11-20 00:10 libthread_db.so
-rw-r--r-- root     root         9220 2008-11-20 00:10 libstdc++.so
-rw-r--r-- root     root       140244 2008-11-20 00:10 libm.so
-rw-r--r-- root     root        79192 2008-11-20 00:10 libz.so
-rw-r--r-- root     root        92572 2008-11-20 00:10 libexpat.so
-rw-r--r-- root     root       767020 2008-11-20 00:10 libcrypto.so
-rw-r--r-- root     root       155760 2008-11-20 00:10 libssl.so
[Other files...]
```

df 命令会显示设备文件系统的有关信息。从这里可以查看存储在设备 /system/lib 文件夹中的库。ls 命令显示了最重要的库：C 运行库（libc.so）、Math 运行库（libm.so）、Gzip（libz.so）、XML（libexpat.so）等库。这些就是为完成链接而需要提取到本地系统的文件。为了提取这些文件，可以创建一个简单的脚本，使用模拟器工具的 adb pull 命令将文件从设备取至本地文件系统。

首先，在主目录中创建一个文件夹来存储这些库：

```
$ mkdir -p $HOME/tmp/android/system/lib
$ cd $HOME/tmp/android/system/lib
```

接下来，创建一个简单的脚本，从设备将文件获取到$HOME/tmp/android/system/lib 文件夹中。代码清单 1-1 中的 bash 脚本循环处理这些库名，并从设备的/system/lib 文件夹将文件取至本地文件系统的当前目录。

代码清单 1-1 将设备库从/system/lib 获取到本地文件系统的脚本

```
#!/bin/bash

# Emulator 1.5+的设备库
# 这些库位于/system/lib
libs="browsertestplugin.so libEGL.so libFFTEm.so
 libGLESv1_CM.so libaes.so libagl.so libandroid_runtime.so
 libandroid_servers.so libaudioflinger.so libc.so libc_debug.so
 libcameraservice.so libcorecg.so libcrypto.so
 libctest.so libcutils.so libdl.so libdrm1.so
 libdrm1_jni.so libdvm.so libemoji.so
 libexif.so libexpat.so libhardware.so libhardware_legacy.so
 libicudata.so libicui18n.so libicuuc.so libjni_latinime.so
 libjni_pinyinime.so liblog.so libm.so libmedia.so libmedia_jni.so
 libmediaplayerservice.so libnativehelper.so libnetutils.so
 libopencoreauthor.so libopencorecommon.so libopencoredownload.so
 libopencoredownloadreg.so libopencoremp4.so libopencoremp4reg.so
 libopencorenet_support.so libopencoreplayer.so libopencorertsp.so
 libopencorertspreg.so libpagemap.so libpixelflinger.so
 libpvasf.so libpvasfreg.so libreference-ril.so libril.so libsgl.so
 libskiagl.so libsonivox.so libsoundpool.so libsqlite.so
 libsrec_jni.so libssl.so libstdc++.so libsurfaceflinger.so
 libsystem_server.so libthread_db.so libui.so
 libutils.so libvorbisidec.so libwbxml.so libwbxml_jni.so
 libwebcore.so libwpa_client.so libxml2wbxml.so libz.so"
```