Android平台 手机软件开发系列丛书

Android移动开发案例详解

张利国 代 闻 龚海平 编著

人民邮电出版社
POSTS & TELECOM PRESS

Android平台 手机软件开发系列丛书

Android移动开发案例详解

张利国 代 闻 龚海平 编著

人民邮电出版社
北京

图书在版编目（CIP）数据

Android移动开发案例详解 / 张利国，代闻，龚海平编著. -- 北京 : 人民邮电出版社，2010.2
（Android平台手机软件开发系列丛书）
ISBN 978-7-115-22005-9

Ⅰ. ①A… Ⅱ. ①张… ②代… ③龚… Ⅲ. ①移动通信－携带电话机－应用程序－程序设计 Ⅳ. ①TN929.53

中国版本图书馆CIP数据核字(2009)第235276号

内 容 提 要

本书作为《Android 平台手机软件开发》系列丛书的第二本分册，将通过 10 多个实际的开发案例对 Android 平台展开详细的实战介绍，内容涉及应用程序（如图片浏览器、文件浏览器、通讯录、任务管理器等），实用软件（如音乐播放器、天气预报、多媒体播放器、短信语音播报、手机远程监控器等），游戏软件（如 JET BOY、“连连看”、“贪吃蛇”游戏等）以及其他程序的详细开发过程。

本书适合对 Android 手机平台开发具有一定基础的人员参考使用，也可用作培训教材以及大专院校 Android 课程的参考书，并适合读者自学。

Android 平台手机软件开发系列丛书

Android 移动开发案例详解

◆ 编　　著　张利国　代　闻　龚海平
责任编辑　王建军
执行编辑　赵　斌

◆ 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
北京昌平百善印刷厂印刷

◆ 开本：787×1092　1/16
印张：16.5　　2010 年 2 月第 1 版
字数：423 千字　　2010 年 2 月北京第 1 次印刷

ISBN 978-7-115-22005-9

定价：42.00 元

读者服务热线：(010)67119329　印装质量热线：(010)67129223
反盗版热线：(010)67171154

序

Google Android SDK 从 2007 年年底发布以来，受到了广大移动开发爱好者的青睐。近年来，手机平台经过 Palm OS、Windows Mobile、Symbian、iPhone 等的发展，终于迎来了真正开放的平台——Google Android Platform。2008 年 10 月 21 日，Google 宣布其手机操作系统 Android 的源代码完全开放，任何人和机构都可以免费使用。之前由于其他手机操作系统众多以及手机软件开发的复杂性，而且手机软件也多是专用软件，要求比较高，使不少开发者望而却步。自从 Android 操作系统问世以来，手机软件的开发脱去了神秘的面纱，使得开发者可以和 Android 进行面对面交流，尤其是 Android Market，各种各样的 Android 软件展示在用户面前，使得学习和使用 Android 的余地也更大。

Android 研发已经逐渐成为了一个热门方向，而其他操作系统的吸引力正在下降。2008 年 11 月初，摩托罗拉宣布放弃自己的 Linux 系统，转向 Android 平台。Android 的崛起，受影响最大的就是诺基亚。在苹果的 iPhone 和 Google 的 Android 尚未推出之前，诺基亚控制了最好的终端，也拥有最成功的智能手机操作系统，还控制了通往移动互联网的入口，这使人们觉得诺基亚向互联网转型是顺理成章的。但是 iPhone 的推出改变了人们对手机的认知，而 Android 的推出则影响了整个移动操作系统行业的开发规则，诺基亚及其所控制的手机操作系统 Symbian 也经受了极大的挑战。Android 的优势在于：第一，Android 应用开发比较容易，因为它是基于 Java 的，目前 Java 开发人员众多；第二，它的底层是基于 Linux 的，而 Linux 在我国也有广泛的基础，很容易移植到各种嵌入式平台中。

本套丛书是在 xmobileapp 工作室开发人员的精心打造下推出的 Android 系列软件开发图书。xmobileapp 团队是国内较早的对智能手机 Android、iPhone、Symbian 等系统进行研究和开发的专业团队，其推出的多款软件已成功地在 App Store 和 Android Market 市场上投放，具体可通过 xmobileapp 团队的网站了解：www.xmobileapp.com.cn。

本套丛书以 Google 强力推出的 Android 平台及 Eclipse 开发环境为基础进行介绍。Eclipse 是 Java 开发者不可或缺的优秀开发环境，因此，Android 平台以其天然的开放性使得开发工程师很容易上手。

前　言

近几年来，手机平台经过 Palm OS、Windows Mobile、Symbian、iPhone 等的发展，终于迎来了真正开放的手机平台——Google Android Platform。2008 年 10 月 21 日，Google 宣布其手机操作系统 Android 的源代码完全开放，任何人和机构都可以免费使用。之前，由于其他手机操作系统众多以及手机软件开发的复杂性，不少开发者望而却步。而且手机软件也多是专用软件，要求比较高。但是，自从 Android 操作系统问世以来，手机软件的开发脱去了神秘的面纱，使得开发者可以和 Android 进行面对面交流，尤其是 Android Market，使得各种各样的 Android 软件展示在用户面前，学习和使用余地也更大。

Android 作为 Google 公司企业发展战略的重要组成部分，将进一步推进“随时随地为每个人提供信息”这一企业目标的实现。作为一款多方倾力打造的平台，Android 具有许多优点：实际应用程序运行速度快；开发限制少，平台开放；程序多任务性能优秀，切换迅速等。当然，它也具有系统细节不完善、电源管理不好、软件的界面不太好、支持的软件厂商还比较少等缺点。但是凭借 Google 公司的强大实力以及与开放手机联盟的通力合作，我们相信 Android 会越来越好，一定会成为主流的手机操作系统平台。本书将通过十多个实际的开发案例对 Android 平台展开详细的实战介绍。

本书主要内容

第 1 章主要介绍 Android 平台发展史、平台架构及开发环境等。

第 2 章主要介绍图片浏览器案例的详细开发过程。

第 3 章主要介绍文件浏览器案例的详细开发过程。

第 4 章主要介绍通讯录案例的详细开发过程。

第 5 章主要介绍任务管理器案例的详细开发过程。

第 6 章主要介绍音乐播放器界面案例的详细开发过程。

第 7 章主要介绍天气预报案例的详细开发过程。

第 8 章主要介绍多媒体播放器案例的详细开发过程。

第 9 章主要介绍短信语音播报案例的详细开发过程。

第 10 章主要介绍手机远程监控器案例的详细开发过程。

第 11 章主要介绍 JET BOY 游戏案例的详细开发过程。

第 12 章主要介绍“连连看”游戏案例的详细开发过程。

第 13 章主要介绍“贪吃蛇”游戏案例的详细开发过程。

第 14 章主要介绍 GPS 和 Google Map API 的使用。

第 15 章主要介绍 AppWidgets 原理和应用。

本书由张利国负责策划和统稿，感谢 xmobileapp 团队的同事罗峰、代闻、龚海平、王植萌、赵栓、徐学东等积极参与本书的编写和修改工作。本文部分案例参考于网上的源代码，代码改动遵循 Apache Licence 和 GPL Licence，适于商业发布，作者如发现被侵权，请及时联系。

由于书稿的内容较多，参考资料有限，虽经再三审查但是编写过程中还可能会有些错误，

恳请广大读者、老师批评指正。

学习建议

本书的源代码可以在人民邮电出版社（http://www.cww-netbook.cn/）的“资源下载”中找到。为了提高学习效果，希望读者能够在理解的基础上修改源代码，完善一些功能，加深印象；同时希望读者尽可能多参阅 Android SDK 文档以及官方论坛。在阅读本书的过程中如果遇到问题，请发送邮件到 liguo.zhang@xmobileapp.com，我们将会及时回复你。

编　者

2009 年冬于北京

目　录

第 1 章
Android 移动平台发展概述

本章知识点：

（1）Google Android 手机的由来和未来发展前景。

（2）Android 平台的架构（应用层、FramWork 层、库、内核、硬件设备）。

（3）Android 开发环境的搭建（Eclipse + ADT + Android SDK）。

1.1 Android 诞生背景

2007 年 11 月 5 日，Google 发布了基于 Linux 平台的开源移动平台——Android。该平台由操作系统、中间件、用户界面和应用软件等组成，号称是首个为移动终端打造的真正开放的移动开发平台。

2008 年 9 月 22 日，美国运营商 T-Mobile USA 在纽约正式发布第一款 Google 手机——T-Mobile G1。该款手机为台湾宏达电（HTC）代工制造，是世界上第一部使用 Android 操作系统的手机，支持 WCDMA/HSPA 网络，理论下载速率为 7.2Mbit/s，并支持 Wi-Fi 无线局域网络。

Google 与开放手机联盟（Open Handset Alliance）合作开发了 Android 移动开发平台。开放手机联盟由摩托罗拉、高通、宏达电、T-Mobile 和中国移动等在内的 30 多家移动通信领域的领军企业组成。Google 通过与运营商、设备制造商、开发商和其他第三方结成深层次的合作伙伴关系，建立标准化、开放式的移动电话软件平台，从而希望在移动产业内形成一个开放式的生态系统。

Android 作为谷歌企业战略的重要组成部分，将进一步推进“随时随地为每个人提供信息”这一企业目标的实现。Google 的目标是让移动通信不依赖于设备甚至平台。出于这个目的，Android 将补充 Google 长期以来奉行的移动发展战略：通过与全球的手机制造商和移动运营商结成合作伙伴，开发并推广移动服务。

1.2 Android 开发框架

1.2.1 Android 平台架构

如图 1-1 所示，Android 平台下层结构的核心为嵌入式 Linux 2.6 操作系统，中间是

Google 为 Android 开发的 Libraries(函数库) 及 Android Runtime(核心库、Dalvik 虚拟机), 上层为 Application Framework(应用程序框架)。用此平台可以开发各种不同的手机应用程序。

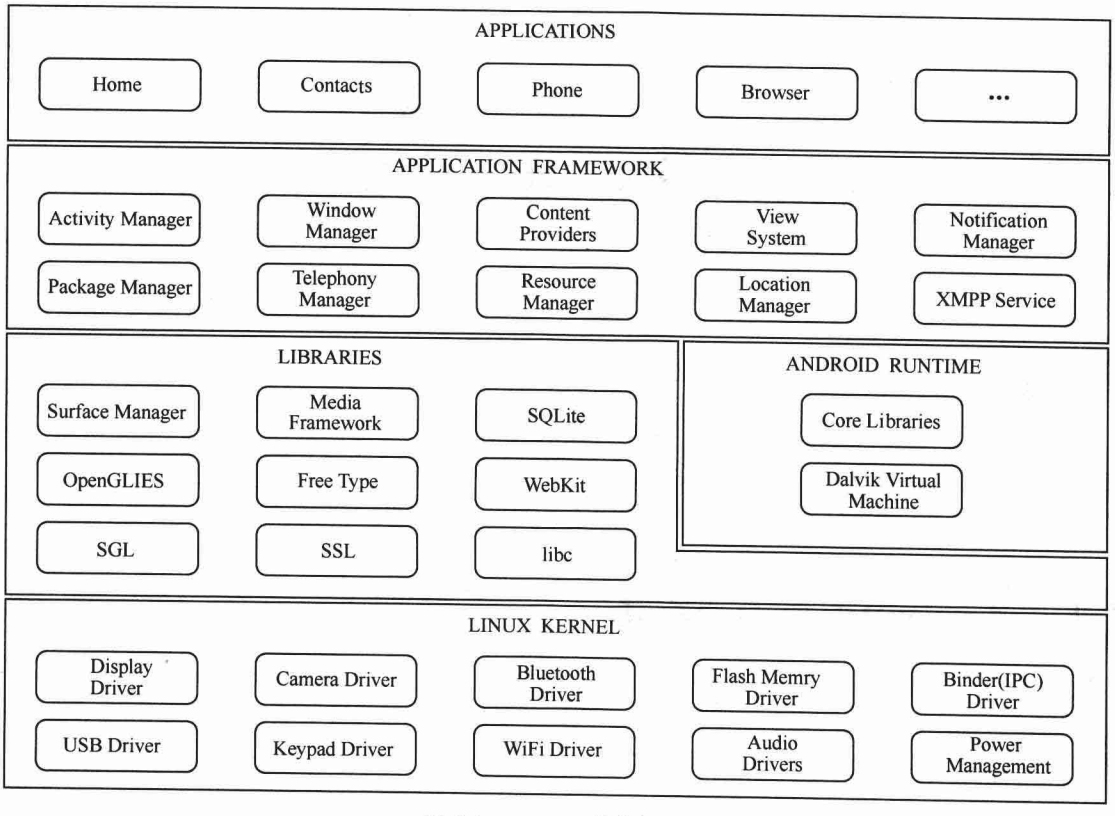

图 1-1　Android 平台架构

(1) Android 应用程序层

Android 会同一系列核心应用程序包一起发布。该应用程序包包括 email 客户端、SMS 短消息程序、日历、地图、浏览器、联系人管理程序等。所有的应用程序都是使用 Java 语言编写的，如图 1-1 “APPLICATIONS” 一栏所示。

(2) Android 应用框架

开发人员也可以完全访问 Android 核心应用程序所使用的 API 框架，如图 1-1 “APPLICATIONS FRAMEWORK” 一栏所示。该应用程序的架构设计简化了组件的重用；任何一个应用程序都可以发布它的功能块并且任何其他的应用程序都可以使用其所发布的功能块 (不过要遵循框架的安全性限制)。同样，该应用程序重用机制也使用户可以方便地替换程序组件。

隐藏在每个应用后面的是一系列的服务和系统，其中包括：

丰富而又可扩展的视图 (Views)，可以用来构建应用程序，它包括列表 (Lists)、网格 (Grids)、文本框 (Text Boxes)、按钮 (Buttons) 等。

内容提供器 (Content Providers)，使得应用程序可以访问另一个应用程序的数据 (如联系人数据库)，或者共享它们自己的数据。

资源管理器 (Resource Manager)，提供非代码资源的访问，如本地字符串、图形和布局文件 (layout files)。

通知管理器（Notification Manager），使得应用程序可以在状态栏中显示自定义的提示信息。

活动管理器（Activity Manager），用来管理应用程序生命周期并提供常用的导航回退功能。

（3）Android 运行库

如图 1-1 “ANDROID RUNTIME” 一栏所示，Android 包括了一个核心库（Core Libraries），该核心库提供了 Java 编程语言核心库的大多数功能。

Dalvik 虚拟机（DVM，Dalvik Virtual Machine）是一种寄存器型态的虚拟机。Google 在 DVM 开发时就已经设想用最少的内存来执行，以及同时可执行多个 VM 为前提。

不过，上述特性需要 Linux 操作系统的协助才能实现，例如程序执行的控制、多线程的支持、内存管理等。事实上不仅 Dalvik 虚拟机如此，Java 虚拟机也一样有操作系统依赖性，不同的操作系统需要不同的 Java 虚拟机，而虚拟机会针对操作系统再进行各项调整，以便能最佳化执行。

Dalvik 虚拟机是 Google 用于移动设备 Android 平台的一个重要组成部分。虚拟机可运行 Java 平台应用程序，这些应用程序被转换成紧凑的 Dalvik 可执行格式（.dex），该格式适合内存和处理器速度受限的系统。

大多数虚拟机（包括 Java 虚拟机）与 Dalvik 虚拟机不同，前者是栈机（Stack Machine），而 Dalvik 虚拟机是基于寄存器的架构。就像 CISC 与 RISC 的争论，这两种方式的相对优点是一个不断争论的话题，而且有时技术界限会变得模糊不清。此外，两种方法的相对优势取决于所选择的解释 / 编译策略。但是，总的来说，基于栈的虚拟机必须使用指令来载入栈上的数据，或使用指令来操纵数据，因此与基于寄存器的虚拟机相比，需要的指令更多。

dx 的工具用于转换 Java 的 .class 文件到 .dex 格式。多个类文件可包含到单个的 .dex 文件中。重复的可用于多个类的字符串和其他常量在转换到 .dex 格式时输出到保留空间。Java 字节码还可转换成可选择的、Dalvik VM 使用的指令集。一个未压缩的 .dex 文件在文件大小方面往往比从同样的 .class 文件压缩成的 .jar 文件更小。

为满足低内存要求而不断优化的 Dalvik 虚拟机，有一些有别于其他标准虚拟机的独特特征：

1）虚拟机很小，使用的空间也小；

2）Dalvik 没有 JIT 编译器；

3）常量池已被修改为只使用 32 位的索引，以简化解释器；

4）使用自己的字节码，而非 Java 字节码。

此外，Dalvik 被设计来满足可高效运行多种虚拟机实例。

（4）Android 程序库

如图 1-1 “LIBRARY” 一栏所示，Android 包含一些 C/C++ 库，这些库能被 Android 系统中不同的组件使用。它们通过 Android 应用程序框架为开发者提供服务。以下是一些核心库。

系统 C 库：一个从 BSD 继承来的标准 C 系统函数库（libc），它是专门为了基于嵌入式 Linux 设备所定制的。

媒体库：基于 PacketVideo OpenCORE，该库支持多种常用的音频、视频格式回放和录制，同时支持静态图像文件。编码格式包括 MPEG4、H.264、MP3、AAC、AMR、JPG、PNG。

Surface Manager：对显示子系统进行管理，并且为多个应用程序提供 2D 和 3D 图层的无缝融合。

LibWebCore：一个最新的 Web 浏览器引擎，支持 Android 浏览器和一个可嵌入的 Web 视图。

SGL：底层的 2D 图形引擎。

3D libraries：基于 OpenGL ES 1.0 APIs 实现；该库可以使用硬件 3D 加速（如果可用）或者使用高度优化的 3D 软加速。

FreeType：位图（Bitmap）和矢量（Vector）字体显示。

SQLite：一个对于所有应用程序可用、功能强大的轻量级关系型数据库引擎。

（5）Android 内核

Android 的核心系统服务依赖于 Linux 2.6 内核，如安全性、内存管理、进程管理、网络协议栈和驱动模型。Linux 内核也同时作为硬件和软件之间的抽象层，如图 1-1 “LINUX KERNEL”一栏所示。

1.2.2 Android 开发环境

Android 为开发者提供了 SDK(Software Development Kit) 和 NDK(Native Development Kit) 以便开发。

SDK 基于 Java 实现，在 Dalvik 虚拟机中运行。第三方应用的开发者可以通过编写 Java 程序调用 SDK 提供的资源来实现新的程序。

NDK 主要用于供 Java 程序调用的 C/C++ 模块，调用通过 Dalvik 虚拟机支持的 JNI 完成。

本书主要关注基于 SDK 的第三方应用开发。Google 为第三方开发者提供的 SDK 不仅包括完善的事件响应机制和各种资源的控制类，而且包含用于运行和调试的模拟器和工具集。对于使用 Eclipse 开发环境的来说，Google 还提供了相应的 Eclipse 插件来提供图形化的 IDE。具体信息可参阅：http://developer.android.com 或 http://androidappdocs.appspot.com。

1.3 Android 最新进展

在 Android 开发方面，Android SDK 1.6 r1 和 Android NDK 1.6 r1 已发布，Eclipse 插件 ADT Plugin 已更新至 0.9.3 版本。Android SDK 1.6 的 API Level 是 4，而 1.5 的是 3。SDK 1.6 的工程目录与 1.5 相同，所以大多数基于 SDK 1.5 的应用程序可以不经修改直接在 SDK 1.6 修改，但如果涉及 API Level 的问题，就需要进行具体的分析。

在 Android 终端设备方面，世界上第一款 Android 手机 G1 出自 HTC，此后 HTC 陆续推出了 HTC Magic G2、HTC Hero、HTC Tatoo 等机型，功能日臻强大。此外，摩托罗拉、三星、索尼爱立信、LG、华为、联想等大牌设备厂商也在积极筹备 Android 手机的上市。

在国内，中国移动最早启动了基于 Android 平台的手机操作系统 OMS(Open Mobile System) 的研发。而中国联通和中国电信也紧追不舍，相应展开了基于 Android 的手机操作系统研发。运营商在开发 Ophone(中国移动)、Uphone(中国联通)、Cphone(中国电信) 等手机操作系统的同时，也在积极部署相应的软件商店。软件商店是利润的主要来源，而优秀的应用程序是构成软件商店的基本要素，这就为国内的移动开发者提供了非常好的机遇。

第 2 章
图片浏览器

本章知识点：

（1）自定义的图片管理视图。

（2）图片放大、缩小的处理方式。

本章难点：

如何获取图片的缩略图。

```
Rect src = new Rect
(0 + halfDeltaW, 0 + halfDeltaH, bw- halfDeltaW, bh - halfDeltaH);
Rect dst = new Rect(left, top, left + w, top + h);
mCanvas.drawBitmap(b, src, dst, mPaint);
```

在这句话有 4 个参数，第 1 个参数为要显示的 Bitmap，src 为显示 Bitmap 的图像区域，可以为空；第 3 个参数 dst 为缩放后图片的显示区域，其缩放后的大小就为 dst 中的 w 和 h。

使用 drawBitmap( ) 的方式只能将图片缩放后显示在画布上，并且不能保存缩放后的图片，使用下列代码可以保存缩放后的图片：

```
// 先从资源文件中解析图片为 Bitmap
Bitmap bitmap = BitmapFactory.decodeResource(getResources(),rid);
// 获取图片实际大小
int iHeight = bitmap.getHeight();
int iWidth = bitmap.getWidth();
float scaleW = 1;
float scaleH = 1;
// 如果是将图片缩放为 100*100 的，获取它的缩放比列
double scalex = (float)100/iWidth;
double scaley = (float)100/iHeight;
scaleW = (float)(scaleW*scalex);
scaleH = (float)(scaleH*scalex);
// 生成一个矩阵对象。
Matrix matrix = new Matrix();
matrix.postScale(scaleW, scaleH);
// 获取缩放后的 Bitmap
newBmp = Bitmap.createBitmap(bitmap, 0,0,iWidth, iHeight,matrix, true);
iHeight = newBmp.getHeight();
iWidth  = newBmp.getWidth();
```

```
// 将缩放后的图片保存在 SDCard 上
File file = new File("/sdcard/battery_002.bmp");
BufferedOutputStream bos = new BufferedOutputStream(new FileOutputStream(file));
newBmp.compress(Bitmap.CompressFormat.PNG, 100, bos);
bos.flush();
bos.close();
```

2.1 案例背景

随着 3G 的普及以及移动终端智能化的提高，用户移动设备中的多媒体内容会越来越多，图片就是其中重要的一项，所以图片浏览器就成为移动平台上比较重要的一类应用。

本章介绍的图片浏览器提供一个可以上下滑动的缩略图列表，用户可以看到目录下的所有图片的缩略图，并可以通过点击某一图片在列表右边查看图片放大的效果。

如图 2-1 所示，用户可以很方便地通过上下按键来控制列表滚动，右边的预览框会显示位于列表顶端的图片。另外，用户可以单击视野内的任一缩略图以预览其放大效果。

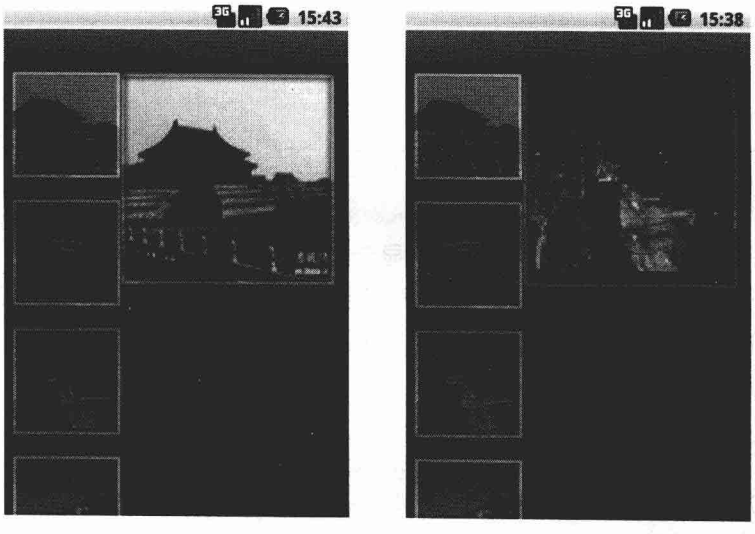

图 2-1 图片浏览器示意

2.2 案例设计与实现

2.2.1 自定义视图

本例使用了自定义的 View，对滑动、绘图、图片管理等做了定制化的处理。

```
<view class="com.xmobileapp.pictureviewer.PictureShow$PictureSlideView"
        android:id="@+id/grid"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:listSelector="@drawable/grid_background"
/>
```

如上所示，在界面 Activity(本例将其命名为 PcitureShow) 中，本例又定义了类 PctureSlideView 来实现本例用到的自定义 View，并且 PictureSlideView 中又定义了一些类来实现对图片的读取和绘制。

2.2.2　逻辑控制

本例的逻辑控制是基于基本的 Activity 与 View 的回调函数机制的。控制逻辑如下所示。

本例只有一个界面，没有涉及界面转换问题，所以只需要一个 Activity 就足够。在界面 Activity 中，本例主要通过 onKeyUp 和 onKeyDown 来实现对用户行为的反应，并调用自定义 View(PictureSlideView) 对象的相应方法来实现缩略图列表的上下滑动。

本例的 Activity 类是 PictureShow，主要变量和回调函数实现如下：

```
private PictureShowUtils mAllImages = new PictureShowUtils();
boolean mLayoutComplete;
boolean mPausing = false;
PictureSlideView mPictureSlideView;

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // 设置标题的样式
    requestWindowFeature(Window.FEATURE_CUSTOM_TITLE);
    setContentView(R.layout.main);
    mPictureSlideView = (PictureSlideView) findViewById(R.id.grid);
    mPictureSlideView.requestFocus();
}

@Override
public boolean onKeyUp(int keyCode, KeyEvent event) {
    return super.onKeyUp(keyCode, event);
}

@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    int sel = mPictureSlideView.mSelImageIndex;
    int count = mAllImages.getCount();

    switch (keyCode) {
    case KeyEvent.KEYCODE_DPAD_UP:
        if (sel >= 1) {
          sel--;
          mPictureSlideView.ScrollUp();
        }
        break;
    case KeyEvent.KEYCODE_DPAD_DOWN:
        if (sel < count - 1) {
          sel++;
          mPictureSlideView.scrollDown();
        }
```

```
            break;
        default:
            break;
        }
        // 将用户当前的选择传给自定义的 View
        mPictureSlideView.mSelImageIndex = sel;
        // 视图预览区域载入当前选择的图片
        mPictureSlideView.mImageBlockManager.loadSelBlock(sel);
        // 重画 Canvas
        mPictureSlideView.invalidate();

        return super.onKeyDown(keyCode, event);
    }

    @Override
    public void onPause() {
        super.onPause();
        mPausing = true;
    }

    @Override
    public void onResume() {
        super.onResume();
        mPausing = false;
    }
```

上面的代码用到了本例最为关键的类：PictureSlideView，一个集成了图片管理（类 ImageBlockManager）的自定义 View。

PictureSlideView 中嵌套了 2 个类：LayoutSpec 和 ImageBlockManager。

LayoutSpec 用于管理布局信息，代码如下：

```
public static class PictureSlideView extends View {
    // 布局信息
    class LayoutSpec {
        LayoutSpec(int cols, int w, int h, int leftEdgePadding,
                int rightEdgePadding, int intercellSpacing) {
            mColumns = cols;
            mCellWidth = w;
            mCellHeight = h;
            mLeftEdgePadding = leftEdgePadding;
            mRightEdgePadding = rightEdgePadding;
            mCellSpacing = intercellSpacing;
        }

        int mColumns;
        int mCellWidth, mCellHeight;
        int mLeftEdgePadding, mRightEdgePadding;
        int mCellSpacing;
    };
```

类 ImageBlockManager 用于管理 View 中图片的读取、显示、操作等。并且，ImageBlock

Manager 又内嵌了一个 ImageBlock 类来管理界面中单个图片（代码中成为 Block）的信息。下文将详细介绍相关的控制逻辑。

（1）缩略图显示

ImageBlockManager.loadBlocks，读取所有缩略图。

ImageBlock.loadBlock，读取某一个缩略图。

ImageBlock.drawBlockBitmap，绘制图片。

ImageBlock.paintBorder，绘制边框。

```
private class ImageBlockManager{
//...
    private int mBlockCacheFirstBlockNumber = 0;
    private int mBlockCacheStartOffset = 0;
    // 缩略图缓存
    private ImageBlock[] mBlockCache;
    // 选定的缩略图对象
    private ImageBlock mSelBlock;
    // 一屏上缩略图的行数
    private static final int sRowsPerPage = 4;
    // 缩略图缓存大小参数
    private static final int sPagesPreCache = 2;
    private static final int sPagesPostCache = 2;
    // 放大预览图的尺寸
    private static final int sSelBlockWidth = 200;
    private static final int sSelBlockHeight = 200;

    ImageBlockManager(Context context) {
        mContext = context;

        mBlockCache = new ImageBlock[sRowsPerPage * (sPagesPreCache + sPagesPostCache + 1)];
        for (int i = 0; i < mBlockCache.length; i++) {
            mBlockCache[i] = new ImageBlock();
        }

        /* 读取图片 */
        loadBlocks();

        mSelBlock = new ImageBlock(sSelBlockWidth, sSelBlockHeight);

    }
    // 加载所有的缩略图
    private void loadBlocks() {
        ImageBlock[] blocks = mBlockCache;
        int numBlocks = blocks.length;
        int first = 0;

        for (int i = 0; i < numBlocks; i++) {
            int j = first + i;
            if (j >= numBlocks)
                j -= numBlocks;
            ImageBlock b = blocks[j];
```