# Android开发教程

极客学院出版

# 前言

Android 是一个专门针对移动设备的软件集，它包括一个操作系统，中间件和一些重要的应用程序。Android SDK 提供了在 Android 平台上使用 JaVa 语言进行 Android 应用开发必须的工具和 API 接口。

## 适用人群

本教程为初级教程，内容比较容易理解，适合 Android 初学者学习

## 学习前提

学习本教程前，需要你对 C 语言、java 基础有所了解。

> 鸣谢：引路蜂移动软件

## 版本信息

> 注: - 安装最新版 JDK ，只有 JRE（Java 运行时）是不够的。 - 下载 Eclipse 3.3 (Europa), 3.4 (Ganymede) - 下载最新版 Android SDK - 下载最新版 Eclipse ADT 插件 - 添加 Android 平台和其它组件。安装 ADT 插件后，在 Eclipse 的 Windows 菜单下选择Android SDK and AVD Manager。

# 目录

# 1

# 概述

自 Google 推出 Android 手机平台以来，采用 Android 作为平台的手机和平板电脑越来越普及。下图是2010年9月和12月三个月几个主流智能手机平台在美国的市场占有率图表。

**Top Smartphone Platforms**
**3 Month Avg. Ending Dec. 2010 vs. 3 Month Avg. Ending Sep. 2010**
**Total U.S. Smartphone Subscribers Ages 13+**
**Source: comScore MobiLens**

| | Share (%) of Smartphone Subscribers | | |
|---|---|---|---|
| | **Sep-10** | **Dec-10** | **Point Change** |
| *Total Smartphone Subscribers* | *100.0%* | *100.0%* | *N/A* |
| RIM | 37.3% | 31.6% | -5.7 |
| Google | 21.4% | 28.7% | 7.3 |
| Apple | 24.3% | 25.0% | 0.7 |
| Microsoft | 9.9% | 8.4% | -1.5 |
| Palm | 4.2% | 3.7% | -0.5 |



可以看到到2010年12月，Android 市场占有率在美国已超过 Apple 的 iPhone，而且由于 Android平台的开放性，个人认为将来它会更加普及。相当于其它常见的手机开发平台，如 Java ME，Windows Mobile ，BlackBerry， iPhone ,Winodws Phone 7。 Android 开发还是比较容易上手的。

# 什么是 Android?

Android 是一个专门针对移动设备的软件集，它包括一个操作系统，中间件和一些重要的应用程序。Android SDK 提供了在 Android 平台上使用 JaVa 语言进行 Android 应用开发必须的工具和 API 接口。

## Android 特性

- 应用程序框架支持组件的重用与替换
- Dalvik 虚拟机专为移动设备优化
- 集成的浏览器基于开源的WebKit 引擎
- 优化的图形库包括定制的 2D 图形库，3D 图形库基于 OpenGL ES 1.0（硬件加速可选）
- SQLite 用作结构化的数据存储
- 多媒体支持包括常见的音频、视频和静态图像格式（如 MPEG4, H.264, MP3, AAC, AMR, JPG, PNG,GIF）
- GSM 电话技术（依赖于硬件）
- 蓝牙 Bluetooth, EDGE, 3G,和 WiFi（依赖于硬件）
- 照相机，GPS，指南针，和加速度计（accelerometer）（依赖于硬件）
- 丰富的开发环境包括设备模拟器，调试工具，内存及性能分析图表，和 Eclipse 集成开发环境插件。

## 应用程序框架

开发人员也可以完全访问核心应用程序所使用的 API 框架。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它的应用程序都可以使用其所发布的功能块（不过得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的替换程序组件。 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）， 甚至可嵌入的 web 浏览器。
- 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（layoutfiles）。
- 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。

## 程序库

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 ）， 它是专门为基于 embedded linux 的设备定制的。
- 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 。
- SurfaceManager – 对显示子系统的管理，并且为多个应用程序提供了 2D 和 3D 图层的无缝融合。
- LibWebCore – 一个最新的 web 浏览器引擎用，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，功能强劲的轻型关系型数据库引擎。

## Android 运行库

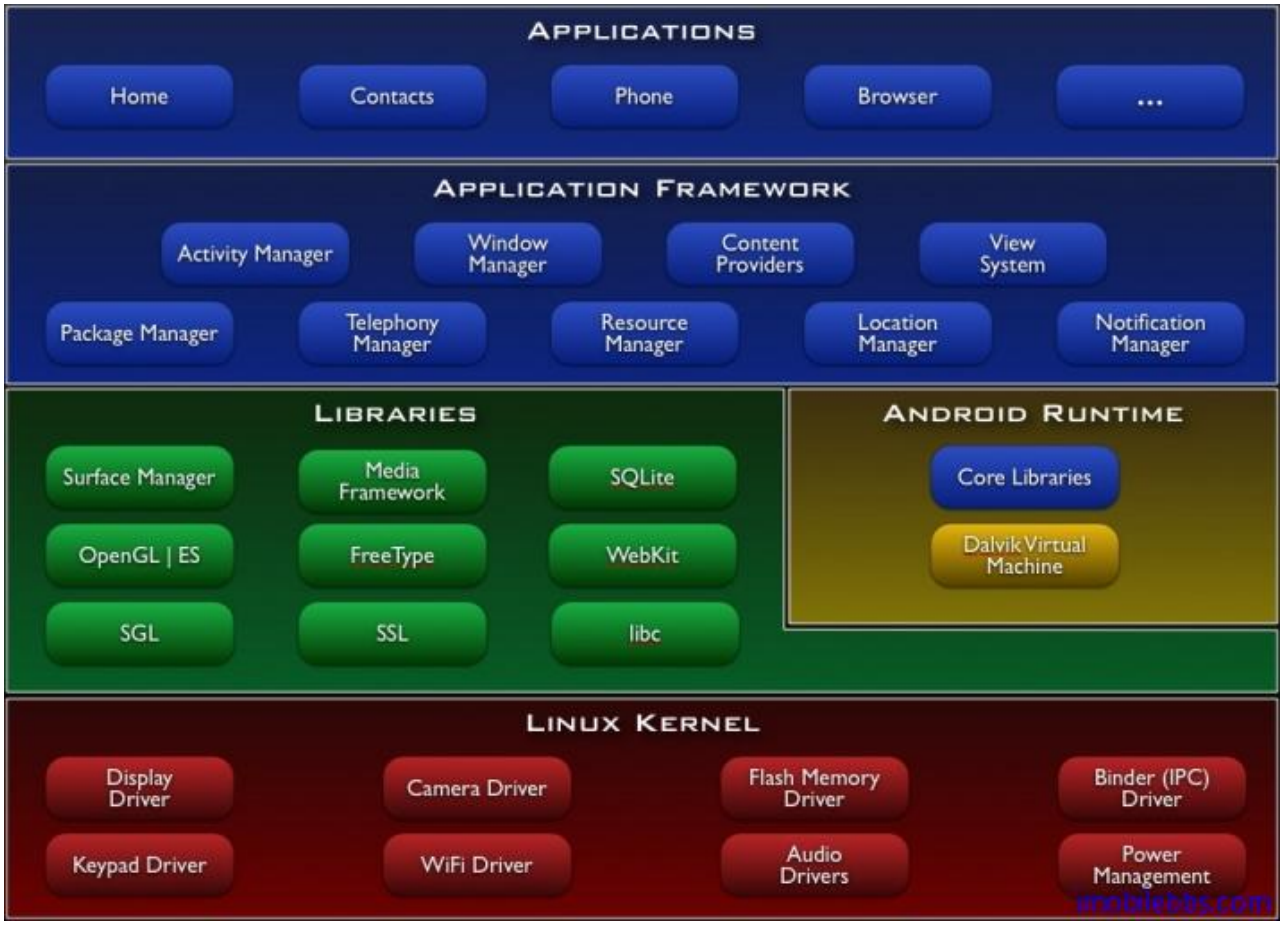
Android 包括了一个核心库，该核心库提供了 JAVA 编程语言核心库的大多数功能。 每一个Android 应用程序都在它自己的进程中运行，都拥有一个独立的 Dalvik 虚拟机实例。Dalvik 被设计成一个设备可以同时高效地运行多个虚拟系统。Dalvik 虚拟机执行（.dex）的 Dalvik 可执行文件，该格式文件针对小内存使用做了优化。同时虚拟机是基于寄存器的，所有的类都经由 JAVA 编译器编译，然后通过 SDK 中的”dx” 工具转化成 .dex 格式由虚拟机执行。 Dalvik 虚拟机依赖于 linux 内核的一些功能，比如线程机制和底层内存管理机制。

## Linux 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栈之间的抽象层。

## Android 应用和框架

下图显示了 Android 系统的主要组成部分。




核心应用，例如联系人，电子邮件，电话，浏览器，日历，地图， … 充分访问所有核心应用框架 API C/C++ 库：被各种 Android 组件使用通过应用程序框架开发者可以使用其功能包括： 媒体库：MPEG4 H.264 MP3 J PG PNG ….. WebKit/LibWebCore：Web 浏览引擎 SQLite 关系数据库引擎 2D，3D 图形库、引擎

Android 使用 Java 作为开发语言，而且有很大一部分库与 Java SE 共有，但不同于 Java ME，Dalvik 虚拟机也不是 Java 虚拟机。Eclipse 是 Android 推荐的开发 IDE，Android 平台自带的各种应用如联系人，电子邮件，电话，浏览器，日历，地图都可以重写。

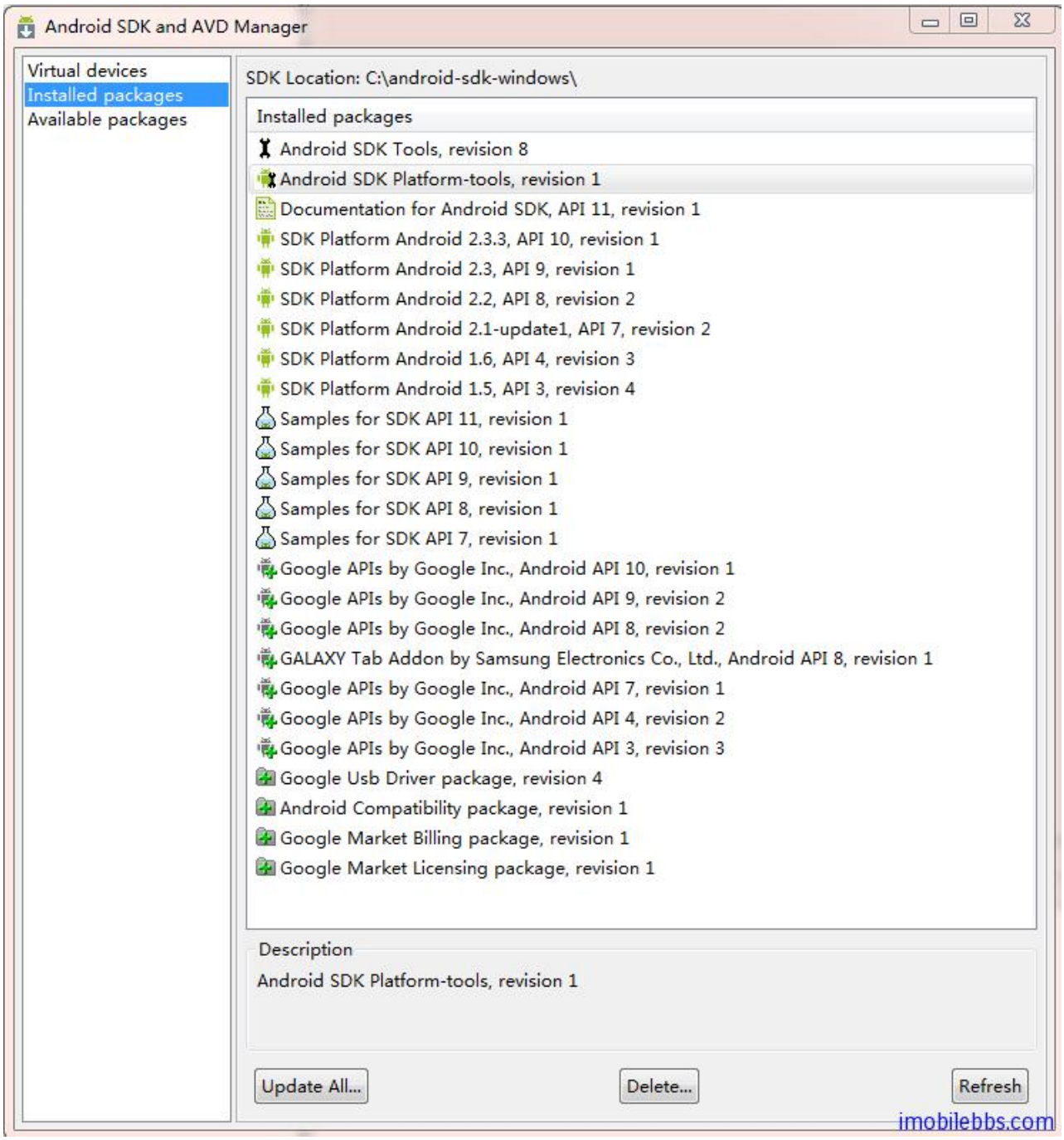
# 2

# 安装开发环境

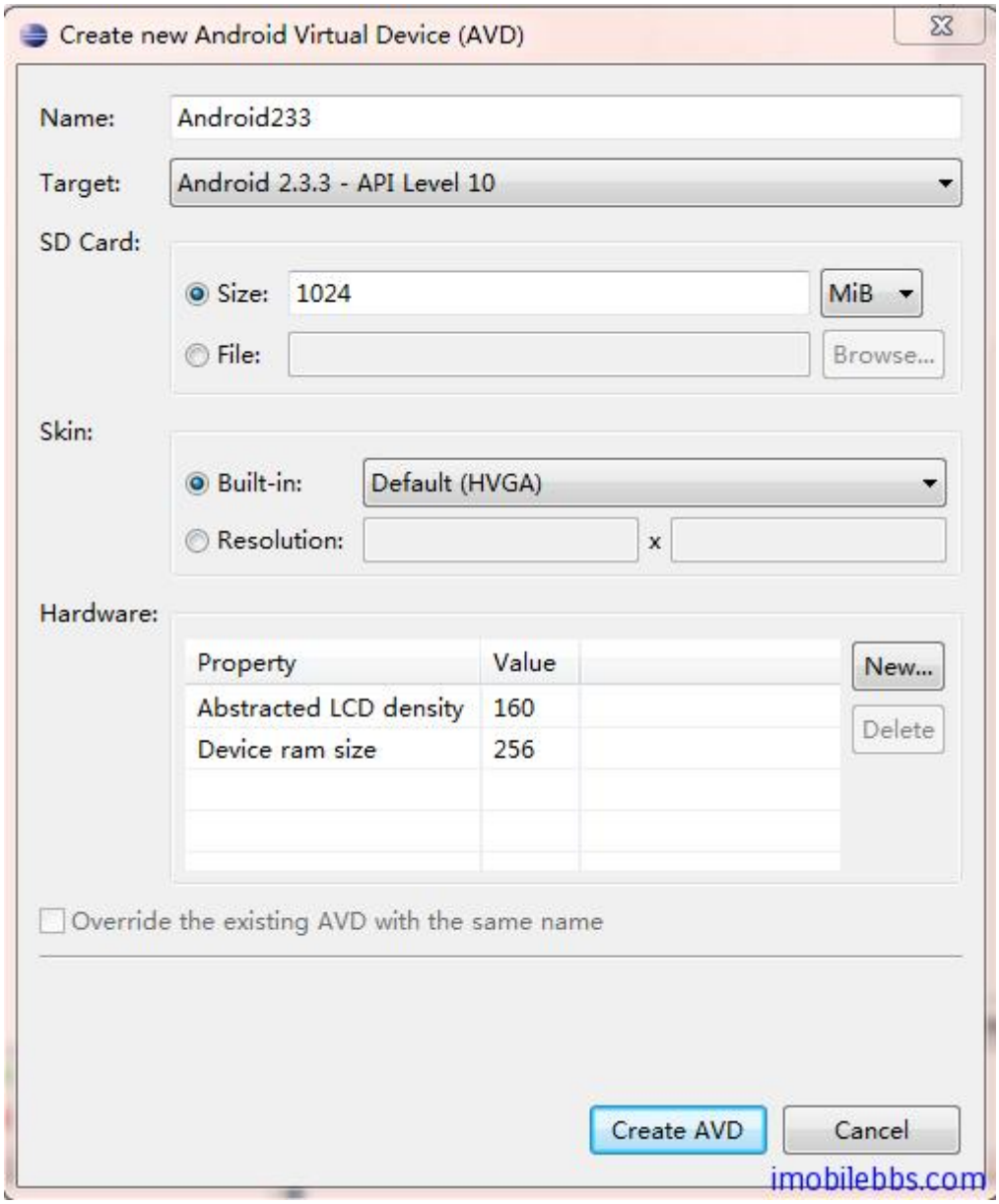
前面 Android 简明开发教程一：概述简要的介绍了 Android 平台，本篇说明如何安装搭建 Android开发环境。

Android 开发支持 Windows（ Windows XP (32−bit), Vista (32− , 64−bit), Windows 7 (32− , 64−bit) ），Mac OS（ >10.5.8)，Linux( Ubuntu Linux, Lucid Lynx etc). 开发Java 应用最常用的是 NetBean 和 Eclispe。但 Google 推荐使用的是 Eclipse。虽然也有支持Android 的 Netbean 插件，但兼容性不是很好。所以最好还是使用 Eclipse。

1. 安装最新版 JDK ，只有 JRE（ Java 运行时 ）是不够的。
2. 下载 Eclipse 3.3 (Europa), 3.4 (Ganymede)
3. 下载最新版 Android SDK
4. 下载最新版 Eclipse ADT 插件
5. 添加 Android 平台和其它组件。安装 ADT 插件后，在 Eclipse 的 Windows 菜单下选择Android SDK and AVD Manager。




在 Avaiable Packages 选择你想安装支持的 Android 平台和一些工具，如果你的硬盘足够大，简单的方法是全部选中，另外，尽管开发 Android 应用可以使用模拟器，但最终测试还是要在手机上进行，为了能够在设备上运行调试，请确保选中 Google Usb Driver Package。也可以在需要时再安装，Android SDK and AVD Manager 可以运行多次更新。 6. 创建模拟器 ，在 Virtual Devices 下可以管理 Android 模拟器

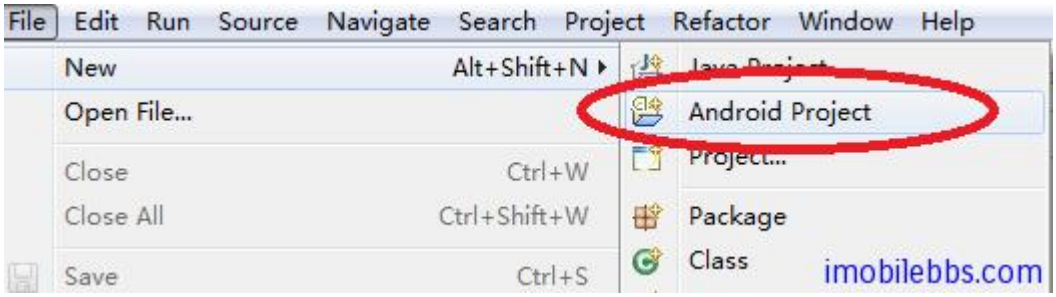



创建模拟器时指定目标 Android 平台的版本，目前市面上 Android 的版本从1.5到2.3多有，一般来说高版本向下兼容低版本。可以为不同的 Android 平台版本都创建一个虚拟机，建议创建两个以上，这样对于测试 SMS，Phone Call 等应用就比较方便。

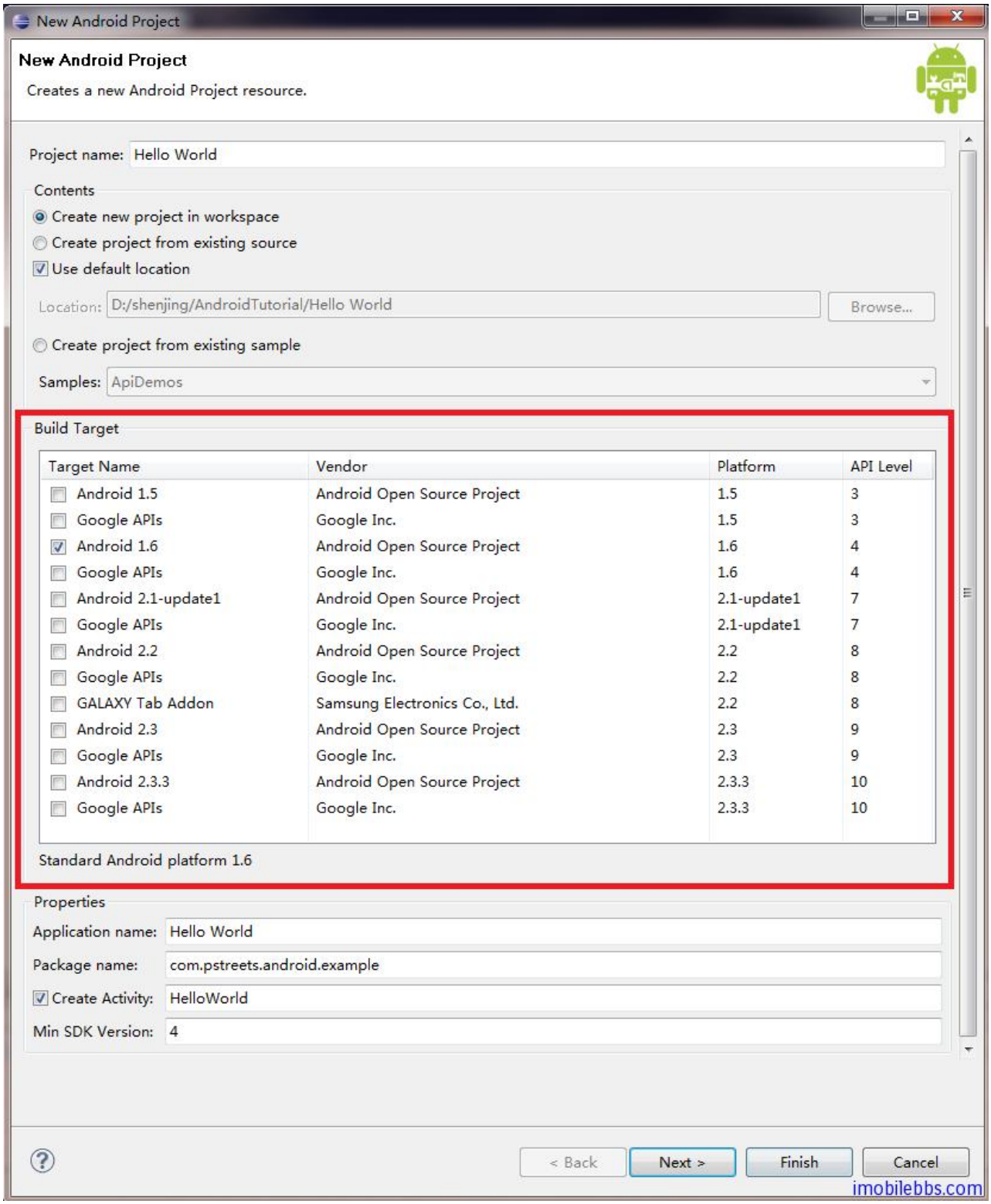
# 3

# 第一个应用 Hello World

在安装后 Android 开发环境和创建好 Android 模拟器之后，就可以开始写第一个 Android 应用“Hello,World”。后面的例子均采用 Eclipse IDE。 安装 ADT plugin 之后，创建的新项目种类就会增加一个 Android Project 类型：




选择 Android Project 项目类型，出现下面对话框：




Project Name： Hello World

Build Target： 这里选择 Android 1.6，如果你的 Build Target 列表为空，则表示你忘记设置Android SDK 安装目录了。可以通过 Windows -> Preferences -> Android 来设置 SDK 路径。

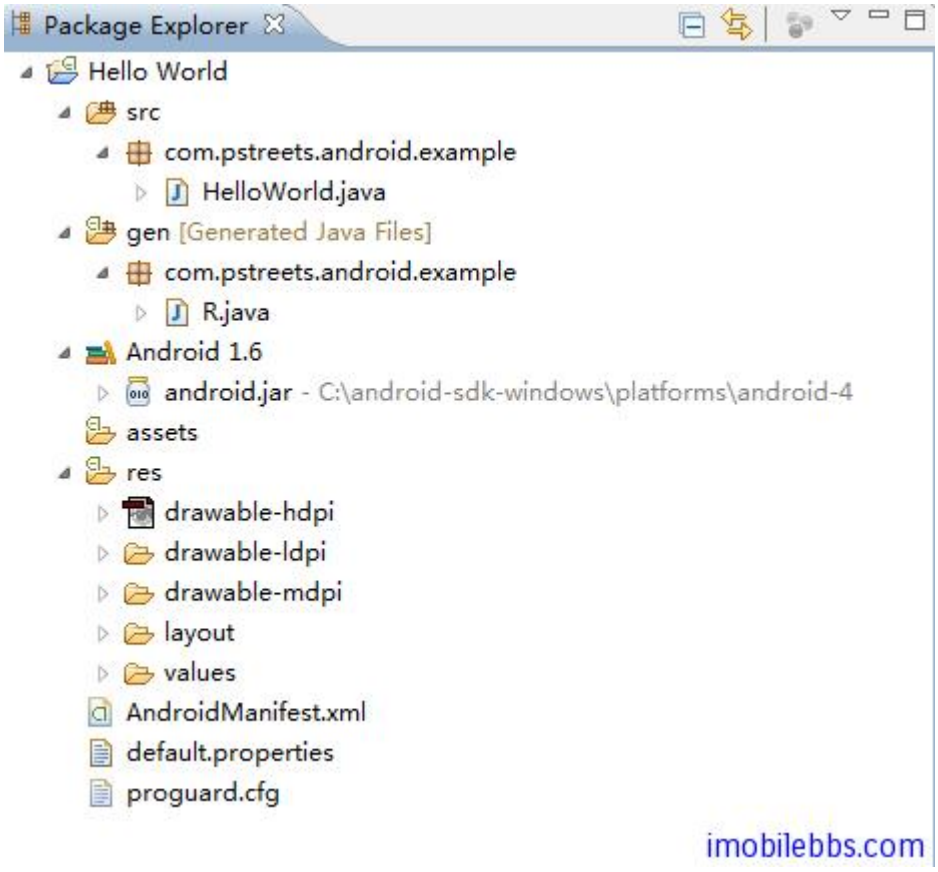
Application Name: Hello World

Package name: com.pstreets.android.example, 如果您开发过 Java 或是 .Net Framework 应用，包名称并不陌生。

Create Activity： HelloWorld。 Activity 是 Android 平台中特有的一个新概念。以 Java ME或是 Windows Mobile CE 应用作参考，它类似于 Java ME 和 Windows Mobile 中 UI 类的 Form类。

Min SDK Version: 可以为空。 Android 平台的版本比较多，从1.5到目前的3.0。Android 平台支持向下兼容。Min SDK Version指出了您开发应用支持的最第版本。4对应于 Android 1.5。

点击“Finish”则在 Eclipse 的 Workspace 中创建了“Hello World”项目：




ADT Plug 自动创建了几个目录：

src 应用源码目录

gen Android 应用自动生成的代码，主要是根据 Android 资源目录 res 下的资源来生成的，这样可以根据资源 ID 来访问应用中的资源。一般不建议手工改动，即使改动，下次编译时也会被重新覆盖。

Android 1.6 表示当前选择的 Android 版本是 Android 1.6，你可以使用 Android 1.6 中提供的API。可以通过项目的属性来修改 Android 版本。

assets 静态文件目录。Hello world 中为空。

res 为应用中的资源目录，res 中含有多个子目录，为多种资源。如果你曾经使用 Silverlight, Polish Java ME 或是 WPF 等使用 XML 来描述 UI 的应用，则您会觉得 res 目录下的各种资源文件似曾相识。Android 也是采用

XML 来描述 UI 的。
AndroidManifest.xml 应用程序描述文件，类同于 Java ME 的 JAD 文件。它定义了应用的构成，组件，权限等信息。
default.properties 和 proguard.cfg 一般不需要改动。proguard.cfg 主要用来扰码(混淆器）来保护应用防止反编译。开发过 Java 或是 .Net 应用的应该对这比较熟悉。

这样就有了第一个应用“Hello World”,可以直接运行。Run As -> Android Application ，将启动模拟器，如果你有 Android 设备，则也可以选择使用 Android 设备运行。


到目前为止我们还没有写一行代码。还不能说了解开发 Android 应用的基本概念。所以需要具体了解一下这个应用的几个重要的组成部分：

主 Activity ，打开类 com.pstreets.android.example.HelloWorld

```
package com.pstreets.android.example;

import android.app.Activity;
```

```
import android.os.Bundle;

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

前面提到 Activity 是 Android 中类似 Windows Mobile 中的 Form 类的基本 UI 类。如果您开发过 Java ME 应用，Activity 更像 MIDlet，当 Android 应用可以有多个Activity，而每个Java ME应用中只能有一个MIDlet派生类。如果熟悉MVC，MVP模型，Activity类似于 MVC或是 MVP模型中的 Controller或是Presenter。Activity 有多个生命周期事件可以实现，onCreate是其中一个，它类似于Java ME MIDlet的 startApp 或是 From 的 From_Load事件。Activity将在后面在详细介绍。 setContentView(R.layout.main); 设置Activity主用户UI。

Layout 资源文件 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”
    >
<TextView
    android:layout_width=”fill_parent”
    android:layout_height=”wrap_content”
    android:text=”@string/hello”
    />
</LinearLayout>
```

Android 是通过 XML 来描述 UI 的，UI 一般通过 res 下 Layout 资源来描述 main.xml 中定义了 HelloWorld主界面。可以看到 LinearLayout 和 TextView 两个元素。这表示主界面采用LinearLayout 布局（类似 Swing 中 Layout），下面是一个 TextView（文本框），文本框显示的内容是@string/hello， @string/hello 为一个 string 资源，@表示资源引用。string 资源定义在 res->values->strings.xml 中，其值为Hello World, HelloWorld!。

View 在 Android 中表示一个可视化组件，刚接触 Android 开发时，可能会有些困惑，因为在其它平台在 View 一般指用户界面（Windows），如果拿 Java ME 或是 Windows Mobile 做类比的话，Android 中的 View 相当于 Windows Mobile 中的 Control 或是 Component， ViewGroup 相当于 Container 或是 Swing 中的 Layout。 R.layout.main 定义在 gen->R.Java 中，为自动为资源生成的资源 ID。

AndroidManifest.xml 应用程序清单

```
<?xml version=”1.0″ encoding=”utf-8″?>
<manifest xmlns:android=”http://schemas.android.com/apk/res/android”
    package=”com.pstreets.android.example”
    android:versionCode=”1″
    android:versionName=”1.0″>
  <application android:icon=”@drawable/icon” android:label=”@string/app_name”>
    <activity android:name=”.HelloWorld”
          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>
  <uses-sdk android:minSdkVersion=”4″ />

</manifest>
```

和 Java ME 的 JAD 文件类似，AndroidManifest.xml 定义了 Android 应用中所有的Activity ，应用的图标，权限等属性。

```
<intent-filter>
 <action android:name=”android.intent.action.MAIN” />
 <category android:name=”android.intent.category.LAUNCHER” />
```

表示这个 Activity 是可以通过 Android 应用菜单来启动，具体含义在介绍Activity时再说明。

此外,Android 除了使用 XML 来描述 UI 外，如果你不怕麻烦的话，也可以通过代码来创建 UI，方法类似 Swing UI。

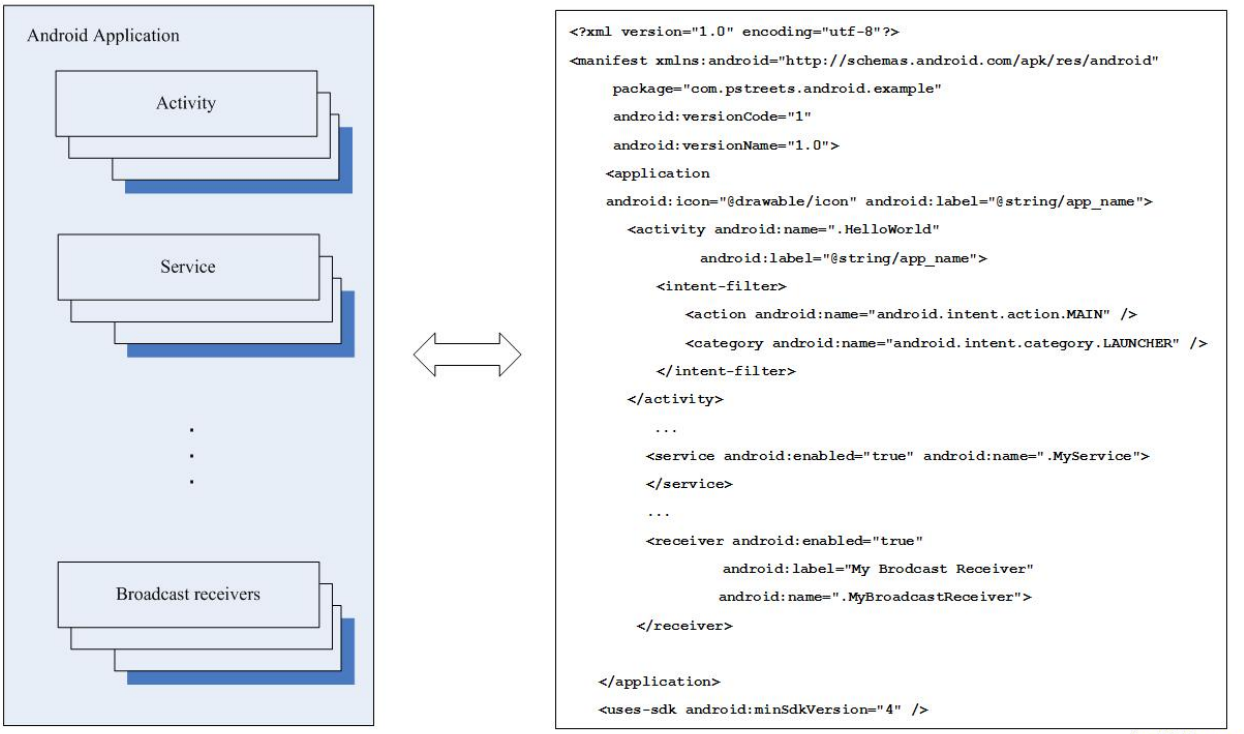
# 4

# Android 应用基本概念

Android 平台的一个显著的特点是“低耦合”。Activity 是 Android 应用的一个最基本的用户 UI 模块。如果采用 Windows Form 应用作为参照，Activity 相当于 Windows 中的 WinForm。和 Windows 应用不同的是，运行一个 Activity 或是 Activity 之间的交互是通过消息来实现的。也就是说如果想在起动一个 Activity 或是在一个 Activity 中启动另一个 Activity，是通过发送Intent 消息来触发，而不像 Windows WinForm 应用，需要调 Form 示例的 Show 或是 Load 方法来实现。通过 Intent 消息来实现 Activity 之间的交互，则最大程度上减小了模块之间的耦合度。这种机制类同 Subscriber/Publisher 机制。

Android 平台的另外一个重要特性是“重用”。一个 Android 应用可以有多个 Activity 组成。拿扑克牌做比方，Android 应用相当于扑克牌的盒子，盒子里的每张牌就是一个相对独立的 Activity。这个Android 应用运行时想当于从扑克牌中抽取牌叠放在一起，最先抽出的牌就是 Android 应用的主Activity，主 Activity 可以在调用其它 Activity（通过发 Intent 消息），被触发的 Activity 就像扑克牌一样发在主 Activity 上面。这样就形成一个“Activity”栈。在设备上按“Back”则可以如浏览器一样回到上一个 Activity。 Android 手机上每个应用都是一样的结构。“重用”指 Android 应用在运行时，可以触发其它应用中定义的 Activity。比如说在 GTalk 中想显示某个朋友在地图上的位置。而 GoogleMap 应用可以显示地图。GTalk 不需要重复同样的代码或是对于类似的 Activity。可以直接通过 Intent 消息来启动 GoogleMap 中的 MapViewActivity。

下图显示了 Android 应用的基本组成部分。




除了 Activity 之外，Android 也可以实现 Service，Service 类同 Windows Service，一般在后台运行，不含用户界面。Brodcast Receiver 可以用来响应一些系统消息。基本功能有点类似 Java ME 中的 PushRegistr

y。比方说你想在收到短信时触发你的应用，可以在 Android 应用的 Manifest 文件中定义一个 Broadcast Receiver 来触发一个 Activity。

如上图所示，Android 应用中，Application 对象好像一个容器，里面可以包含多个 Activity，多个 Service 或是多个 Broadcast Receiver。这些 Activity，Service，Broadcast Receiver 相对独立，相互之间交互只能通过 Intent 消息。如同 Java ME 的 MIDlet 的 JAD 文件一样，每个Android 应用都有一个 Manifest 文件，文件名固定为 AndroidManifest.xml。Android 应用中定义的 Activity，Service，Broadcast Receiver 等都需要定义在这个 Mainifest 文件中才能被本应用或是其它应用所调用。 这里还是借用 Publisher/Subscriber 的概念来说明。 一个 Activity，Service 等 如果能被调用的话则需要在 Mainifest 中 Subscriber 某类消息。

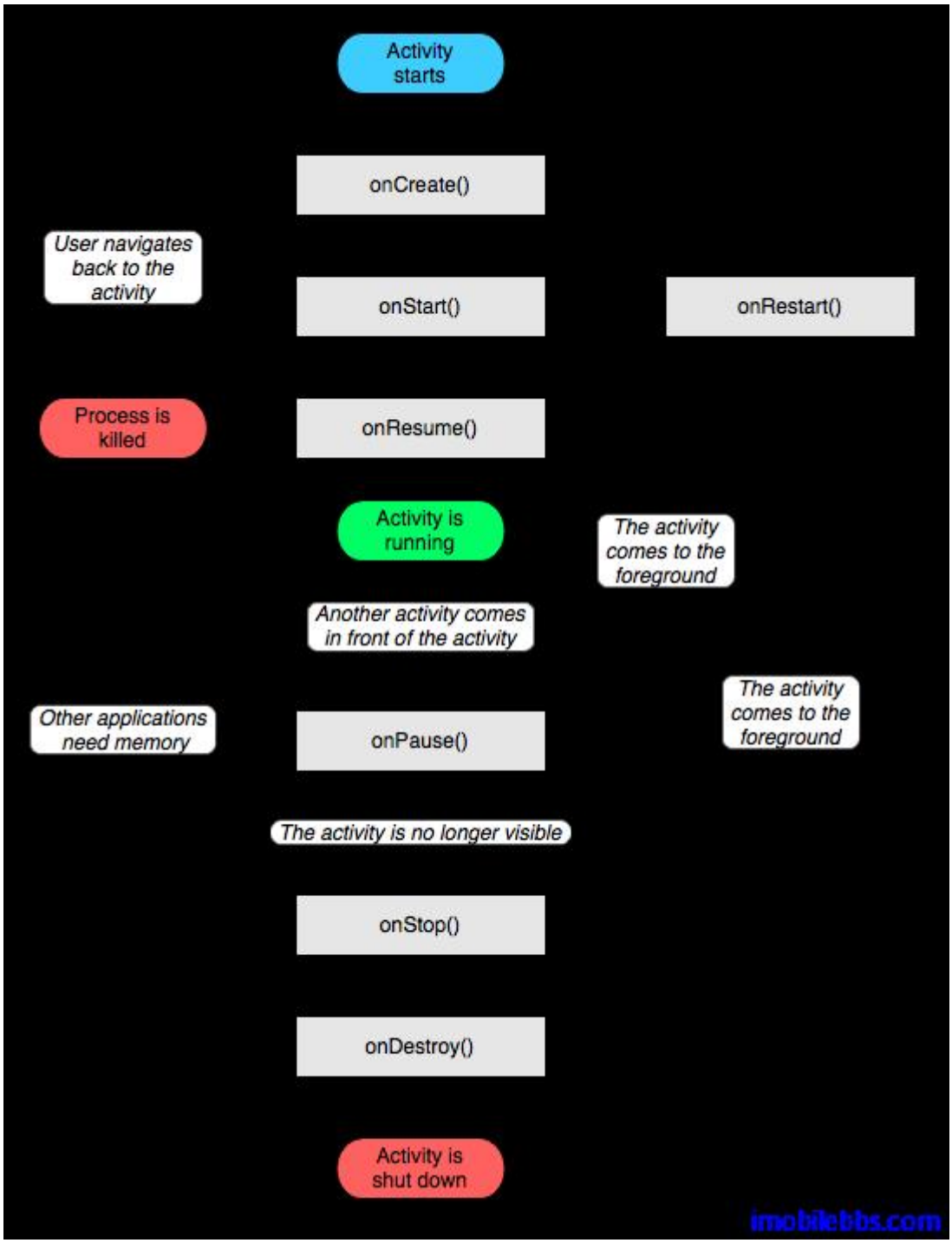
```
<activity android:name=”.HelloWorld”
            android:label=”@string/app_name”>
        <intent-filter>
          <action android:name=”android.intent.action.MAIN” />
          <category android:name=”android.intent.category.LAUNCHER” />
        </intent-filter>
    </activity>
```

上面是 HelloWorld 中主 Activity 在 AndroidManifest.xml 的定义，定义了这个 Activity 的对应的 class，以及可以触发该 Activity 的 intent-filter ，（相当于 Subscriber 某种消息），但用户点击该应用图标时，Android 操作系统将发送一个 Intent 消息，Android 系统检查subscribe 该 Intent 消息的 Activity，Service 或是 Broadcast Receiver，如果找到，则其动该 Activity，Service 或是 Broadcast Receiver。对于 HelloWorld，则在屏幕上显示“Hello World”。 除了系统可以发送 Intent 外，Android 引用也可以通过 startActivity(Intent) ,StartService(Intent)来向启动其它 Activity 或是 Service。Intent 可以带传入数据（参数）。即使在同一个应用中，也需要通过 Intent 来传送信息，这样大大降低了应用中各个模块之间的耦合度，从而可以无缝更换应用中的某个模块而不会影响其它部分。 刚开始接触 Android 这种机制时可能会觉得不如 WinForm 的 (new Form 1()).Show()来的直接方便。但从应用的可维护性，可扩展性来看，Android 这种低耦合设计是非常有利的。 此外，如果需要在多个 Activity 这间共享一些数据，可以通过扩展 Application 类实现，在 Application 类中定义的变量可以被应用中所有Activity 所访问。

# 5
# Activities

Android 应用中的 Activity 指具有屏幕显示支持用户交互的基本模块，类似于 Java ME 中的MIDlet， Windows 应用中的 Form。比如可以是拨号，发送邮件的 UI。每个 Activity 都可以含有一个 Windows 用于绘制用户界面。这个 Windows 提出占据整个屏幕，但也可以只占据部分屏幕或说现在在其它 UI 上面。

一个 Android 应用通常由多个 Activity 组成，其中有一个“主 Activity”，为用户启动应用时第一个显示的 UI。Activity 可以启动其它 Activity 来实现其它功能。新的 Activity 又可以再启动新的 Activity。新启动的 Activity 的 UI 将覆盖之前的 UI。从而形成一个“UI 栈”。新启动的Activity 将暂停上一个 Activity 的运行。当用户按“BACK”按键时，“UI 栈”最上的 Activity 出栈，之前的 UI 重新显示在屏幕上并恢复该 UI 对应的 Activity 的运行。这意味着 Activiy 具有一个“生命周期”。




写过 MIDlet 或是 Windows Mobile 应用的对以上“生命周期”不会陌生。MIDlet 也有类似的生命周期。Windows Form 也有 Load, Unload , Active 等事件。和桌面系统不太一样的说，一般来说移动应用的生命周期不受应用本身控制，而是有手机操作系统来决定。Activity 则实现对每个生命周期事件的处理来完成某个功能。比如在 onCreate()事件中调用 setContentView()来设置 UI 布局。在onPause()事件中暂停下载，使用 GPS 等，在 onResume()事件中恢复下载，重连 GPS 设备等。

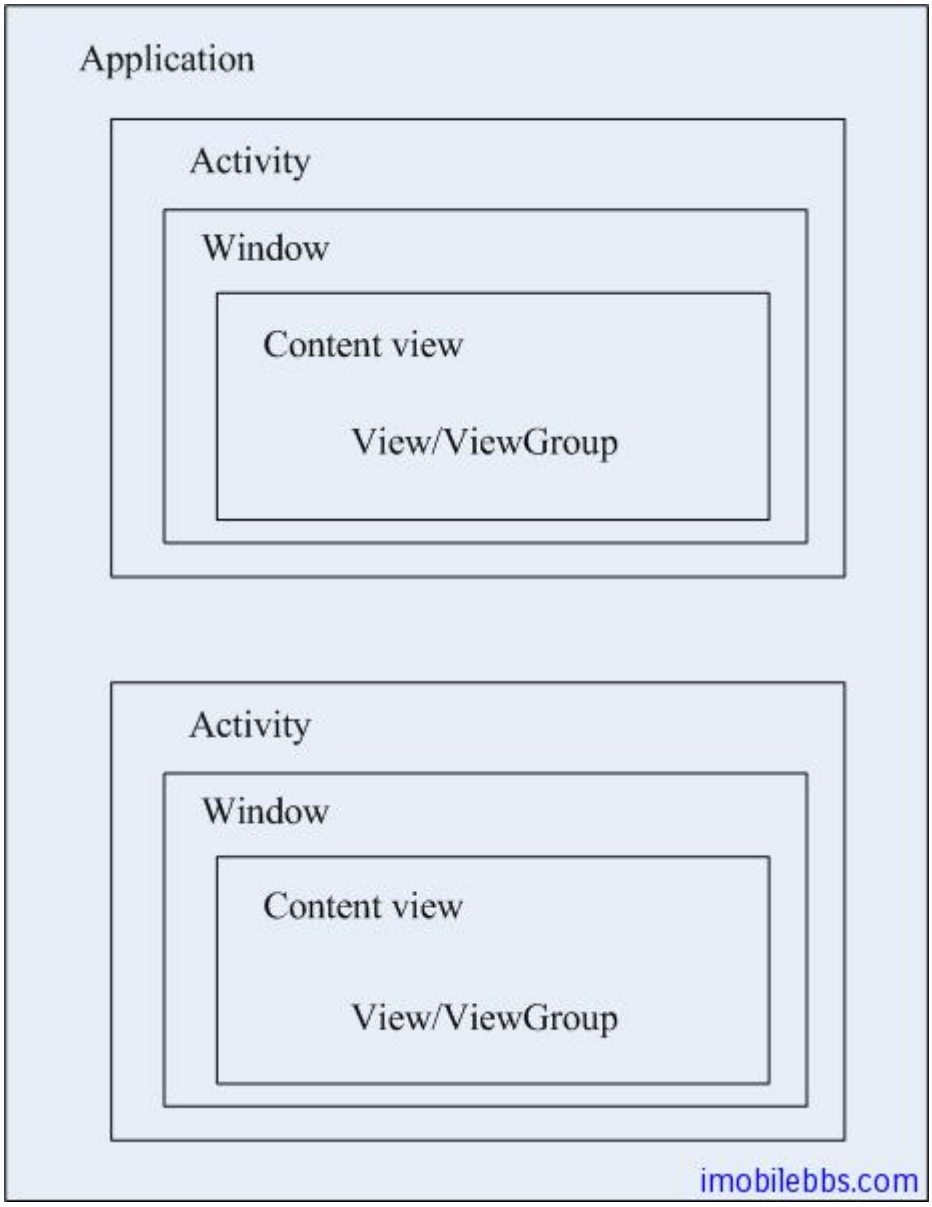
# 6

# 用户界面设计

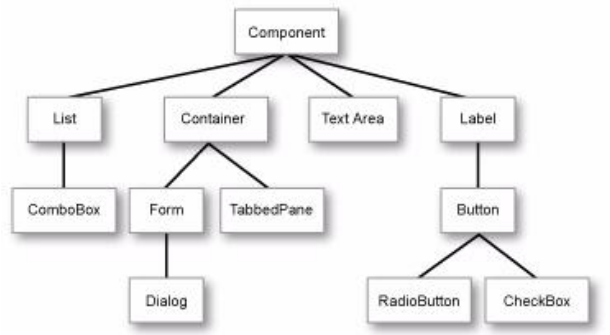
Activity 是 Android 应用用户界面的基本组成部件。但 Activity 本身并不提供用户界面(User Interface)。从程序结构层次上来说，一个 Android 应用是类 android.app.Application 的一个实例， Application 中可以包含多个 android.app.Activity 实例。每个 Activity 带一个Window 类，这个类在 Android 平台上没有提供太多功能，主要可以用来控制标题栏（屏幕顶端）。比如设置UI全屏显示可以使用如下代码：

```
requestWindowFeature(Window.FEATURE_NO_TITLE);
getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                     WindowManager.LayoutParams.FLAG_FULLSCREEN);
```

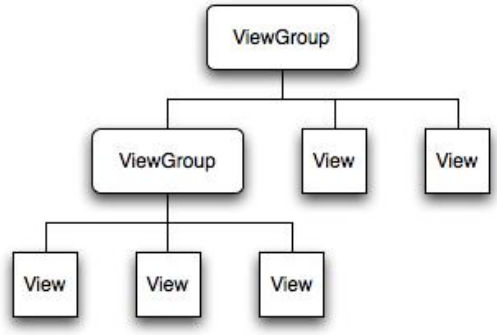
Activty 缺省是不含用户界面，如需显示用户界面，则可以调用 setContentView()来设置 Activity 的 ContentView。 ConentView 描述了具体的 UI 组件，如文本框，标签，列表框，图片框的。



Android 的用户界面其实就是指 ContentView 的设计。“View”开始会使人产生误解，在其它平台“View”一般指类似Form的概念。而在 Android 平台上 View 是 UI 组件，相当于其他平台的Component，ViewGroup 相当于其它平台的 Container，如下图所示：




Java SE/Java ME/.NET



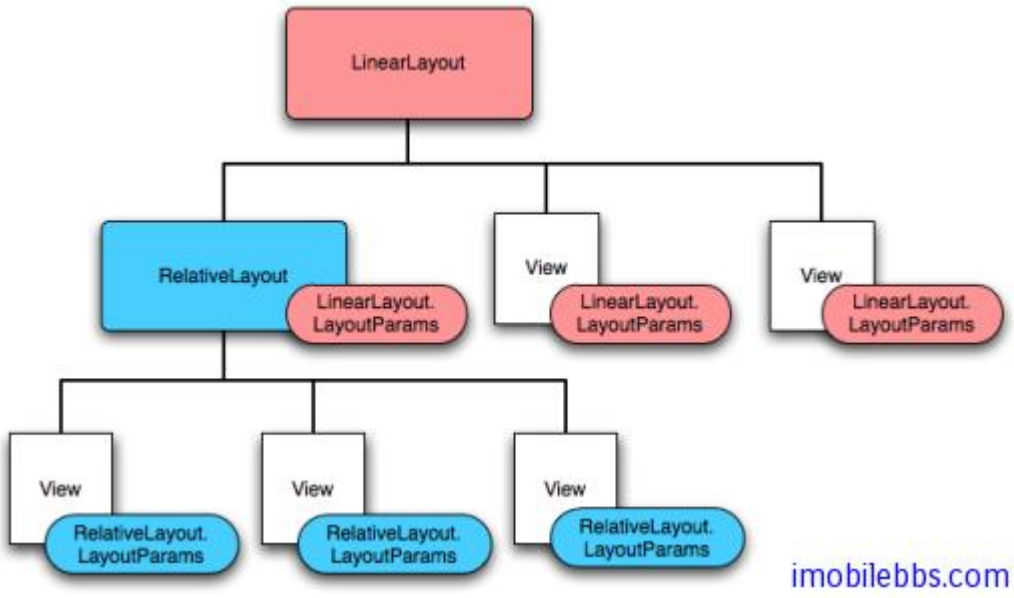

Android



有了这个对应关系就很容易将你已有的用户界面设计知识用在 Android 的用户界面设计上来。

此外 Android 用户界面设计一个推荐的方法是使用 XML 来描述 UI，这也不是 Android 平台的首创，Java ME Polish，WPF，Silverlight 等都采用 XML 来描述 UI，使用 XML 来描述的好处是将用户界面和程序逻辑分开，可以做到用户界面的改变不影响程序逻辑，程序逻辑的变动也可以不影响用户界面，实际上是采用了 MVC 模式的设计。Activity 是 MVC 中的 Controller，Activity 的ContentView 则是 MVC 中的 View。如果你不想使用 XML 来描述 UI，也可以使用代码来创建 UI，不过这种方法既麻烦，也增加了模块之间的耦合度。

理解了 Android 的 View 和 ViewGroup 之后，具体设计用户界面并不复杂，一般来说 ViewGroup定义它的子 View 的布局 Layout，也就是其它 View（文本框，标签等控件或是其它 ViewGroup）在用户界面的位置安排。如上图所示，这个层次关系可以嵌套。通过嵌套，你可以定义出任意用户界面。



Android 中的基本布局如下：

FrameLayout

最简单的布局对象

在屏幕上故意保留的空白空间，你可以之后填充一个单独的对象

例如：一个你要更换的图片

所有子元素都钉到屏幕的左上角

不能为子元素指定位置

LinearLayout

在一个方向上(垂直或水平)对齐所有子元素

所有子元素一个跟一个地堆放

一个垂直列表每行将只有一个子元素(无论它们有多宽)

一个水平列表只是一列的高度（最高子元素的高度来填充）

TableLayout

把子元素放入到行与列中

不显示行、列或是单元格边界线

单元格不能横跨行，如 HTML 中一样

AbsoluteLayout

使子元素能够指明确切的X / Y 坐标显示在屏幕上

(0,0)是左上角

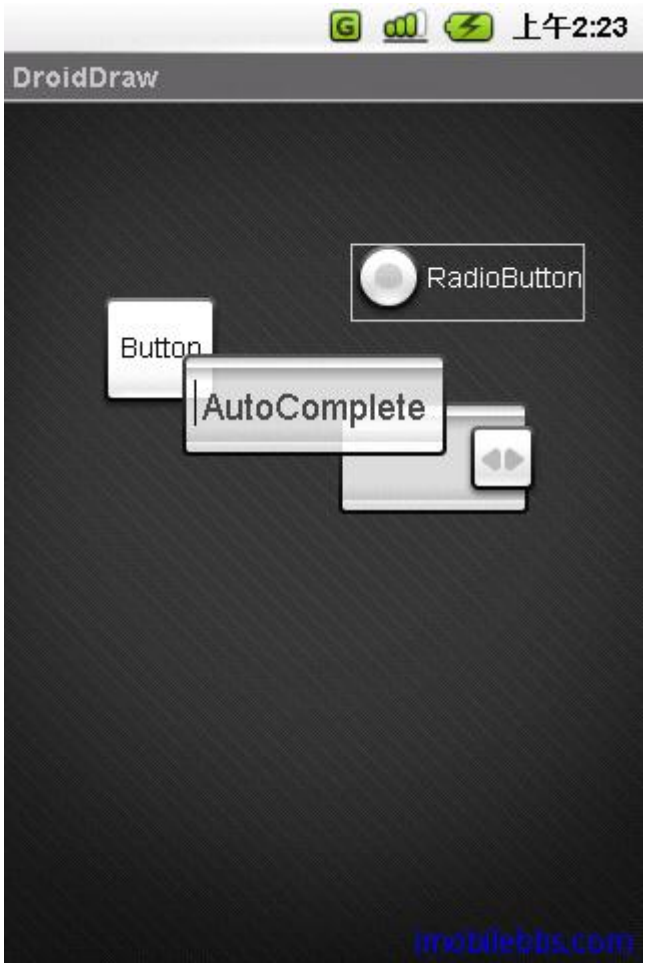
当你下移或右移时，坐标值增加

允许元素重叠(但是不推荐)

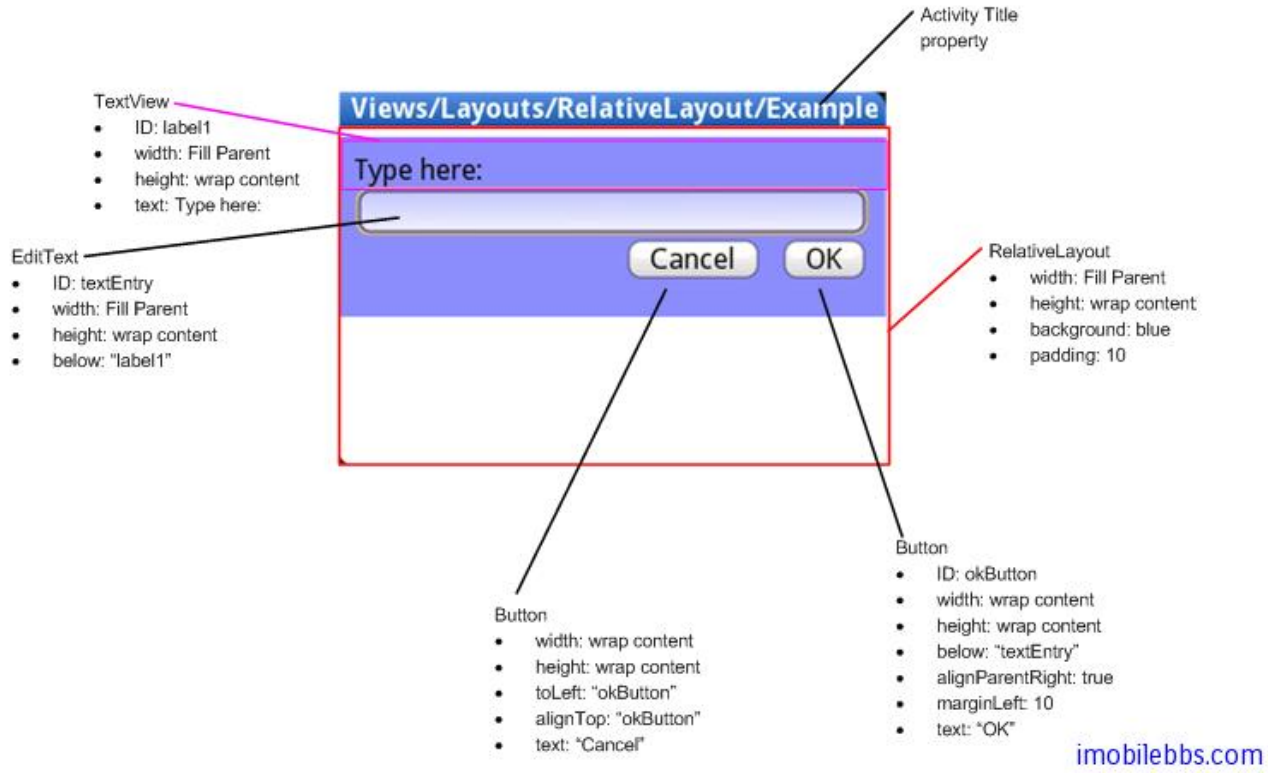
注意：

一般建议不使用 AbsoluteLayout 除非你有很好的理由来使用它

因为它相当严格并且在不同的设备显示中不能很好地工作




RelativeLayout 让子元素指定它们相对于其他元素的位置(通过 ID 来指定)或相对于父布局对象




如果不喜欢 Eclipse IDE 自带的 UI 设计工具，可以使用免费 Android UI 设计软件 DroidDraw，下载 DroidDraw。

Android SDK 的 ApiDemo 中也介绍 Android 提供的各个 UI 组件（Menu，Dialog，TextView，Button，List 等以及各个 Layout）的用法。这里就不一一介绍了。

7

# Intents 和 Intent Filters

Android 应用中的三个核心组件：Activities，Services 和 broadcast receivers 都是通过称为“Intent”的消息来激活的。Android 应用一个特点是“低耦合”，各个 Activities，Services 和broadcast receivers 相当独立，可以看成是一个个“迷你应用”，而 Intent 是这些“迷你应用”的粘合剂，Intent 不但可以用于同一个 Application 之间 Activities，Services 和 broadcast receivers 的交互，也可以用于不同 Application 之间 Activities，Services 和 broadcast receivers 的交互。 Intent 本身为一个数据载体，可以描述想要执行的操作以及用于这个操作的数据和其它属性。用个容易理解的概念，在访问网站时，我们需要提供网站的URL，有时还需要通过 URL 参数，在 Android 世界里，Intent 的功能类似于 URL，Android 操作系统根据 Intent 来触发对于的Activitives,Services 或是 Broadcast Receivers。

Android 应用中的三个核心组件：Activities，Services 和 broadcast receivers 都是通过Intent 来触发的，当它们触发的机制各不相同，而且不会有重叠，也就是说发给 Activity 的 Intent 不会激活 Service 或是 broadcast receivers，发给 broadcast receivers 的 Intent 也不会触发 Activity 和 Service。

1. Activity 通过方法 Context.startActivity() 和 Activity.startActivityForResult()来调用。以函数调用为参考 startActivity() 相当于调用无返回值的函数，startActivityForResult()调用的Activity有返回值，可以通过Activity.setResult()来返回结果。
2. Context.startService()用来初始化Service，Context.bindService()可以用来建立与目标Service之间的连接，如果Service没有运行，则会启动该Service。
3. Broadcast Receiver是通过 Context.sendBroadcast(), Context.sendOrderedBroadcast(), 和Context.sendStickyBroadcast()来触发的。大部分的Broadcast消息来自于Android操作系统，如电池状态，来电，短消息等。

和URL不太一样的是，URL 和网站一般是一一对应的，而一个 Intent 可以用来触发某个指定的Activity，Service 或是 Broadcast Receiver，或是触发多个满足 Intent 条件的 Activities，Services 或是 Broadcast Receivers。

下面来看看 Android OS 如何根据 Intent 来找到满足触发条件的 Activity，Service 或是Broadcast Receiver. 借用 SQL 数据库的概念可以更好的理解。

```
SELECT (Activitives|Services|Broadcast Receivers) AS Target
    FROM（List in AndroidManifest.xml)
    WHERE Intent Meet Target’s (Intent Filter)
```

意思就是从 AndroidManifest.xml 中定义的 Activities,Services 和 Broadcast Receiver 列表中查找符合 Intent 条件的 Activities，Services，或是 Broadcast Receivers。所有能活被激活的 Activity，Service 和 Broadcast Receiver 都必须在 AndroidManifest.xml 有定义，否则 Android OS 无法查询到该目标，相当于数据库中无记录，即使你在代码中定义了该Activity，Service 或 Broadcast Receiver。

Intent

Intent 本身为一个数据载体，可以描述想要执行的操作以及用于这个操作的数据和其它属性。它主要包含下列信息：

Component name：可以处理该 Intent 的组件名称，组件名称指定义 Activity，Service 的包和类的全名称。比如类名为 com.pstreets.gisengine.AndroidGISEngineTutorial,包名为 com.pstreets.gisengine。 组件名称为可选项，如果指定了，意味明确指定用来响应该 Intent 的 Activity，Service。

Action: 列出需要执行的操作名称。或者在 Broadcast Intents 的情况下给出发生的事件名称。

| Constant | Target component | Action |
|---|---|---|
| ACTION_CALL | activity | 开始打电话. |
| ACTION_EDIT | activity | 显示编辑对话框. |
| ACTION_MAIN | activity | 作为一个任务（应用）的起始 Activity，对于可以从 Android 应用列表的应用来说，都需要在 AndroidManifest.xml 中设置 ACTION_MAIN 的 Intent-Filter 属性。 |
| ACTION_SYNC | activity | 同步数据. |
| ACTION_BATTERY_LOW | broadcast receiver | 电池电量低告警. |
| ACTION_HEADSET_PLUG | broadcast receiver | 耳机插入或拔出. |
| ACTION_SCREEN_ON | broadcast receiver | 屏幕打开或关闭. |
| ACTION_TIMEZONE_CHANGED | broadcast receiver | 时区变动. |

Data：定义不数据的 URL 以及数据的 MIME 类型。不同的 Action 能够处理的 Data 类型也不一样，比如 ACTION_CALL,它处理的数据格式为 tel: URI，URI 为电话号码。 Category: 定义了可以响应 Intent 的附加信息，一个 Intent 可以指定多个 Category 类型。和 Action 类似，Android 预定义了一些 Category 类型：

| Constant | Meaning |
|---|---|
| CATEGORY_BROWSABLE | 表示目标 Activity 可以使用浏览器安全显示指定连接，比如说一个图片或是 Email 消息. |
| CATEGORY_GADGET | 表示该 Activity 可以当作一个 Gadget 嵌入到其它可以放置 Gadget 的 Activity 中。 |
| CATEGORY_HOME | 表示还 Activity 是 Home Screen，可以设置这个属性来替换 Android 自带的 Home Screen。 |

| Constant | Meaning |
| --- | --- |
| CATEGORY_LAUNCHER | 该 Activity 可以显示在 Android 程序管理器中。一般应用的主 Activity 都会在 AndroidManifest.xml 定义该属性。 |

Extra: 附加 Key-Value 列表，可以向目标 Activity 传送附加参数。可以理解成函数调用时的参数。
Flags：指出 Android 启动目标 Activity 时的一些选项（比如目标 Activity 隶属于那个应用等）。

除非是 Intent 明确指定目标（Explicitly）Activity 的类和包名称，这是 Activity 无需在 AndroidManifest.xml 定义 intent-filter，其它情况也叫隐含（Implicit）方式启动目标 Activity，在这种情况下 Android 操作系统查找目标 Activity，Service 或是 Broadcast Receiver 时主要根据 Intent 的 Action，Data 和 Category 属性来匹配定义在 AndroidManifest.xml 中 Activity，Service 或是 Broadcast Receiver 的 Intent Filters。

Intent Filters

Android 中没个有效的 Activity，Service，Broadcast Receiver 都必须在 AndroidManifest.xml 有对应的定义。除非只使用明确调用发生来启动目标 Activity，每个 Activity 都需要定义一个 intent-filter。下面是是明确指定目标Activity 的示例代码：

```
Intent intent=new Intent(SplashActivity.this,GNavigator.class);
startActivity(intent);
```

而更一般的情况，Activity 在 AndroidManifest.xml 具有如下定义：

```
<activity android:name=".AndroidGISEngineTutorial"
   android:label="@string/app_name">
 <intent-filter>
  <action android:name="android.intent.action.MAIN" />
  <action android:name="com.example.project.SHOW_CURRENT" />
  <action android:name="com.example.project.SHOW_RECENT" />
  <action android:name="com.example.project.SHOW_PENDING" />

  ….
  <category android:name="android.intent.category.DEFAULT" />
  <category android:name="android.intent.category.BROWSABLE" />
  …
   <data android:mimeType="video/mpeg" android:scheme="http" . . . />
   <data android:mimeType="audio/mpeg" android:scheme="http" . . . />
   ….
 </intent-filter>
</activity>
```

Activity 的 intent-filter 可以包含 action，category，data 子元素，给出了该 Activity 能够处理的 Intent 的Action，Category 和数据类型。Android 操作系统就是根据 Activity 的 intent-filter 来匹配 Intent，从而触发目标Activity，或是 Service，Broadcast Receiver。

最常见的一个 Intent Filter 组合如下：

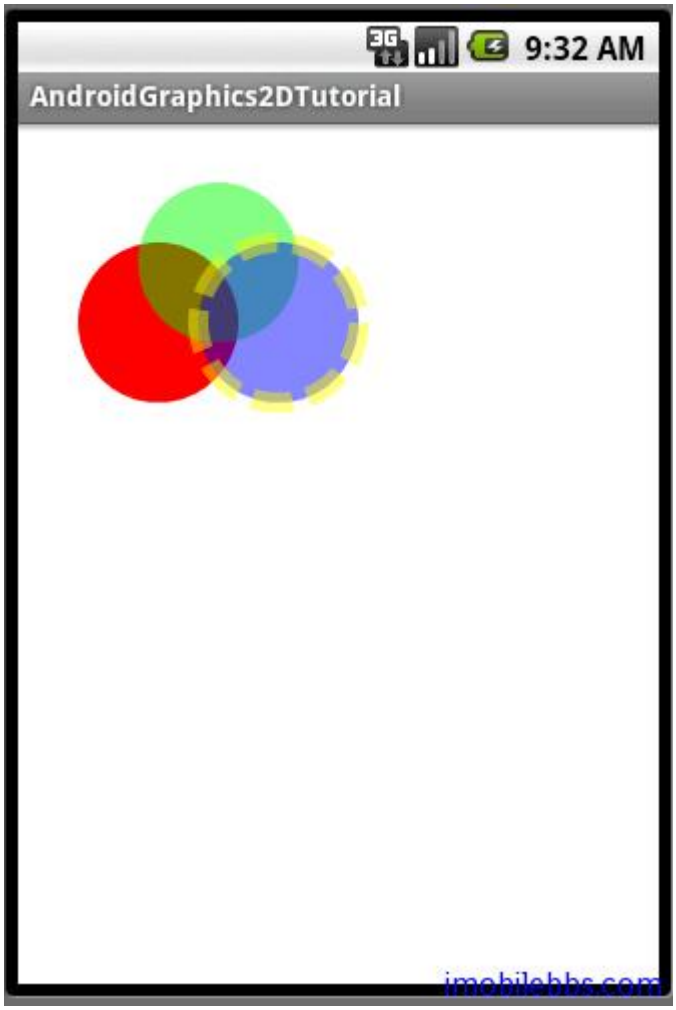
```
<intent-filter . . . >
  <action android:name=”code android.intent.action.MAIN” />
  <category android:name=”code android.intent.category.LAUNCHER” />
 </intent-filter>
```

表示用户可以从 Android 设备的应用程序管理器启动该 Activity，这个 Activity 为应用的主 Activity，主 Activity 可以再使用 Intent 触发或是启动其它 Activity。

# 8

# 引路蜂二维图形绘制实例功能定义

有了前面对 Android 平台的介绍，基本上可以开始编写 Android 应用了，这里将以绘制二维图形为例，对 Android 开发的一般方法做过介绍，其中涉及到自定义 Application 类，扩展 View，Intent 定义，发送消息，Data Binding（Adapter），和基本 UI 设计。示例没有使用 Android 平台自带的二维图形 API，而是调用了引路蜂二维图形库，引路蜂二维图形库 Graphics 2D API 实现了移动平台上图形引擎,它能够以一种统一的方式处理各种基本图形(Shape),路径(Path),文本(Texts),适量字体及图像。基本类定义类同 Windows GDI+库。 所有示例和 Silverlight 二维图形库类似 Silverlight 引路蜂二维图形库下载 实例将提供源码，内含引路蜂二维图形库（免费使用）。




二维图形按功能分成下表所示：

| 功能 | 示例 |
| --- | --- |
| Color | Colors |
| Brush | Pattern ， Gradients |
| Pen | Lines, Dashes, LineCap,LineJoin |
| Path | Polys, Paths |
| Shape | Oval ,Pear ,Shape2DDemo |
| Image | DrawMap, JumbleImage, SeeThroughImage |

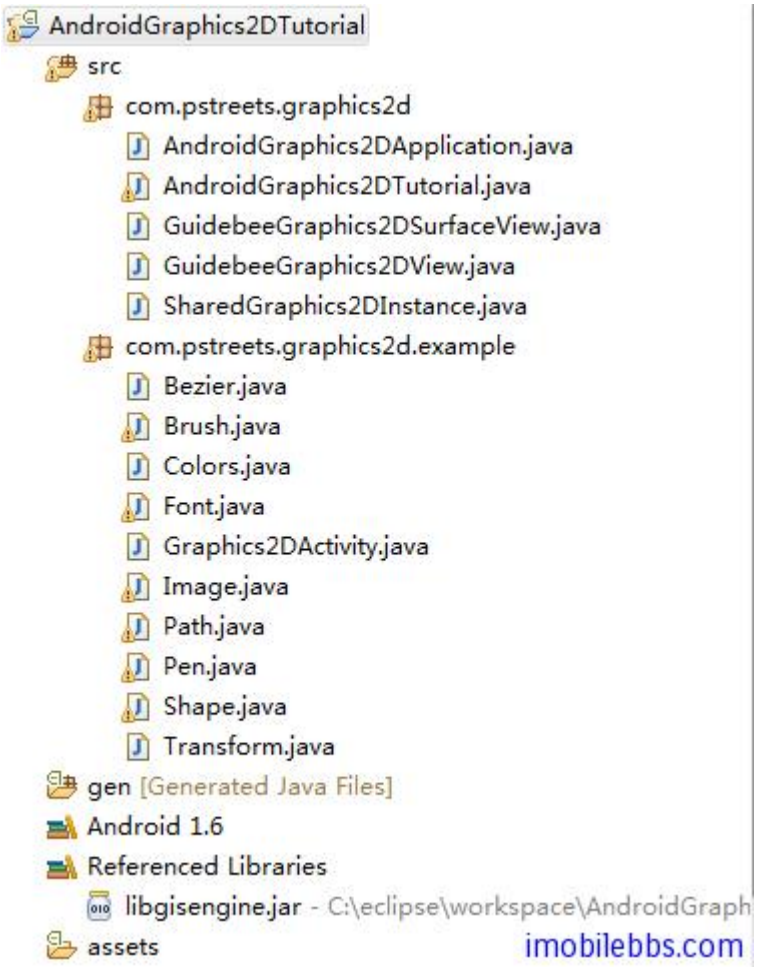
| 功能 | 示例 |
| --- | --- |
| Font | FontDemo, FontTypes |
| Transform | Transform |
| Dynamic Shape | Bezier |

每个功能设计成一个 Activity，包含在 AndroidGraphics2DApplication 中。

9

# 创建应用程序框架

Android 简明开发教程八说明了程序需要实现的功能，就可以创建 Android 项目了。请参见 Android简明开发教程三：第一个应用 Hello World ，创建一个新项目 AndroidGraphics2DTutorial。今天先介绍创建的程序的框架。然后再项目添加如下类定义：



## 添加第三方库文件

AndroidGraphics2DTutorial 调用了引路蜂二维图形库，因此需要在项目中添加第三方库引用(libgisengine.jar),打开 Android 属性窗口，添加 External JARs。把 libgisengine.jar 添加到项目中，引路蜂二维图形库是引路蜂地图开发包的一部分。添加库引用可以参见 Android引路蜂地图开发示例：基本知识。

类说明，下表列出了项目中定义的类的简要说明：

| 类 | 说明 |
| --- | --- |
| AndroidGraphics2DApplication | 应用程序类，为Application子类 |
| AndroidGraphics2DTutorial | 主Activity，为ListActivity子类，用于列出其它示例。 |
| GuidebeeGraphics2DSurfaceView | SurfaceView子类用于显示图形 |
| GuidebeeGraphics2DView | View子类用于显示图形，与GuidebeeGraphics2DSurfaceView 功能一样，在程序中可以互换。 |
| SharedGraphics2DInstance | 定义了共享类对象，主要包含Graphics2D |
| Graphics2DActivity | Activity子类，为所有示例基类，定义一些所有示例共享的类变量和函数。 |
| Bezier，Brush，Colors，Font，Image，Path，Pen，Shape，Transform | 为Graphics2DActivity的子类，为二维图形演示各个功能 |

AndroidGraphics2DApplication ，其实在一般的 Android 应用中，无需定义 Application 的派生类，比如在 Hello World 中就没有定义，当是如果想在多个 Activity 中共享变量，或是想初始化一些全局变量，可以定义 Application 的派生类，然后可以在 Activity 或 Service 中调用 getApplication() 或 getApplicationContext()来取得 Application 对象，可以访问定义在Application 中的一些共享变量。在这个例子中 AndroidGraphics2DApplication 严格些也可不定义，为了说明问题，还是定义了用来初始化 Graphics2D 实例，Graphics2D 实例可以被所有示例Activity，如 Colors，Font 访问。如果定义了 Application 的派生类，就需要在AndroidManifest.xml 中说明 Application 派生类的位置。

```
<manifest xmlns:android=”http://schemas.android.com/apk/res/android”
    package=”com.pstreets.graphics2d”
    android:versionCode=”1″
    android:versionName=”1.0″ >
  <application android:name=”AndroidGraphics2DApplication”
     android:icon=”@drawable/icon” android:label=”@string/app_name” >
    <activity android:name=”.AndroidGraphics2DTutorial”
          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>
   <uses-sdk android:minSdkVersion=”4” />

</manifest>
```

Application 可以重载 onCreate()和 onTerminate() ，onCreate()在应用启动时执行一次，onTerminate()在应用推出执行一次。AndroidGraphics2DApplication 的 onCreate() 中初始化Graphics2D 实例：

```
public void onCreate() {
  SharedGraphics2DInstance.graphics2d=
     new Graphics2D(SharedGraphics2DInstance.CANVAS_WIDTH,
       SharedGraphics2DInstance.CANVAS_HEIGHT);
 }
```

```
AndroidGraphics2DTutorial 为ListActivity 子类，直接从 AndroidManifest.xml 中读取Intent-Filter Catetory 为 com.pstreets.g

private static final String SAMPLE_CATEGORY="com.pstreets.graphics2d.SAMPLE_CODE";

Intent mainIntent = new Intent(Intent.ACTION_MAIN, null);
mainIntent.addCategory(SAMPLE_CATEGORY);
...
```

AndroidGraphics2DTutorial 为ListActivity子类，直接从 AndroidManifest.xml 中读取Intent-Filter Catetor y 为 com.pstreets.graphics2d.SAMPLE_CODE 的所有 Activity。

```
private static final String SAMPLE_CATEGORY="com.pstreets.graphics2d.SAMPLE_CODE";

Intent mainIntent = new Intent(Intent.ACTION_MAIN, null);
mainIntent.addCategory(SAMPLE_CATEGORY);
...
```

GuidebeeGraphics2DSurfaceView 和 GuidebeeGraphics2DView 分别为 SurfaceView 和 View的子类，都可以用来显示图形结果。在程序中可以互换。

```
package com.pstreets.graphics2d;

import android.content.Context;
import android.graphics.Canvas;
import android.util.AttributeSet;
import android.view.View;

public class GuidebeeGraphics2DView extends View {

 public GuidebeeGraphics2DView(Context context, AttributeSet attrs,
```

```
  int defStyle) {
 super(context, attrs, defStyle);

}

public GuidebeeGraphics2DView(Context context, AttributeSet attrs) {
 super(context, attrs);

}

public GuidebeeGraphics2DView(Context context) {
 super(context);

}

public void onDraw(Canvas canvas) {
 super.onDraw(canvas);
 canvas.drawColor(0xFFFFFFFF);
 if (SharedGraphics2DInstance.graphics2d != null) {
  int offsetX = (getWidth() -
    SharedGraphics2DInstance.CANVAS_WIDTH) / 2;
  int offsetY = (getHeight()
    - SharedGraphics2DInstance.CANVAS_HEIGHT) / 2;
  canvas.drawBitmap(SharedGraphics2DInstance.graphics2d.getRGB(), 0,
    SharedGraphics2DInstance.CANVAS_WIDTH,
    offsetX, offsetY,
    SharedGraphics2DInstance.CANVAS_WIDTH,
    SharedGraphics2DInstance.CANVAS_HEIGHT,
    true, null);
 }
}

}
```

```
package com.pstreets.graphics2d;

import android.content.Context;
import android.graphics.Canvas;
import android.util.AttributeSet;
import android.view.SurfaceHolder;
import android.view.SurfaceView;

public class GuidebeeGraphics2DSurfaceView extends
  SurfaceView implements SurfaceHolder.Callback {

 SurfaceHolder holder;
```

```
private void initHolder() {
 holder = this.getHolder();
 holder.addCallback(this);
}

public GuidebeeGraphics2DSurfaceView(Context context,
  AttributeSet attrs,
  int defStyle) {
 super(context, attrs, defStyle);
 initHolder();

}

public GuidebeeGraphics2DSurfaceView(Context context,
  AttributeSet attrs) {
 super(context, attrs);
 initHolder();

}

public GuidebeeGraphics2DSurfaceView(Context context) {
 super(context);
 initHolder();

}

@Override
public void surfaceChanged(SurfaceHolder arg0,
  int arg1, int arg2, int arg3) {
 // TODO Auto-generated method stub

}

@Override
public void surfaceCreated(SurfaceHolder arg0) {
 new Thread(new MyThread()).start();

}

@Override
public void surfaceDestroyed(SurfaceHolder arg0) {
 // TODO Auto-generated method stub

}
```

```
 class MyThread implements Runnable {

  @Override
  public void run() {
   Canvas canvas = holder.lockCanvas(null);
   canvas.drawColor(0xFFFFFFFF);
   if (SharedGraphics2DInstance.graphics2d != null) {
    int offsetX = (getWidth() -
      SharedGraphics2DInstance.CANVAS_WIDTH) / 2;
    int offsetY = (getHeight() -
      SharedGraphics2DInstance.CANVAS_HEIGHT) / 2;
    canvas.drawBitmap
      (SharedGraphics2DInstance.graphics2d.getRGB(),
      0, SharedGraphics2DInstance.CANVAS_WIDTH,
      offsetX,
      offsetY,
      SharedGraphics2DInstance.CANVAS_WIDTH,
      SharedGraphics2DInstance.CANVAS_HEIGHT,
      true, null);
   }
   holder.unlockCanvasAndPost(canvas);

  }

 }

}
```

SurfaceView 动态显示性能比较好，一般用在游戏画面的显示。图形的绘制可以在单独的线程中完成。

修改 res\layout\main.xml

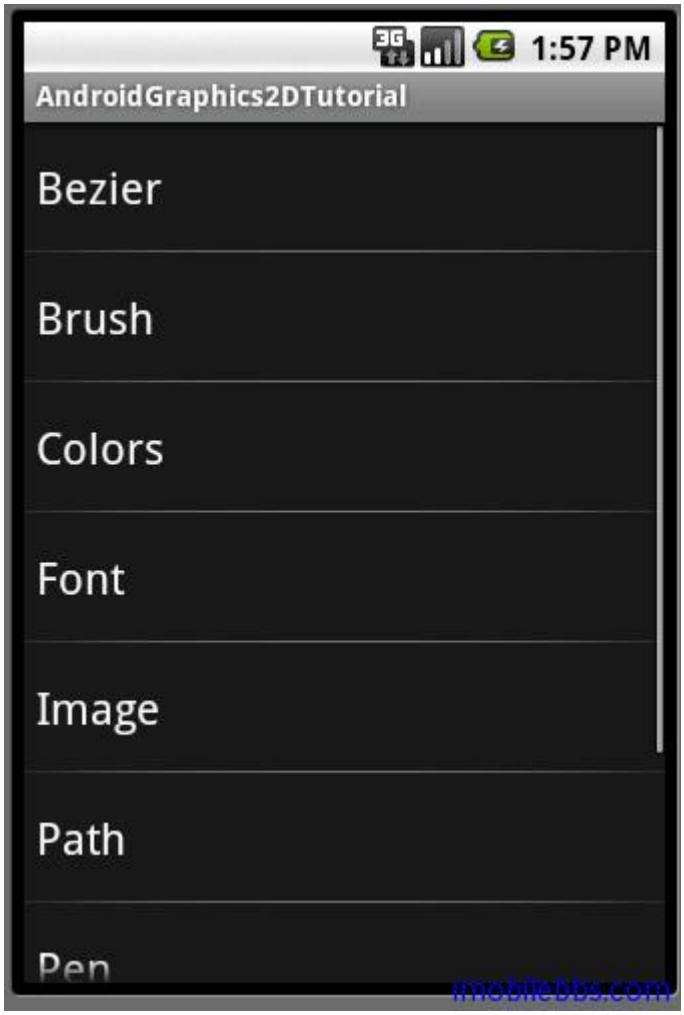
```
<?xml version=”1.0″ encoding=”utf-8″?>
<LinearLayout xmlns:android=”http://schemas.android.com/apk/res/android”
   android:orientation=”vertical”
   android:layout_width=”fill_parent”
   android:layout_height=”fill_parent”
   >
<com.pstreets.graphics2d.GuidebeeGraphics2DSurfaceView
    android:id=”@+id/graphics2dview”

    android:layout_width=”fill_parent”
    android:layout_height=”fill_parent” />
</LinearLayout>
```

如果使用 GuidebeeGraphics2DView 作为显示，则只需将上面红色部分该成GuidebeeGraphics2DView 即可。

为了能在 AndroidGraphics2DTutorial 列表中列出，对项目中的示例 Activity 的都定义下列intent-filter

```
<activity android:name=”.example.Colors” android:label=”@string/activity_colors”>
        <intent-filter>
          <action android:name=”android.intent.action.MAIN” />
          <category android:name=”com.pstreets.graphics2d.SAMPLE_CODE” />
        </intent-filter>
      </activity>
```

这样就完成了程序框架的设计，起始界面如下：




Tags: Android

10

# 数据绑定 Data Binding

前面提到 AndroidGraphics2DTutorial 说过它是 ListActivity 派生出来的。ListActivity 中显示的是 ListVie w，ListView 和 Gallery ，Spinner 有一个共同点：它们都是 AdapterView 的子类。AdapterView 的显示可以通过数据绑定来实现，数据源可以是数组或是数据库记录，数据源和AdapterView 是通过 Adapter 作为桥梁。通过 Adapter，AdatperView 可以显示数据源或处理用户选取时间，如：选择列表中某项。


AndroidGraphics2DTutorial 读取 AndroidManifest.xml中Intent-Filter为

的所有 Activity，以列表方式显示。使用了 Android API 自带的 SimpleAdapter。 来看看AndroidGraphics2D Tutorial.java 中相关代码：

```
public class AndroidGraphics2DTutorial extends ListActivity {

 private static final String SAMPLE_CATEGORY
        ="com.pstreets.graphics2d.SAMPLE_CODE";

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

  setListAdapter(new SimpleAdapter(this, getData(),
    android.R.layout.simple_list_item_1, new String[] { "title" },
    new int[] { android.R.id.text1 }));
  getListView().setTextFilterEnabled(true);

 }

 protected List getData() {
  List<Map> myData = new ArrayList<Map>();

  Intent mainIntent = new Intent(Intent.ACTION_MAIN, null);
  mainIntent.addCategory(SAMPLE_CATEGORY);

  PackageManager pm = getPackageManager();
  List<ResolveInfo> list = pm.queryIntentActivities(mainIntent, 0);

  if (null == list)
   return myData;

  String[] prefixPath;
```

```
 prefixPath = null;

 int len = list.size();

 Map<String, Boolean> entries = new HashMap<String, Boolean>();

 for (int i = 0; i < len; i++) {
  ResolveInfo info = list.get(i);
  CharSequence labelSeq = info.loadLabel(pm);
  String label = labelSeq != null ? labelSeq.toString()
    : info.activityInfo.name;

  String[] labelPath = label.split("/");

  String nextLabel = prefixPath == null ? labelPath[0]
    : labelPath[prefixPath.length];

  if ((prefixPath != null ? prefixPath.length : 0)
== labelPath.length - 1) {
   addItem(myData,
     nextLabel,
     activityIntent(
       info.activityInfo.applicationInfo.packageName,
       info.activityInfo.name));
  } else {
   if (entries.get(nextLabel) == null) {
    addItem(myData, nextLabel, browseIntent(nextLabel));
    entries.put(nextLabel, true);
   }
  }

 }

 Collections.sort(myData, sDisplayNameComparator);

 return myData;
}

 private final static Comparator<Map> sDisplayNameComparator
= new Comparator<Map>() {
 private final Collator collator = Collator.getInstance();

 public int compare(Map map1, Map map2) {
  return collator.compare(map1.get("title"), map2.get("title"));
 }
```

```
};

protected Intent activityIntent(String pkg, String componentName) {
 Intent result = new Intent();
 result.setClassName(pkg, componentName);
 return result;
}

protected Intent browseIntent(String path) {
 Intent result = new Intent();
 result.setClass(this, AndroidGraphics2DTutorial.class);
 return result;
}

protected void addItem(List<Map> data, String name, Intent intent) {
 Map<String, Object> temp = new HashMap<String, Object>();
 temp.put("title", name);
 temp.put("intent", intent);
 data.add(temp);
}

@Override
protected void onListItemClick(ListView l, View v,
   int position, long id) {
 Map map = (Map) l.getItemAtPosition(position);
 Intent intent = (Intent) map.get("intent");
 startActivity(intent);
}
}
```

使用数据显示 Layout，上面代码中

setListAdapter(new SimpleAdapter(this, getData(), android.R.layout.simple_list_item_1, new String[] { “title” }, new int[] { android.R.id.text1 }));

为ListActivity 中 ListView 指定 Adapter，这个 Adapter 的数据源为 getData()，getData()从Manifest.xml 中查找出所有符合条件的示例 Activity 列表。 这里 DataSource 是静态的从文件中读取，如果 DataSource 为数组或是其它数据源，如果程序中修改数值的内容，则你应该notifyDataSetChanged()来通知 UI 数据有变动。UI则会刷新显示以反映数据变化。简单的说Android 数据绑定和 .Net WinForm ,WPF 中数据绑定类似。

处理用户选取事件，AdapterView.OnItemClickListener()可以用来处理选取事件，对于ListActivity，可以用 protected void onListItemClick(ListView l, View v, int position, long id)。AndroidGraphics2DTutorial中的实现是用户选取 Activity 名称好，则启动对应的 Activity。

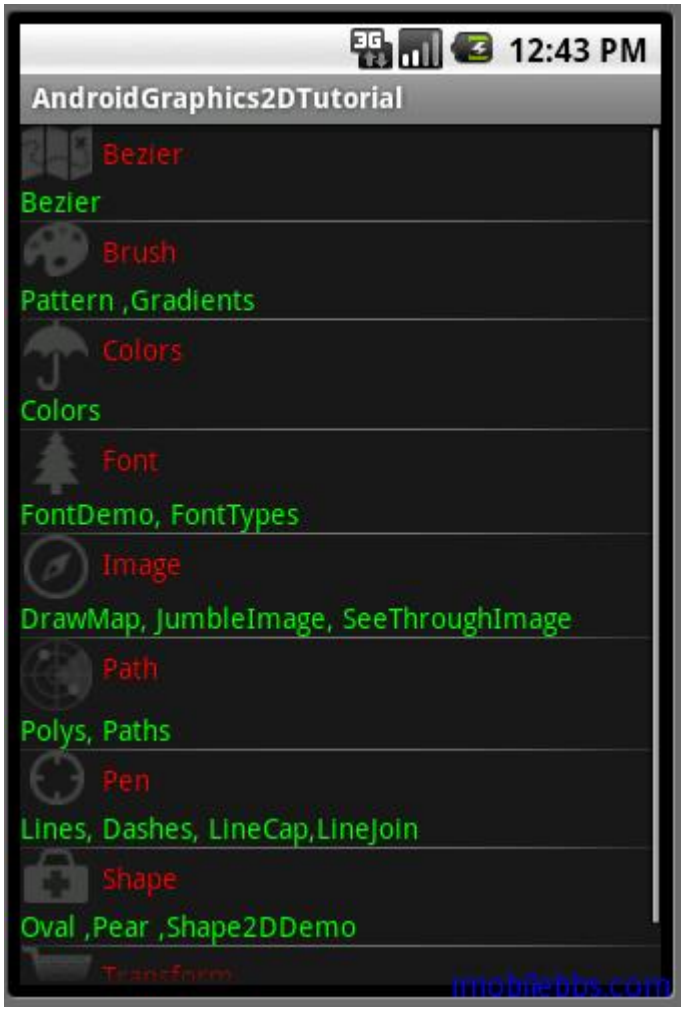
上面代码中使用 SimpleAdapter，并使用 Android 提供的android.R.layout.simple_list_item_1来显示数据，Andrid 也允许使用自定义的 Layout 来显示数据，对这个例子来说，可以使用图片加说明来显示列表，将在后面介绍如果使用自定义 Adapter 和自定义 Layout 来显示绑定的数据。

Tags: Android

11

# 自定义 Adapter 显示列表

在介绍数据绑定时，我们使用了系统自带的 SimpleAdapter。Android 允许自定义 Adapter ，理论上可以使用任意的 View（Layout）来显示数据。下图是对 AndroidGraphics2DTutorial 做改动，使用自定义 Adapter 来显示示例 Activity 列表。




在例子中我们把原来的 AndroidGraphics2DTutorial 改名为 AndroidGraphics2DTutorial1，重新创建一个类 AndroidGraphics2DTutorial 来显示示例列表。打算使用三个 View 来显示列表中的一项，一个图标（例子中随机使用了一些图标），一个文本框显示示例 Activity 名称，另一个文本框显示示例的具体信息。在 res\layout 目录下创建一个 activitylist.xml。内容如下：


这次我们不从 AndroidManifest.xml 中读取 Activity 列表，而是使用 String Array 资源。在res\value\string.xml 中添加下列 Array 资源：

```
<string-array name=”activity_name”>
 <item>Bezier</item>
 <item>Brush</item>
 <item>Colors</item>
 <item>Font</item>
 <item>Image</item>
 <item>Path</item>
 <item>Pen</item>
 <item>Shape</item>
 <item>Transform</item>
</string-array>

<string-array name=”activity_info”>
 <item>Bezier</item>
 <item>Pattern ,Gradients</item>
 <item>Colors</item>
 <item>FontDemo, FontTypes</item>
 <item>DrawMap, JumbleImage, SeeThroughImage</item>
 <item>Polys, Paths</item>
 <item>Lines, Dashes, LineCap,LineJoin</item>
 <item>Oval ,Pear ,Shape2DDemo</item>
 <item>Transform</item>
</string-array>
```

定义了这些资源后，可以在程序中使用自定义 Adapter 来显示列表：

```
class ActivityInfo{
 int iconIndex;
 String activityName;
 String activityInfo;
}

class ActivityInfoAdapter extends ArrayAdapter<ActivityInfo>{

 int resource;
 public ActivityInfoAdapter(Context context, int resourceId,
   List<ActivityInfo> objects) {
  super(context, resourceId, objects);
  resource=resourceId;
 }

 @Override
 public View getView(int position,View ConvertView,ViewGroup parent){
  LinearLayout activityInfoView;
  ActivityInfo activityInfo=getItem(position);
  String activity_Name=activityInfo.activityName;
  String activity_Info=activityInfo.activityInfo;
  int iconIndex=activityInfo.iconIndex;
  if(ConvertView==null){
   activityInfoView=new LinearLayout(getContext());
   String inflater=Context.LAYOUT_INFLATER_SERVICE;
   LayoutInflater vi;
   vi=(LayoutInflater)getContext().getSystemService(inflater);
   vi.inflate(resource, activityInfoView,true);
  }else{
   activityInfoView=(LinearLayout)ConvertView;
  }
  TextView activity_NameView
    =(TextView)activityInfoView.findViewById(R.id.activityName);
  TextView activity_InfoView
     =(TextView)activityInfoView.findViewById(R.id.activityInfo);
  ImageView iconView
     =(ImageView)activityInfoView.findViewById(R.id.iconImage);
  activity_NameView.setText(activity_Name);
  activity_InfoView.setText(activity_Info);
  iconView.setImageResource(iconIndex);
  return activityInfoView;

 }
```

```
}

public class AndroidGraphics2DTutorial extends ListActivity {

 private ArrayList<ActivityInfo> activityInfos
    =new ArrayList<ActivityInfo>();
 private ActivityInfoAdapter aa;
 private final static String packgeName="com.pstreets.graphics2d";

 @Override
 public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  Resources res = getResources();
  String[] activity_Names = res.getStringArray(R.array.activity_name);
  String[] activity_Infos = res.getStringArray(R.array.activity_info);
  for(int i=0;i<activity_Names.length;i++){
   ActivityInfo activityInfo=new ActivityInfo();
   activityInfo.activityName=activity_Names[i];
   activityInfo.activityInfo=activity_Infos[i];
   activityInfo.iconIndex=R.drawable.icon1+i;
   activityInfos.add(activityInfo);
  }

  aa=new ActivityInfoAdapter(this,R.layout.activitylist,activityInfos);
  setListAdapter(aa);
 }

 @Override
 protected void onListItemClick(ListView l, View v, int position, long id) {
  ActivityInfo activityInfo = (ActivityInfo) l.getItemAtPosition(position);
  Intent intent = new Intent();
  intent.setClassName(this,
      packgeName+".example." +activityInfo.activityName);
  startActivity(intent);
 }
}
```

类 ActivityInfo 定义列表每个元素的 Data Model，为 Activity 的 Icon 资源 ID，Activity Name以及Activity Info.

类 ActivityInfoAdapter 为自定义 Adapter，关键的是 public View getView(int position,View ConvertView,ViewGroup parent)。这个函数返回用来显示没个类别元素的 View的示例。例子中为 activitylist.xml 对应的 Layout。

这个例子使用的列表框，自定义 Adapter 适用所有 AdapterView，如 Spinner，Gallery 等。

Tags: Android

12

# 引路蜂二维图形库简介及颜色示例

AndroidGraphics2DTutorial 定义了应用的主 Activity，下面就可以开始写每个具体的二维绘图示例。不同的例子将尽量采用不同的 UI 控件：Menu，Content Menu，Dialog，Custom Dialog，Button 等等。例子采用了引路蜂二维图形库，引路蜂二维图形库 Graphics 2D API 实现了移动平台(Java ME,Blackberry,iPhone,Android,Windows Phone)上图形引擎,它能够以一种统一的方式处理各种基本图形(Shape),路径(Path),文本(Texts),适量字体及图像。 简单的说来，Graphics 2D API 实现了与之对应的 Java SE 上类似的二维图形库 API。

主要功能如下： – 支持各种基本图形：曲线，矩形，椭圆等； – 支持绘制任意几何图形 – 支持在图形，文体，图象上的碰撞检测 – 增强的颜色扶持及颜色管理 – 控制图形绘制的质量 – 填充，外框，各种线条绘制 – 二维图形变换 – 矢量字体 – 从左到右，从右到左，从上到下显示文体 – 反走样 – 透明度支持 – 图标，及图象绘制

详细的内容可以参见 Silverlight 引路蜂二维图形库示例

我们在 Android简明开发教程九：创建应用程序框架中定义了一个基类 Graphics2DActivity 作为所有示例Activity的父类：

```
public abstract class Graphics2DActivity extends Activity{

 protected Graphics2D graphics2D
   =SharedGraphics2DInstance.graphics2d;

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

 protected abstract void drawImage();

 public void onStart() {
  super.onStart();
  drawImage();

 }
}
```

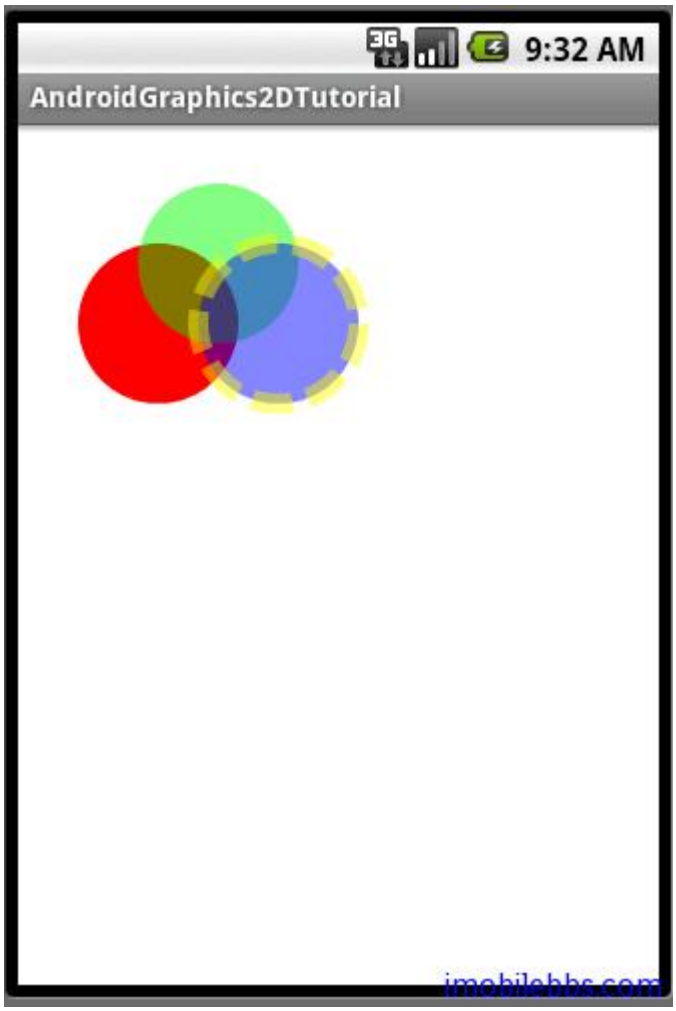
其中 graphics2D 为图形画板对象(Canvas)是以 width x height 的二维整型数组来表示的。这个数组的每个值为一个32为整数。格式为 ARGB，分别代表透明度，红色，绿色，蓝色。在画板上的绘制操作（点，线，多边形，填充等）是修改这些颜色值。

R.layout.main 中可以使用 GuidebeeGraphics2DSurfaceView 或是 GuidebeeGraphics2DView 来作为画板的显示结果。

抽象方法 protected abstract void drawImage()；用来绘制不同的内容。

修改 com.pstreets.graphics2d.example.Colors 来使用引路蜂二维图形库绘制不同的颜色，如果以前用过 Java SE或是 .Net Framework,你会觉得引路蜂二维图形库提供的API和它们非常相似，代码很好理解。

```
public class Colors extends Graphics2DActivity{

 protected void drawImage(){

    /**
    * The solid (full opaque) red color in the ARGB space
    */
   Color redColor    = new Color(0xffff0000);

   /**
    * The semi-opaque green color in the ARGB space (alpha is 0x78)
    */
   Color greenColor = new Color(0x7800ff00,true);

   /**
    * The semi-opaque blue color in the ARGB space (alpha is 0x78)
    */
   Color blueColor = new Color(0x780000ff,true);
   /**
    * The semi-opaque yellow color in the ARGB space ( alpha is 0x78)
    */
   Color yellowColor = new Color(0x78ffff00,true);

   /**
    * The dash array
    */
   int dashArray[] = { 20 ,8 };
   graphics2D.clear(Color.WHITE);
   SolidBrush brush=new SolidBrush(redColor);
   graphics2D.fillOval(brush,30,60,80,80);
   brush=new SolidBrush(greenColor);
   graphics2D.fillOval(brush,60,30,80,80);
   Pen pen=new Pen(yellowColor,10,Pen.CAP_BUTT,Pen.JOIN_MITER,dashArray,0);
   brush=new SolidBrush(blueColor);
   graphics2D.setPenAndBrush(pen,brush);
   graphics2D.fillOval(null,90,60,80,80);
   graphics2D.drawOval(null,90,60,80,80);

 }

}
```




Colors Activity 非常简单，除 View 之外，没有其它 UI。按“Back”后可以退回示例列表显示UI。

Tags: Android

13

# Option Menu 画笔示例

引路蜂二维图形画笔（Pen）示例含有四个示例，Lines ，Dashes ,LineJoin 和 LineCap。打算采用Option Menu（主菜单）的方式来选择不同示例。

首先要对 GuidebeeGraphics2DView，和 Graphics2DActivity做些改动，从这个示例开始，GuidebeeGraphics2DView 需要动态绘制不同图形（可以通过菜单，或是 Thread）。在GuidebeeGraphics2DView 增加下面两个方法：

```
final Runnable updateCanvas = new Runnable() {
 public void run() {
  invalidate();
 }
};

public void refreshCanvas(){
 post(updateCanvas);
}
```

post 可以用在非 UI Thread 中 Call UI Thread 中方法。这里只是触发屏幕重绘事件以刷新屏幕显示。

在 Graphics2DActivity 增加一个变量 protected GuidebeeGraphics2DView graphic2dView; 来获得对应的 GuidebeeGraphics2DView 实例。

graphic2dView=(GuidebeeGraphics2DView)findViewById(R.id.graphics2dview);

使用 Option Menu，尽管也可以完全使用代码来创建菜单，更一般的方法是使用菜单资源。在 res 下创建 menu 子目录，然后在 res\menu 下创建 menu_option_line.xml 用来显示 Pen 的四个示例选项：

```
<?xml version=”1.0″ encoding=”utf-8″?>
<menu
 xmlns:android=”[http://schemas.android.com/apk/res/android](http://schemas.android.com/apk/res/android)“>
<item
 android:id=”@+id/mnuLines”
 android:title=”Lines”
 android:icon=”@drawable/icon1″ >
</item>
<item
 android:id=”@+id/mnuDashes”
 android:title=”Dashes”
 android:icon=”@drawable/icon2″ >
</item>
<item
 android:id=”@+id/mnuLineCap”
 android:title=”LineCap”
 android:icon=”@drawable/icon3″ >
```

```
</item>
<item
 android:id=”@+id/mnuLineJoin”
 android:title=”LineJoin”
 android:icon=”@drawable/icon4″ >
</item>
</menu>
```

定义菜单，定义菜单项，可以嵌套以定义子菜单。菜单可以定义 id, Icon,Text 等属性。也可以支持单选，多选，此时就需要借助,使用 group 可以把一最菜单项定义为一个组，可以使用 setGroupVisible()来显示隐藏整个菜单组，Enable 或是 Disable 整个菜单组 setGroupEnabled()等。最关键的，有了组才能实现菜单的单选和多选功能：

```
<?xml version=”1.0″ encoding=”utf-8″?>
<menu xmlns:android=”[http://schemas.android.com/apk/res/android](http://schemas.android.com/apk/res/android)“>
 <item android:id=”@+id/item1″
 android:icon=”@drawable/item1″
 android:title=”@string/item1″ />
 <!– menu group –>
 <group android:id=”@+id/group1″ >
  <item android:id=”@+id/groupItem1″
  android:title=”@string/groupItem1″ />
  <item android:id=”@+id/groupItem2″
  android:title=”@string/groupItem2″ />
 </group>
</menu>
```

定义好菜单资源后，就可以使用 MenuInflater.inflate()展开菜单，一般需要在 Activity 的onCreateOptionsMenu()展开菜单：

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

最后是响应菜单事件：

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
 menuOption = item.getItemId();
 drawImage();
 return true;
}
```

item.getItemId(); 返回菜单的 ID（在菜单资源中定义）。


Line Join


Line Cap



完整代码如下：

```
public class Pen extends Graphics2DActivity {

 int menuOption;

 @Override
 protected void drawImage() {
  switch (menuOption) {
  case R.id.mnuLines:
   drawLines();
   break;

  case R.id.mnuDashes:
   drawDash();
   break;
```

```
 case R.id.mnuLineCap:
  drawLineCap();
  break;

 case R.id.mnuLineJoin:
  drawLineJoin();
  break;
 default:
  drawLines();
  break;

 }
 graphic2dView.refreshCanvas();

}

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

@Override
public boolean onOptionsItemSelected(MenuItem item) {
 menuOption = item.getItemId();
 drawImage();
 return true;
}

private void drawLineJoin() {
 Color blackColor = new Color(0xff000000);
 Color whiteColor = new Color(0xffffffff);

 com.mapdigit.drawing.geometry.Path path
   = new com.mapdigit.drawing.geometry.Path();
 path.moveTo(40, 60);
 path.lineTo(90, 20);
 path.lineTo(140, 60);
 // Clear the canvas with white color.
 graphics2D.clear(Color.WHITE);

 AffineTransform matrix = new AffineTransform();
 graphics2D.setAffineTransform(matrix);
 com.mapdigit.drawing.Pen pen
```

```
    = new com.mapdigit.drawing.Pen(blackColor,
   20, com.mapdigit.drawing.Pen.CAP_BUTT,
   com.mapdigit.drawing.Pen.JOIN_MITER);
  graphics2D.draw(pen, path);
  pen = new com.mapdigit.drawing.Pen(whiteColor, 1);
  graphics2D.draw(pen, path);

  matrix.translate(0, 50);
  graphics2D.setAffineTransform(matrix);

  pen = new com.mapdigit.drawing.Pen(blackColor, 20,
    com.mapdigit.drawing.Pen.CAP_BUTT,
    com.mapdigit.drawing.Pen.JOIN_ROUND);
  graphics2D.draw(pen, path);
  pen = new com.mapdigit.drawing.Pen(whiteColor, 1);
  graphics2D.draw(pen, path);

  matrix = new AffineTransform();
  matrix.translate(0, 100);
  graphics2D.setAffineTransform(matrix);

  pen = new com.mapdigit.drawing.Pen(blackColor, 20,
    com.mapdigit.drawing.Pen.CAP_BUTT,
    com.mapdigit.drawing.Pen.JOIN_BEVEL);
  graphics2D.draw(pen, path);
  pen = new com.mapdigit.drawing.Pen(whiteColor, 1);
  graphics2D.draw(pen, path);

 }

 private void drawLineCap() {
  Color blackColor = new Color(0xff000000);
  Color whiteColor = new Color(0xffffffff);
  // Clear the canvas with white color.
  graphics2D.clear(Color.WHITE);
  AffineTransform matrix = new AffineTransform();
  graphics2D.setAffineTransform(matrix);

  com.mapdigit.drawing.Pen pen
     = new com.mapdigit.drawing.Pen(blackColor,
   20, com.mapdigit.drawing.Pen.CAP_BUTT,
   com.mapdigit.drawing.Pen.JOIN_MITER);
  graphics2D.drawLine(pen, 40, 60, 140, 60);
  pen = new com.mapdigit.drawing.Pen(whiteColor, 1);
  graphics2D.drawLine(pen, 40, 60, 140, 60);
```

```
 pen = new com.mapdigit.drawing.Pen(blackColor, 20,
   com.mapdigit.drawing.Pen.CAP_ROUND,
   com.mapdigit.drawing.Pen.JOIN_MITER);
 graphics2D.drawLine(pen, 40, 100, 140, 100);
 pen = new com.mapdigit.drawing.Pen(whiteColor, 1);
 graphics2D.drawLine(pen, 40, 100, 140, 100);

 pen = new com.mapdigit.drawing.Pen(blackColor, 20,
   com.mapdigit.drawing.Pen.CAP_SQUARE,
   com.mapdigit.drawing.Pen.JOIN_MITER);
 graphics2D.drawLine(pen, 40, 140, 140, 140);
 pen = new com.mapdigit.drawing.Pen(whiteColor, 1);
 graphics2D.drawLine(pen, 40, 140, 140, 140);
}

private void drawLines() {
 Color greenColor = new Color(0xff00ff00);
 // Clear the canvas with white color.
 graphics2D.clear(Color.WHITE);
 AffineTransform matrix = new AffineTransform();
 graphics2D.setAffineTransform(matrix);

 com.mapdigit.drawing.Pen pen
   = new com.mapdigit.drawing.Pen(greenColor,1);
 graphics2D.drawLine(pen, 20, 150, 60, 50);

 pen = new com.mapdigit.drawing.Pen(greenColor, 2);
 graphics2D.drawLine(pen, 40, 150, 80, 50);

 pen = new com.mapdigit.drawing.Pen(greenColor, 3);
 graphics2D.drawLine(pen, 60, 150, 100, 50);

 pen = new com.mapdigit.drawing.Pen(greenColor, 5);
 graphics2D.drawLine(pen, 80, 150, 120, 50);

 pen = new com.mapdigit.drawing.Pen(greenColor, 7);
 graphics2D.drawLine(pen, 100, 150, 140, 50);

 pen = new com.mapdigit.drawing.Pen(greenColor, 10);
 graphics2D.drawLine(pen, 120, 150, 160, 50);

}

private void drawDash() {
```

```
 Color blackColor = new Color(0xff000000);
 int dashArray1[] = { 2, 2 };
 int dashArray2[] = { 6, 6 };
 int dashArray3[] = { 4, 1, 2, 1, 1, 6 };
 // Clear the canvas with white color.
 graphics2D.clear(Color.WHITE);
 AffineTransform matrix = new AffineTransform();
 graphics2D.setAffineTransform(matrix);

 com.mapdigit.drawing.Pen pen
    = new com.mapdigit.drawing.Pen(blackColor,
  20, com.mapdigit.drawing.Pen.CAP_BUTT,
  com.mapdigit.drawing.Pen.JOIN_MITER, dashArray1, 0);
 graphics2D.drawLine(pen, 40, 60, 140, 60);

 pen = new com.mapdigit.drawing.Pen(blackColor, 20,
  com.mapdigit.drawing.Pen.CAP_BUTT,
  com.mapdigit.drawing.Pen.JOIN_MITER, dashArray2, 0);
 graphics2D.drawLine(pen, 40, 100, 140, 100);

 pen = new com.mapdigit.drawing.Pen(blackColor, 20,
  com.mapdigit.drawing.Pen.CAP_BUTT,
  com.mapdigit.drawing.Pen.JOIN_MITER, dashArray3, 0);
 graphics2D.drawLine(pen, 40, 140, 140, 140);
 }
}
```

# 14

# Context Menu 绘制几何图形

上下文相关菜单（Context Menu）类同 PC 上按鼠标右键显示的菜单，在 Android 平台上是长按来激活Context Menu，Context Menu 一般用来显示和当前UI内容相关的菜单。

Context Menu 的用法和 Option Menu 非常类似：

首先是创建 菜单资源，在 res\menu 下新建 menu_context_shape.xml，用来显示 Oval，Pear，Shape2D:

```
<?xml version=”1.0″ encoding=”utf-8″?>
<menu
 xmlns:android=”[http://schemas.android.com/apk/res/android](http://schemas.android.com/apk/res/android)“>
<item
  android:id=”@+id/mnuOval”
  android:title=”Oval”>
</item>
<item
   android:id=”@+id/mnuPear”
   android:title=”Pear”>
</item>
<item
   android:id=”@+id/mnuShape2DDemo”
   android:title=”Shape2D”>
</item>
</menu>
```

展开 Context Menu，是通过 onCreateContextMenu 方法：

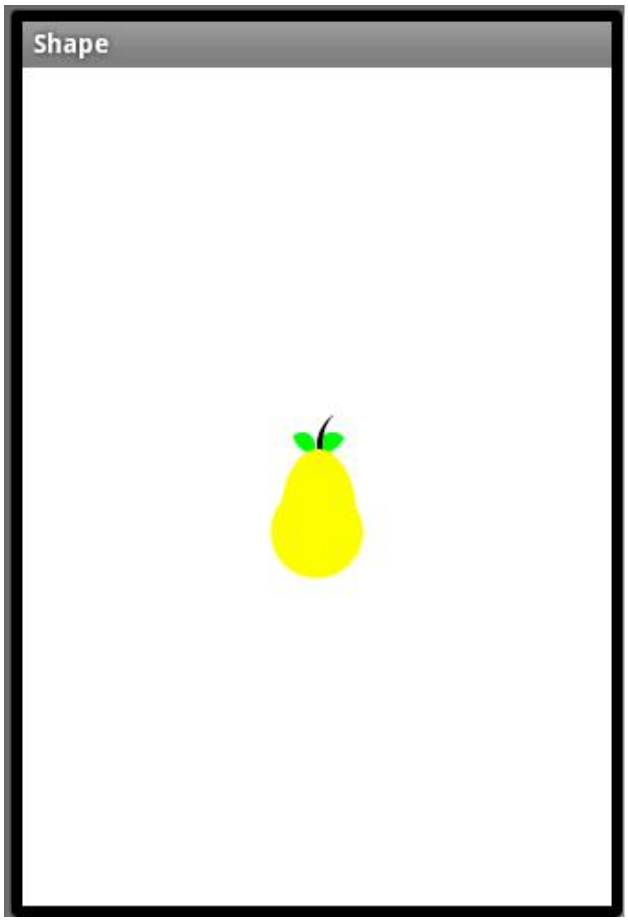
```
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
 ContextMenuInfo menuInfo) {
 super.onCreateContextMenu(menu, v, menuInfo);
 MenuInflater inflater = getMenuInflater();
 inflater.inflate(R.menu.menu_context_shape, menu);
}
```

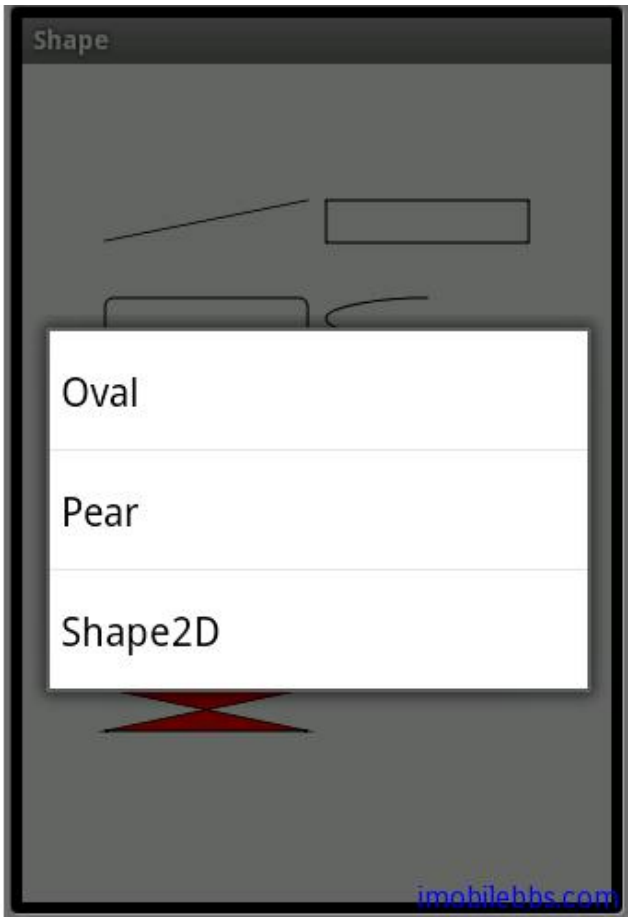
处理 Context Menu 事件：

```
@Override
public boolean onContextItemSelected(MenuItem item) {

 menuOption = item.getItemId();
 drawImage();
 return super.onContextItemSelected(item);

}
```

为了在长按时能在 View 上显示 Context Menu，需要为 View 注册 Context Menu：

```
public void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 registerForContextMenu(graphic2dView);
}
```




完整代码如下：

```
public class Shape extends Graphics2DActivity {

 private int menuOption;

 public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  registerForContextMenu(graphic2dView);
 }

 @Override
 protected void drawImage() {
  switch (menuOption) {
  case R.id.mnuOval:
   drawOval();
   break;
```

```
 case R.id.mnuPear:
  drawPear();
  break;
 case R.id.mnuShape2DDemo:
  drawShape2D();
  break;
 default:
  drawOval();
  break;
 }
 graphic2dView.refreshCanvas();

}

@Override
public void onCreateContextMenu(ContextMenu menu, View v,
  ContextMenuInfo menuInfo) {
 super.onCreateContextMenu(menu, v, menuInfo);
 MenuInflater inflater = getMenuInflater();
 inflater.inflate(R.menu.menu_context_shape, menu);
}

@Override
public boolean onContextItemSelected(MenuItem item) {

 menuOption = item.getItemId();
 drawImage();
 return super.onContextItemSelected(item);

}

private void drawOval() {
 AffineTransform mat1;

 /** Colors */
 Color redColor = new Color(0x96ff0000, true);
 Color greenColor = new Color(0xff00ff00);
 mat1 = new AffineTransform();
 mat1.translate(30, 40);
 mat1.rotate(-30 * Math.PI / 180.0);
 // Clear the canvas with white color.
 graphics2D.clear(Color.WHITE);
 graphics2D.Reset();

 graphics2D.setAffineTransform(new AffineTransform());
```

```
 SolidBrush brush = new SolidBrush(greenColor);
 graphics2D.fillOval(brush, 20, 60, 100, 50);

 com.mapdigit.drawing.Pen pen
   = new com.mapdigit.drawing.Pen(redColor, 5);
 graphics2D.setAffineTransform(mat1);
 graphics2D.drawOval(pen, 20, 60, 100, 50);
}

private void drawPear() {
 Ellipse circle, oval, leaf, stem;
 Area circ, ov, leaf1, leaf2, st1, st2;
 circle = new Ellipse();
 oval = new Ellipse();
 leaf = new Ellipse();
 stem = new Ellipse();
 circ = new Area(circle);
 ov = new Area(oval);
 leaf1 = new Area(leaf);
 leaf2 = new Area(leaf);
 st1 = new Area(stem);
 st2 = new Area(stem);
 graphics2D.clear(Color.WHITE);
 graphics2D.Reset();
 int w = SharedGraphics2DInstance.CANVAS_WIDTH;
 int h = SharedGraphics2DInstance.CANVAS_HEIGHT;
 int ew = w / 2;
 int eh = h / 2;
 SolidBrush brush = new SolidBrush(Color.GREEN);
 graphics2D.setDefaultBrush(brush);
 // Creates the first leaf by filling the
 //intersection of two Area
 // objects created from an ellipse.
 leaf.setFrame(ew - 16, eh - 29, 15, 15);
 leaf1 = new Area(leaf);
 leaf.setFrame(ew - 14, eh - 47, 30, 30);
 leaf2 = new Area(leaf);
 leaf1.intersect(leaf2);
 graphics2D.fill(null, leaf1);

 // Creates the second leaf.
 leaf.setFrame(ew + 1, eh - 29, 15, 15);
 leaf1 = new Area(leaf);
 leaf2.intersect(leaf1);
 graphics2D.fill(null, leaf2);
```

```
 brush = new SolidBrush(Color.BLACK);
 graphics2D.setDefaultBrush(brush);

 // Creates the stem by filling the Area
 //resulting from the subtraction of two
 //Area objects created from an ellipse.
 stem.setFrame(ew, eh − 42, 40, 40);
 st1 = new Area(stem);
 stem.setFrame(ew + 3, eh − 47, 50, 50);
 st2 = new Area(stem);
 st1.subtract(st2);
 graphics2D.fill(null, st1);

 brush = new SolidBrush(Color.YELLOW);
 graphics2D.setDefaultBrush(brush);

 // Creates the pear itself by filling the
 //Area resulting from the union of two Area
 //objects created by two different ellipses.
 circle.setFrame(ew − 25, eh, 50, 50);
 oval.setFrame(ew − 19, eh − 20, 40, 70);
 circ = new Area(circle);
 ov = new Area(oval);
 circ.add(ov);
 graphics2D.fill(null, circ);
}

private void drawShape2D() {
 Color bg = Color.white;
 Color fg = Color.black;
 Color red = Color.red;
 Color white = Color.white;
 com.mapdigit.drawing.Pen pen
    = new com.mapdigit.drawing.Pen(fg, 1);
 SolidBrush brush = new SolidBrush(red);
 // Clear the canvas with white color.
 graphics2D.clear(bg);
 graphics2D.Reset();
 Dimension d = new Dimension(SharedGraphics2DInstance.CANVAS_WIDTH,
   SharedGraphics2DInstance.CANVAS_HEIGHT);
 int gridWidth = d.width / 2;
 int gridHeight = d.height / 6;

 int x = 5;
```

```
int y = 7;
int rectWidth = gridWidth − 2 * x;
int stringY = gridHeight − 3 − 2 − 16;
int rectHeight = stringY − y − 2;
graphics2D.draw(pen, new Line(x, y + rectHeight − 1,
  x + rectWidth, y));
x += gridWidth;
graphics2D.draw(pen, new Rectangle(x, y, rectWidth,
  rectHeight));
x += gridWidth;
x = 5;
y += gridHeight;
stringY += gridHeight;
graphics2D.draw(pen, new RoundRectangle(x, y, rectWidth,
  rectHeight,
  10, 10));
x += gridWidth;
graphics2D.draw(pen, new Arc(x, y, rectWidth,
  rectHeight, 90, 135,
  Arc.OPEN));
x = 5;
y += gridHeight;
stringY += gridHeight;
graphics2D.draw(pen, new Ellipse(x, y, rectWidth,
  rectHeight));
x += gridWidth;
// draw GeneralPath (polygon)
int x1Points[] = { x, x + rectWidth, x,
  x + rectWidth };
int y1Points[] = { y, y + rectHeight,
  y + rectHeight, y };
com.mapdigit.drawing.geometry.Path polygon
  = new com.mapdigit.drawing.geometry.Path(
  com.mapdigit.drawing.geometry.Path.WIND_EVEN_ODD,
  x1Points.length);
polygon.moveTo(x1Points[0], y1Points[0]);
for (int index = 1; index < x1Points.length; index++) {
 polygon.lineTo(x1Points[index], y1Points[index]);
}
polygon.closePath();
graphics2D.draw(pen, polygon);
x = 5;
y += gridHeight;
stringY += gridHeight;
int x2Points[] = { x, x + rectWidth, x, x + rectWidth };
```

```
int y2Points[] = { y, y + rectHeight, y + rectHeight, y };
com.mapdigit.drawing.geometry.Path polyline
  = new com.mapdigit.drawing.geometry.Path(
  com.mapdigit.drawing.geometry.Path.WIND_EVEN_ODD,
  x2Points.length);
polyline.moveTo(x2Points[0], y2Points[0]);
for (int index = 1; index < x2Points.length; index++) {
 polyline.lineTo(x2Points[index], y2Points[index]);
}
graphics2D.draw(pen, polyline);
x += gridWidth;
graphics2D.setPenAndBrush(pen, brush);
graphics2D.fill(null,
  new Rectangle(x, y, rectWidth, rectHeight));
graphics2D.draw(null,
  new Rectangle(x, y, rectWidth, rectHeight));
x = 5;
y += gridHeight;
stringY += gridHeight;
Color[] colors = new Color[] { red, white };
int[] fractions = new int[] { 0, 255 };
LinearGradientBrush redtowhite
    = new LinearGradientBrush(x, y, x
  + rectWidth, y, fractions, colors,
  com.mapdigit.drawing.Brush.NO_CYCLE);
graphics2D.setPenAndBrush(pen, redtowhite);
graphics2D.fill(null, new RoundRectangle(x, y, rectWidth,
  rectHeight,
  10, 10));
graphics2D.draw(null, new RoundRectangle(x, y, rectWidth,
  rectHeight,
  10, 10));
x += gridWidth;
graphics2D.setPenAndBrush(pen, brush);
graphics2D.fill(null, new Arc(x, y, rectWidth,
  rectHeight, 90, 135,
  Arc.CHORD));
graphics2D.draw(null, new Arc(x, y, rectWidth,
  rectHeight, 90, 135,
  Arc.CHORD));
x = 5;
y += gridHeight;
stringY += gridHeight;
int x3Points[] = { x, x + rectWidth, x, x + rectWidth };
int y3Points[] = { y, y + rectHeight, y + rectHeight, y };
```

```
  com.mapdigit.drawing.geometry.Path filledPolygon
  = new com.mapdigit.drawing.geometry.Path(
    com.mapdigit.drawing.geometry.Path.WIND_EVEN_ODD,
    x3Points.length);
  filledPolygon.moveTo(x3Points[0], y3Points[0]);
  for (int index = 1; index < x3Points.length; index++) {
   filledPolygon.lineTo(x3Points[index], y3Points[index]);
  }
  filledPolygon.closePath();
  graphics2D.setPenAndBrush(pen, brush);
  graphics2D.fill(null, filledPolygon);
  graphics2D.draw(null, filledPolygon);

 }
}
```

菜单除了这里介绍的功能外，Android 也支持动态菜单或动态修改菜单。具体可以参见 Android 文档。

15

# RadioButton 多边形及路径绘制

这个例子是绘制多边形，多义形和路径，采用单选钮 RadioButton 来选择 Polys 和 Path 示例:

UI 设计为 上部分用来显示绘图内容，下部分为两个单选按钮 Polys ，Path。这样 layout 就和main.xml 不一样，main.xml 只含一个 com.pstreets.graphics2d.GuidebeeGraphics2DView。因此需在 res\layout 下新建一个 polys.xml:

```
<?xml version=”1.0″ encoding=”utf-8″ ?>
<LinearLayout xmlns:android=”[http://schemas.android.com/apk/res/android](http://schemas.android.com/apk/res/andro
   android:orientation=”vertical”
   android:background=”@drawable/white”
 android:layout_width=”fill_parent”
 android:layout_height=”fill_parent”>
   <com.pstreets.graphics2d.GuidebeeGraphics2DView
    android:id=”@+id/graphics2dview”
    android:layout_weight=”1″
    android:layout_width=”fill_parent”
    android:layout_height=”wrap_content”/>
 <LinearLayout xmlns:android=”[http://schemas.android.com/apk/res/android](http://schemas.android.com/apk/res/andr
 android:layout_width=”wrap_content” android:layout_height=”wrap_content”
 android:orientation=”horizontal”
```

```
>
 <RadioGroup
    android:layout_width=”wrap_content”
    android:orientation=”horizontal”
    android:textSize=”20dp”
    android:layout_height=”wrap_content”>
  <RadioButton android:text=”Polys”
      android:id=”@+id/radioPolys”
   android:layout_width=”wrap_content”
   android:textColor=”@color/black”
   android:checked=”true”
   android:layout_height=”wrap_content”>
  </RadioButton>
  <RadioButton android:text=”Path”
      android:id=”@+id/radioPath”
   android:layout_width=”wrap_content”
   android:textColor=”@color/black”
   android:layout_height=”wrap_content”>
  </RadioButton>
 </RadioGroup>
</LinearLayout>

</LinearLayout>
```

RadioButton 需包含在 RadioGroup 中做为一个分组，这里将 Polys 设为选中。

定义好 Layout 资源后，修改 Path.java

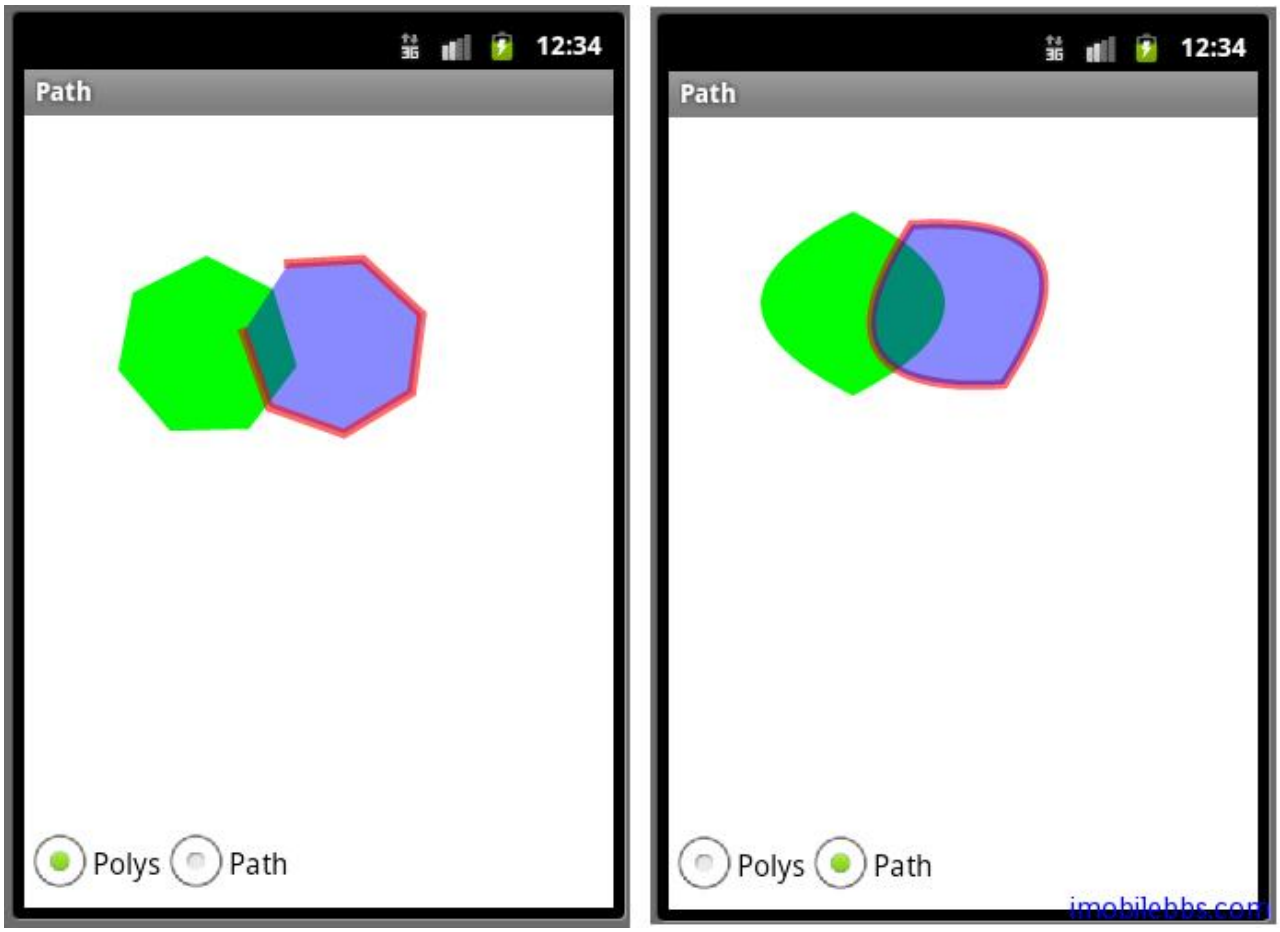
```
private RadioButton radioPoly;
private RadioButton radioPath;

public void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.polys);
 graphic2dView
  = (GuidebeeGraphics2DView)
    findViewById(R.id.graphics2dview);
 radioPath = (RadioButton) findViewById(R.id.radioPath);
 radioPoly = (RadioButton) findViewById(R.id.radioPolys);
 radioPath.setOnClickListener(this);
 radioPoly.setOnClickListener(this);
}
```

应为需要处理按键消息，所以定义了两个 RadioButton 对象，可以通过 findViewById 获取实例。因为两个 RadioButton 这里采用同样的处理方法，可以让 Path 实现 OnClickListener ，即：public class Path extends Graphics2DActivity implements OnClickListener。完整代码如下：

```
public class Path extends Graphics2DActivity
  implements OnClickListener {

   private RadioButton radioPoly;
   private RadioButton radioPath;

   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.polys);
      graphic2dView
       = (GuidebeeGraphics2DView)
         findViewById(R.id.graphics2dview);
      radioPath = (RadioButton) findViewById(R.id.radioPath);
      radioPoly = (RadioButton) findViewById(R.id.radioPolys);
      radioPath.setOnClickListener(this);
      radioPoly.setOnClickListener(this);
   }

   @Override
   protected void drawImage() {
      if (radioPoly.isChecked()) {
         drawPolys();
```

```
    } else {
        drawPaths();
    }
    graphic2dView.refreshCanvas();

}

@Override
public void onClick(View view) {
    drawImage();
}

private void drawPaths() {
    AffineTransform mat1;

    // The path.
    com.mapdigit.drawing.geometry.Path path;

    // Colors
    Color redColor = new Color(0x96ff0000, true);
    Color greenColor = new Color(0xff00ff00);
    Color blueColor = new Color(0x750000ff, true);

    String pathdata
      = "M 60 20 Q −40 70 60 120 Q 160 70 60 20 z";
    mat1 = new AffineTransform();
    mat1.translate(30, 40);
    mat1.rotate(−30 * Math.PI / 180.0);
    path = com.mapdigit.drawing.geometry.Path.fromString(pathdata);
    // Clear the canvas with white color.
    graphics2D.clear(Color.WHITE);

    graphics2D.setAffineTransform(new AffineTransform());
    SolidBrush brush = new SolidBrush(greenColor);
    graphics2D.fill(brush, path);
    graphics2D.setAffineTransform(mat1);

    brush = new SolidBrush(blueColor);
    com.mapdigit.drawing.Pen pen
      = new com.mapdigit.drawing.Pen(redColor, 5);
    graphics2D.setPenAndBrush(pen, brush);
    graphics2D.draw(null, path);
    graphics2D.fill(null, path);

}
```

```
  private void drawPolys() {
    AffineTransform mat1;

    // Colors
    Color redColor = new Color(0x96ff0000, true);
    Color greenColor = new Color(0xff00ff00);
    Color blueColor = new Color(0x750000ff, true);

    Polyline polyline;
    Polygon polygon;
    Polygon polygon1;

    String pointsdata1
    = "59,45,95,63,108,105,82,139,39,140,11,107,19,65";
    mat1 = new AffineTransform();
    mat1.translate(30, 40);
    mat1.rotate(-30 * Math.PI / 180.0);
    polyline = new Polyline();
    polygon = new Polygon();
    polygon1 = new Polygon();
    Point[] points = Point.fromString(pointsdata1);
    for (int i = 0; i < points.length; i++) {
      polyline.addPoint(points[i].x, points[i].y);
      polygon.addPoint(points[i].x, points[i].y);
      polygon1.addPoint(points[i].x, points[i].y);
    }
    // Clear the canvas with white color.
    graphics2D.clear(Color.WHITE);

    graphics2D.setAffineTransform(new AffineTransform());
    SolidBrush brush = new SolidBrush(greenColor);
    graphics2D.fillPolygon(brush, polygon);
    graphics2D.setAffineTransform(mat1);

    brush = new SolidBrush(blueColor);
    com.mapdigit.drawing.Pen pen
      = new com.mapdigit.drawing.Pen(redColor, 5);
    graphics2D.setPenAndBrush(pen, brush);
    graphics2D.fillPolygon(null, polygon1);
    graphics2D.drawPolyline(null, polyline);

  }

}
```


12:34
Path
Polys
Path
12:34
Path
Polys
Path
imobilebbs.com

16

# Button 画刷示例

将 RadioButton 换成 Button，类似的在 res\layout 中新建 brush.xml：

```
<?xml version=”1.0″ encoding=”utf-8″ ?>
<LinearLayout xmlns:android=”[http://schemas.android.com/apk/res/android](http://schemas.android.com/apk/res/andro
   android:orientation=”vertical”
   android:background=”@drawable/white”
 android:layout_width=”fill_parent”
 android:layout_height=”fill_parent”>
   <com.pstreets.graphics2d.GuidebeeGraphics2DView
    android:id=”@+id/graphics2dview”
    android:layout_weight=”1″
    android:layout_width=”fill_parent”
    android:layout_height=”wrap_content”/>
 <LinearLayout xmlns:android=”[http://schemas.android.com/apk/res/android](http://schemas.android.com/apk/res/andr
  android:layout_width=”wrap_content” android:layout_height=”wrap_content”
  android:orientation=”horizontal”
>

  <Button android:text=”Pattern”
     android:id=”@+id/btnPattern”
   android:layout_width=”wrap_content”
   android:textColor=”@color/black”
   android:checked=”true”
   android:layout_height=”wrap_content”>
  </Button>
  <Button android:text=”Gradients”
     android:id=”@+id/btnGradients”
   android:layout_width=”wrap_content”
   android:textColor=”@color/black”
   android:layout_height=”wrap_content”>
  </Button>

 </LinearLayout>

</LinearLayout>
```

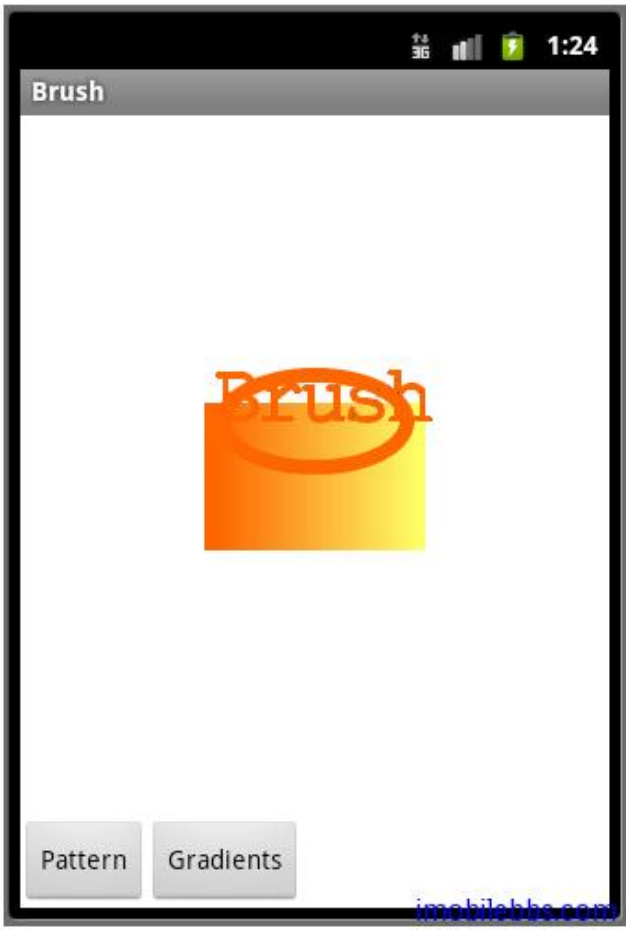
修改 Brushes.java ,完整代码如下：

```
public class Brushes extends Graphics2DActivity
  implements OnClickListener {

 private Button btnPattern;
 private Button btnGradients;

 public void onCreate(Bundle savedInstanceState) {
```

```
 super.onCreate(savedInstanceState);
 setContentView(R.layout.brush);
 graphic2dView = (GuidebeeGraphics2DView)
    findViewById(R.id.graphics2dview);
 btnPattern = (Button) findViewById(R.id.btnPattern);
 btnGradients = (Button) findViewById(R.id.btnGradients);
 btnPattern.setOnClickListener(this);
 btnGradients.setOnClickListener(this);
}

@Override
protected void drawImage() {
 drawPatterns();

}

@Override
public void onClick(View view) {
 if (view == btnPattern) {
  drawPatterns();
 } else {
  drawGradient();
 }
 graphic2dView.refreshCanvas();

}

private void drawPatterns() {
 TextureBrush brush1;
 TextureBrush brush2;
 TextureBrush brush3;

 AffineTransform matrix1 = new AffineTransform();
 AffineTransform matrix2 = new AffineTransform();
 Bitmap bitmap
  = BitmapFactory.decodeResource(getResources(),
  R.drawable.brick);
 int[] rgbData = new int[bitmap.getHeight()
                 * bitmap.getWidth()];
 bitmap.getPixels(rgbData, 0, bitmap.getWidth(), 0, 0,
  bitmap.getWidth(), bitmap.getHeight());
 brush1 = new TextureBrush(rgbData, bitmap.getWidth(),
  bitmap.getHeight());

 bitmap = BitmapFactory.decodeResource(getResources(),
```

```
    R.drawable.bird);
  rgbData = new int[bitmap.getHeight() * bitmap.getWidth()];
  bitmap.getPixels(rgbData, 0, bitmap.getWidth(), 0, 0,
    bitmap.getWidth(), bitmap.getHeight());
  brush2 = new TextureBrush(rgbData, bitmap.getWidth(),
    bitmap.getHeight());
  brush3 = new TextureBrush(rgbData, bitmap.getWidth(),
    bitmap.getHeight(), 127);
  matrix2.translate(50, 50);
  // Clear the canvas with white color.
  graphics2D.clear(Color.WHITE);
  graphics2D.setAffineTransform(matrix1);
  graphics2D.fillRectangle(brush1,
     new Rectangle(20, 50, 100, 100));
  graphics2D.fillOval(brush2, 10, 10, 80, 80);
  graphics2D.setAffineTransform(matrix2);
  graphics2D.fillOval(brush3, 10, 10, 80, 80);

 }

 private void drawGradient() {
  /* The linear gradient color */
  LinearGradientBrush brush1;
  /* The radial gradient color */
  RadialGradientBrush brush2;
  /* The second radial gradient color */
  RadialGradientBrush brush3;

  char[] engText = "Brush".toCharArray();

  FontEx font = FontEx.getSystemFont();

  int fontSize = 44;
  int X = 15;
  int Y = 50;
  int[] fractions = new int[] { 13, 242 };
  Color[] colors = new Color[] { new Color(0xffff6600),
    new Color(0xffffff66) };
  brush1 = new LinearGradientBrush(50, 50, 150, 125,
    fractions, colors,
    Brush.NO_CYCLE);

  fractions = new int[] { 13, 128, 255 };
  colors = new Color[] { new Color(0xffff6600),
    new Color(0xffffff66),
```

```
    new Color(0xffff6600) };
  brush2 = new RadialGradientBrush(90, 100, 50,
    fractions, colors);

  fractions = new int[] { 0, 255 };
  colors = new Color[] { new Color(0xFFFFFF00),
    new Color(0xFF000000) };
  brush3 = new RadialGradientBrush(50, 50, 100,
    fractions, colors);
  // Clear the canvas with white color.
  graphics2D.clear(Color.white);
  graphics2D.fillRectangle(brush1,
    new Rectangle(10, 75, 120, 80));

  Pen pen = new Pen(brush2, 8);
  graphics2D.drawOval(pen, 20, 60, 100, 50);
  graphics2D.setDefaultBrush(brush3);
  pen = new Pen(brush2, 2);
  graphics2D.setDefaultPen(pen);
  graphics2D.drawChars(font, fontSize, engText, 0,
    engText.length, X, Y);
 }

}
```


介绍了 RadioButton和Button 后，这时应该对使用 Android 提供的控件的用法有了基本的认识。 控件提供了 onClick(),onLongClick(),onFocusChange(),onKey(),onTouch(),onCreateContextMenu()等多种事件以相应用户。用多种方法来处理用户事件。一种是示例代码同过 Activity 实现OnClickListener 接口，再有是采用如下代码为 Button 支持事件处理方法：

```
// Create an anonymous implementation of OnClickListenerprivate
OnClickListener mCorkyListener = new OnClickListener() {
  public void onClick(View v) {
  // do something when the button is clicked
  }
};

protected void onCreate(Bundle savedValues) {
  ...
  // Capture our button from layout
  Button button = (Button)findViewById(R.id.corky);
  // Register the onClick listener with the implementation above
  button.setOnClickListener(mCorkyListener);
  ...
 }
```

在创建自定义控件时，也可以重载 onKeyDown(int, KeyEvent),onKeyUp(int, KeyEvent) ,onTouchEvent(MotionEvent)等来处理用户事件。

17

# Dialog 显示图像

Dialog 一般指可以显示在 Activity 前面的小窗口，当前的 Activity 失去焦点（Focus），Dialog 将接受用户输入，一般可以用来显示消息或接受用户输入等等。使用 Dialog 时一般不需要直接创建 Dialog 类的实例。而是可以使用AlertDialog,ProgressDialog,DatePickerDialog,TimePickerDialog。最常用的是AlertDialog。下面就以使用 AlertDialog 为例，使用 AlertDialog 来选择显示图像的三个例子：DrawMap, JumbleImage，SeeThroughImage。其中 DrawMap 暂时不介绍，将在后面介绍 Internet 应用显示在线地图时再说。

通常 Dialog 是作为 Activity 一部分来创建的，也就是说在 Activity 的 onCreateDialog(int)中创建。当在 onCreateDialog(int)创建 Dialog 时，Android 系统将自动管理 Dialog 的状态，并把当前 Activity 作为 Dialog 的所有者。并且 Dialog 也继承当前 Activity 的一些属性，比如说 Option Menu。

创建好 Dialog 后，可以使用 showDialog(int) 来显示 Dialog ，showDialog 的参数为 Dialog的 ID。在显示 Dialog 之前，如果想对 Dialog 做些改动，可以在 onPrepareDialog(int, Dialog) 添加代码。dismiss()关闭对话框。如果在 Activity 中则使用 dismissDialog(int) 。

本例中使用一个按钮来触发 Dialog，在 res\layout 在添加 images.xml

```
<?xml version=”1.0″ encoding=”utf-8″?>
<LinearLayout xmlns:android=”[http://schemas.android.com/apk/res/android](http://schemas.android.com/apk/res/andro
   android:orientation=”vertical”
   android:background=”@drawable/white”
 android:layout_width=”fill_parent”
 android:layout_height=”fill_parent”>
   <com.pstreets.graphics2d.GuidebeeGraphics2DView
    android:id=”@+id/graphics2dview”
    android:layout_weight=”1″
    android:layout_width=”fill_parent”
    android:layout_height=”wrap_content”/>
 <LinearLayout xmlns:android=”http://schemas.android.com/apk/res/android”
 android:layout_width=”wrap_content” android:layout_height=”wrap_content”
 android:orientation=”horizontal”

 >

  <Button android:text=”Images”
     android:id=”@+id/btnImages”
   android:layout_width=”wrap_content”
   android:textColor=”@color/black”
   android:checked=”true”
   android:layout_height=”wrap_content”>
  </Button>

</LinearLayout>
```

```
</LinearLayout>
```

修改 Image.java

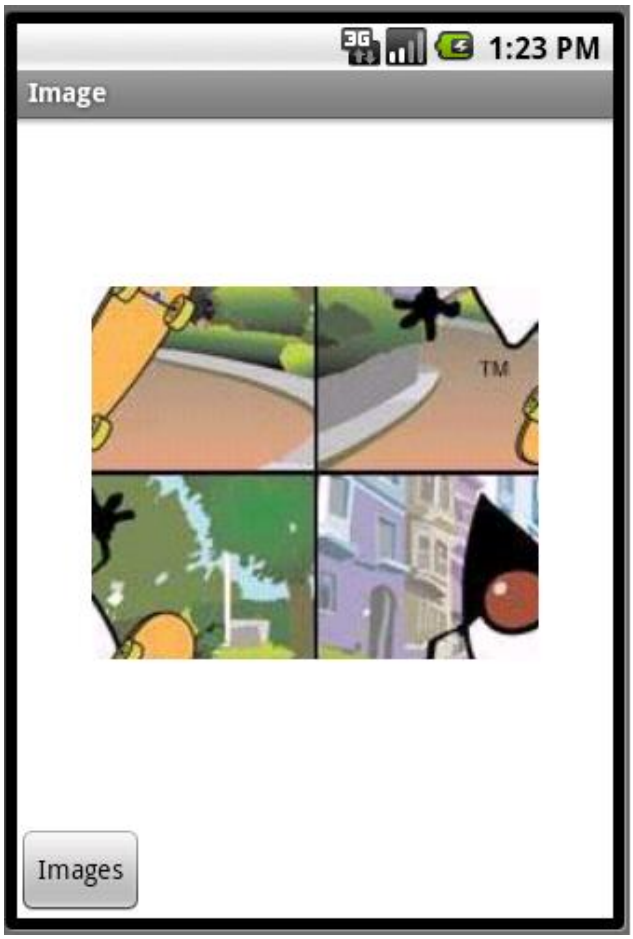
```
public class Images extends Graphics2DActivity
implements OnClickListener{

 private Button btnImages;
 private int[] imageDuke;

 static final private int IMAGE_DIALOG=1;

 int w, h;
   int offX, offY;

 int alpha = 128;
 FontEx font = FontEx.getSystemFont();
   int fontSize = 24;
   Pen pen = new Pen(Color.RED, 2);
   char[] message = "Guidebee".toCharArray();
   int widthOfMessage = 0;


   private int numlocs = 2;
   private int numcells = numlocs * numlocs;
   private int[] cells;
   int  cw, ch;



 public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.images);
  graphic2dView = (GuidebeeGraphics2DView)
     findViewById(R.id.graphics2dview);
  btnImages = (Button) findViewById(R.id.btnImages);
  btnImages.setOnClickListener(this);
  Bitmap bitmap
   = BitmapFactory.decodeResource(getResources(),
   R.drawable.duke_skateboard);
  imageDuke = new int[bitmap.getHeight()
                    * bitmap.getWidth()];
  bitmap.getPixels(imageDuke, 0, bitmap.getWidth(), 0, 0,
   bitmap.getWidth(), bitmap.getHeight());
  widthOfMessage = font.charsWidth(message, 0,
```

```
  message.length, fontSize);
 w=bitmap.getWidth();
   h=bitmap.getHeight();
      offX = (SharedGraphics2DInstance.CANVAS_WIDTH − w) / 2;
      offY = (SharedGraphics2DInstance.CANVAS_HEIGHT − h) / 2;

      cw = w / numlocs;
      ch = h / numlocs;
      cells = new int[numcells];
      for (int i = 0; i < numcells; i++) {
         cells[i] = i;
      }


}

private void drawJumbleImage(){
 Random rand = new Random();
      int ri;
      for (int i = 0; i < numcells; i++) {
         while ((ri = rand.nextInt(numlocs)) == i) {
         }

         int tmp = cells[i];
         cells[i] = cells[ri];
         cells[ri] = tmp;
      }
      graphics2D.clear(Color.WHITE);
      graphics2D.Reset();

      int dx, dy;
      for (int x = 0; x < numlocs; x++) {
         int sx = x * cw;
         for (int y = 0; y < numlocs; y++) {
            int sy = y * ch;
            int cell = cells[x * numlocs + y];
            dx = (cell / numlocs) * cw;
            dy = (cell % numlocs) * ch;
            graphics2D.drawImage(imageDuke, w, h,
                  dx + offX, dy + offY,
                  sx, sy, cw, ch);
         }
      }

      graphic2dView.refreshCanvas();
```

```
}

private void drawSeeThroughImage(){
 alpha += 16;
 if(alpha>255) alpha=0;
 graphics2D.clear(Color.WHITE);
 graphics2D.Reset();
 graphics2D.setDefaultPen(pen);
      graphics2D.drawChars(font, fontSize, message,
        0, message.length, offX
             + (w − widthOfMessage) / 2, offY + h / 2);
      graphics2D.drawImage(imageDuke, w, h,
             offX, offY,
             0xFFFF00FF, alpha);
      graphic2dView.refreshCanvas();
}

protected Dialog onCreateDialog(int id) {
 Dialog dialog;
 switch(id) {
   case IMAGE_DIALOG:
    final CharSequence[] items = {"DrawMap",
      "JumbleImage","SeeThroughImage"};
    AlertDialog.Builder builder
    = new AlertDialog.Builder(this);
    builder.setTitle("Images");
    builder.setSingleChoiceItems(items,
      −1, new DialogInterface.OnClickListener() {
         public void onClick(DialogInterface dialog,
           int item) {
            switch(item){
            case 0:

             break;
            case 1:
             drawJumbleImage();
             break;
            case 2:
             drawSeeThroughImage();
             break;

            }
            dialog.dismiss();
        }
        });
```

```
        AlertDialog alert = builder.create();
       dialog=alert;
       break;
        default:
         dialog = null;
       }
   return dialog;
 }

 @Override
 protected void drawImage() {
  drawJumbleImage();

 }

 @Override
 public void onClick(View view) {
  showDialog(IMAGE_DIALOG);

 }

}
```

从代码中看到，Dialog 是通过 AlertDialog.Builder 来创建的，这里 Dialog 显示了三个选项，通过 builder.setSingleChoiceItems 添加处理事件。实际 AlertDialog 可以有多种选项，具体请参考 Android AlertDialog 文档。


1:23 PM
Image
TM
Images


1:23 PM
Image
Images
DrawMap
JumbleImage
SeeThroughImage
Images
imobilebbs.com

18

# 自定义对话框 Transform

Android 自带的 AlertDialog,ProgressDialog,DatePickerDialog,TimePickerDialog 可以用于简单的对话框显示。当还是有这些对话框不能满足应用需要的时候，这时就可以使用一些自定义的对话框。有多种方法可以实现自定义对话框。一是使用 Activity 作为 Dialog，可以通过设置 Activity 显示 Dialog 风格，使得该 Activity 在外观上和 Dialog 一致：显示在其它 Activity 前面且半透明。

```
<Activity android:name=”MyDialogActivity” Android:theme=”@android:style/Theme.Dialog“></Activity>
```

本例采用另外两种方法来使用自定义对话框,将用这个对话框来最为图形变换(Transform)的选项：

```
Primitive: Rectange, Ellipse,Text
Pen: Thin,Thick,Dashed
Brush: Solid, Gradient,Texture
Transform: Identity, Rotate, Scale, Shear
Rendering: Stroke, Fill, Stoke and Fill
```

首先在 res\layout 下新建 transformoption.xml 作为自定义对话框布局：

```
<?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”

 >
<ScrollView
 android:layout_width=”fill_parent”
 android:layout_height=”fill_parent”>
<LinearLayout
 android:layout_width=”fill_parent”
 android:layout_height=”fill_parent”
 android:orientation=”vertical”
 >
<TextView
  android:text=”Primitive”
  android:layout_width=”wrap_content”
  android:layout_height=”wrap_content”>
 </TextView>
 <RadioGroup
 android:layout_width=”wrap_content”
 android:layout_height=”wrap_content”>

<RadioButton
android:text=”Rectangle”
android:id=”@+id/radioRectangle”
```

```
android:layout_width=”wrap_content”
android:layout_height=”wrap_content”>
</RadioButton>
<RadioButton
android:text=”Ellipse”
android:id=”@+id/radioEllipse”
android:layout_width=”wrap_content”
android:layout_height=”wrap_content”>
</RadioButton>
<RadioButton
android:text=”Text”
android:id=”@+id/radioText”
android:layout_width=”wrap_content”
android:layout_height=”wrap_content”>
</RadioButton>
 </RadioGroup>

<TextView
  android:text=”Pen”
  android:layout_width=”wrap_content”
  android:layout_height=”wrap_content”>
 </TextView>
 <RadioGroup
 android:layout_width=”wrap_content”
 android:layout_height=”wrap_content”>

<RadioButton
android:text=”Thin”
android:id=”@+id/radioThin”
android:layout_width=”wrap_content”
android:layout_height=”wrap_content”>
</RadioButton>
<RadioButton
android:text=”Thick”
android:id=”@+id/radioThick”
android:layout_width=”wrap_content”
android:layout_height=”wrap_content”>
</RadioButton>
<RadioButton
android:text=”Dashed”
android:id=”@+id/radioDashed”
android:layout_width=”wrap_content”
android:layout_height=”wrap_content”>
</RadioButton>
 </RadioGroup>
```

```
<TextView
  android:text=”Brush”
  android:layout_width=”wrap_content”
  android:layout_height=”wrap_content”>
 </TextView>
 <RadioGroup
 android:layout_width=”wrap_content”
 android:layout_height=”wrap_content”>

<RadioButton
android:text=”Solid”
android:id=”@+id/radioSolid”
android:layout_width=”wrap_content”
android:layout_height=”wrap_content”>
</RadioButton>
<RadioButton
android:text=”Gradient”
android:id=”@+id/radioGradient”
android:layout_width=”wrap_content”
android:layout_height=”wrap_content”>
</RadioButton>
<RadioButton
android:text=”Texture”
android:id=”@+id/radioTexture”
android:layout_width=”wrap_content”
android:layout_height=”wrap_content”>
</RadioButton>
 </RadioGroup>

 <TextView
  android:text=”Transform”
  android:layout_width=”wrap_content”
  android:layout_height=”wrap_content”>
 </TextView>
 <RadioGroup
 android:layout_width=”wrap_content”
 android:layout_height=”wrap_content”>

<RadioButton
android:text=”Identity”
android:id=”@+id/radioIdentity”
android:layout_width=”wrap_content”
android:layout_height=”wrap_content”>
</RadioButton>
```

```
<RadioButton
android:text=” Rotate”
android:id=” @+id/radioRotate”
android:layout_width=” wrap_content”
android:layout_height=” wrap_content” >
</RadioButton>
<RadioButton
android:text=” Scale”
android:id=” @+id/radioScale”
android:layout_width=” wrap_content”
android:layout_height=” wrap_content” >
</RadioButton>
<RadioButton
android:text=” Shear”
android:id=” @+id/radioShear”
android:layout_width=” wrap_content”
android:layout_height=” wrap_content” >
</RadioButton>
 </RadioGroup>

 <TextView
  android:text=” Rendering”
  android:layout_width=” wrap_content”
  android:layout_height=” wrap_content” >
 </TextView>
 <RadioGroup
 android:layout_width=” wrap_content”
 android:layout_height=” wrap_content” >

<RadioButton
android:text=” Stroke”
android:id=” @+id/radioStroke”
android:layout_width=” wrap_content”
android:layout_height=” wrap_content” >
</RadioButton>
<RadioButton
android:text=” Fill”
android:id=” @+id/radioFill”
android:layout_width=” wrap_content”
android:layout_height=” wrap_content” >
</RadioButton>
<RadioButton
android:text=” Stroke and Fill”
android:id=” @+id/radioStrokeFill”
android:layout_width=” wrap_content”
```

```
android:layout_height=”wrap_content”>
</RadioButton>
 </RadioGroup>
 </LinearLayout>
</ScrollView>
</LinearLayout>
```

一种方法是重新定制 AlertDialog，基本步骤和 Android 简明开发教程十七：Dialog 显示图像类似，但是在 protected Dialog onCreateDialog(int id)，需要重新设定 Dialog 的 Content View 并给 RadioButton添加事件处理：

```
protected Dialog onCreateDialog(int id) {
 final Dialog dialog;
 switch (id) {
 case OPTION_DIALOG:
  LayoutInflater li
    = LayoutInflater.from(this);
  View optionView
    = li.inflate(R.layout.transformoption, null);
  AlertDialog.Builder optionDialog
    = new AlertDialog.Builder(this);
  optionDialog.setTitle("Options");
  optionDialog.setView(optionView);
  dialog = optionDialog.create();
  RadioButton button = (RadioButton) optionView
    .findViewById(R.id.radioRectangle);
  button.setOnClickListener(new Button.OnClickListener() {

   public void onClick(View v) {
    primitiveIndex = PRIMITIVE_RECTANGLE;
    drawImage();
    dialog.dismiss();

   }
  });

  ...

  button = (RadioButton) optionView
    .findViewById(R.id.radioStrokeFill);
  button.setOnClickListener(new Button.OnClickListener() {

   public void onClick(View v) {
    renderingIndex = RENDERING_STROKE_AND_FILL;
    drawImage();
```

```
    dialog.dismiss();

   }
  });
  return dialog;
 }

 return null;

}
```

第二种是通过派生 Dialog，定义了一个 OptionDialog 类作为 Dialog 子类。

```
class OptionDialog extends Dialog {

 public OptionDialog(Context context) {
  super(context);
  // TODO Auto-generated constructor stub
 }

 protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.transformoption);
  setTitle("Options");
  RadioButton button
     = (RadioButton) findViewById(R.id.radioRectangle);
  button.setOnClickListener(new Button.OnClickListener() {

   public void onClick(View v) {
    primitiveIndex = PRIMITIVE_RECTANGLE;
    drawImage();
    dismiss();

   }
  });
  ...
  button = (RadioButton) findViewById(R.id.radioStrokeFill);
  button.setOnClickListener(new Button.OnClickListener() {

   public void onClick(View v) {
    renderingIndex = RENDERING_STROKE_AND_FILL;
    drawImage();
    dismiss();

   }
  });
```

```
 }

}
```

这两种方法在显示 Dialog 时有所不同：

```
private OptionDialog optionDialog;
static final private int OPTION_DIALOG = 1;

...
optionDialog = new OptionDialog(this);

...
@Override
public void onClick(View view) {
 // optionDialog.show();
 showDialog(OPTION_DIALOG);

}
```

下面是完整代码：

```
public class Transform extends Graphics2DActivity
  implements OnClickListener {

 static int PRIMITIVE_RECTANGLE = 0;
 static int PRIMITIVE_ELLIPSE = 1;
 static int PRIMITIVE_TEXT = 2;
 static int PEN_THIN = 0;
 static int PEN_THICK = 1;
 static int PEN_DASHED = 2;
 static int BRUSH_SOLID = 0;
 static int BRUSH_GRADIENT = 1;
 static int BRUSH_TEXTURE = 2;
 static int TRANSFORM_IDENTITY = 0;
 static int TRANSFORM_ROTATE = 1;
 static int TRANSFORM_SCALE = 2;
 static int TRANSFORM_SHEAR = 3;
 static int RENDERING_STROKE = 0;
 static int RENDERING_FILL = 1;
 static int RENDERING_STROKE_AND_FILL = 2;
 int primitiveIndex;
 int penIndex;
 int brushIndex;
 int transformIndex;
 int renderingIndex;
```

```
int[] rgbData;
int bitmapWidth;
int bitmapHeight;

class OptionDialog extends Dialog {

 public OptionDialog(Context context) {
  super(context);
  // TODO Auto-generated constructor stub
 }

 protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.transformoption);
  setTitle("Options");
  RadioButton button
    = (RadioButton) findViewById(R.id.radioRectangle);
  button.setOnClickListener(new Button.OnClickListener() {

   public void onClick(View v) {
    primitiveIndex = PRIMITIVE_RECTANGLE;
    drawImage();
    dismiss();

   }
  });
  button = (RadioButton) findViewById(R.id.radioEllipse);
  button.setOnClickListener(new Button.OnClickListener() {

   public void onClick(View v) {
    primitiveIndex = PRIMITIVE_ELLIPSE;
    drawImage();
    dismiss();

   }
  });
  button = (RadioButton) findViewById(R.id.radioText);
  button.setOnClickListener(new Button.OnClickListener() {

   public void onClick(View v) {
    primitiveIndex = PRIMITIVE_TEXT;
    drawImage();
    dismiss();

   }
```

```
});

button = (RadioButton) findViewById(R.id.radioThin);
button.setOnClickListener(new Button.OnClickListener() {

 public void onClick(View v) {
  penIndex = PEN_THIN;
  drawImage();
  dismiss();

 }
});

button = (RadioButton) findViewById(R.id.radioThick);
button.setOnClickListener(new Button.OnClickListener() {

 public void onClick(View v) {
  penIndex = PEN_THICK;
  drawImage();
  dismiss();

 }
});

button = (RadioButton) findViewById(R.id.radioDashed);
button.setOnClickListener(new Button.OnClickListener() {

 public void onClick(View v) {
  penIndex = PEN_DASHED;
  drawImage();
  dismiss();

 }
});

button = (RadioButton) findViewById(R.id.radioSolid);
button.setOnClickListener(new Button.OnClickListener() {

 public void onClick(View v) {
  brushIndex = BRUSH_SOLID;
  drawImage();
  dismiss();

 }
});
```

```
button = (RadioButton) findViewById(R.id.radioGradient);
button.setOnClickListener(new Button.OnClickListener() {

 public void onClick(View v) {
  brushIndex = BRUSH_GRADIENT;
  drawImage();
  dismiss();

 }
});

button = (RadioButton) findViewById(R.id.radioTexture);
button.setOnClickListener(new Button.OnClickListener() {

 public void onClick(View v) {
  brushIndex = BRUSH_TEXTURE;
  drawImage();
  dismiss();

 }
});

button = (RadioButton) findViewById(R.id.radioIdentity);
button.setOnClickListener(new Button.OnClickListener() {

 public void onClick(View v) {
  transformIndex = TRANSFORM_IDENTITY;
  drawImage();
  dismiss();

 }
});

button = (RadioButton) findViewById(R.id.radioRotate);
button.setOnClickListener(new Button.OnClickListener() {

 public void onClick(View v) {
  transformIndex = TRANSFORM_ROTATE;
  drawImage();
  dismiss();

 }
});
```

```
button = (RadioButton) findViewById(R.id.radioScale);
button.setOnClickListener(new Button.OnClickListener() {

 public void onClick(View v) {
  transformIndex = TRANSFORM_SCALE;
  drawImage();
  dismiss();

 }
});

button = (RadioButton) findViewById(R.id.radioShear);
button.setOnClickListener(new Button.OnClickListener() {

 public void onClick(View v) {
  transformIndex = TRANSFORM_SHEAR;
  drawImage();
  dismiss();

 }
});

button = (RadioButton) findViewById(R.id.radioStroke);
button.setOnClickListener(new Button.OnClickListener() {

 public void onClick(View v) {
  renderingIndex = RENDERING_STROKE;
  drawImage();
  dismiss();

 }
});

button = (RadioButton) findViewById(R.id.radioFill);
button.setOnClickListener(new Button.OnClickListener() {

 public void onClick(View v) {
  renderingIndex = RENDERING_FILL;
  drawImage();
  dismiss();

 }
});

button = (RadioButton) findViewById(R.id.radioStrokeFill);
```

```
  button.setOnClickListener(new Button.OnClickListener() {

   public void onClick(View v) {
    renderingIndex = RENDERING_STROKE_AND_FILL;
    drawImage();
    dismiss();

   }
  });
 }

}

private OptionDialog optionDialog;
private Button btnOptions;

static final private int OPTION_DIALOG = 1;

private AffineTransform at = new AffineTransform();
private int w, h;
private IShape shapes[] = new IShape[3];

private boolean firstTime = true;
private FontEx font = FontEx.getSystemFont();

@Override
protected void drawImage() {
 drawTransform();

}

protected Dialog onCreateDialog(int id) {
 final Dialog dialog;
 switch (id) {
 case OPTION_DIALOG:
  LayoutInflater li
    = LayoutInflater.from(this);
  View optionView
    = li.inflate(R.layout.transformoption, null);
  AlertDialog.Builder optionDialog
    = new AlertDialog.Builder(this);
  optionDialog.setTitle("Options");
  optionDialog.setView(optionView);
  dialog = optionDialog.create();
  RadioButton button = (RadioButton) optionView
```

```
  .findViewById(R.id.radioRectangle);
button.setOnClickListener(new Button.OnClickListener() {

 public void onClick(View v) {
  primitiveIndex = PRIMITIVE_RECTANGLE;
  drawImage();
  dialog.dismiss();

 }
});
button = (RadioButton)
    optionView.findViewById(R.id.radioEllipse);
button.setOnClickListener(new Button.OnClickListener() {

 public void onClick(View v) {
  primitiveIndex = PRIMITIVE_ELLIPSE;
  drawImage();
  dialog.dismiss();

 }
});
button = (RadioButton)
  optionView.findViewById(R.id.radioText);
button.setOnClickListener(new Button.OnClickListener() {

 public void onClick(View v) {
  primitiveIndex = PRIMITIVE_TEXT;
  drawImage();
  dialog.dismiss();

 }
});

button = (RadioButton)
   optionView.findViewById(R.id.radioThin);
button.setOnClickListener(new Button.OnClickListener() {

 public void onClick(View v) {
  penIndex = PEN_THIN;
  drawImage();
  dialog.dismiss();

 }
});
```

```
button = (RadioButton)
   optionView.findViewById(R.id.radioThick);
button.setOnClickListener(new Button.OnClickListener() {

 public void onClick(View v) {
  penIndex = PEN_THICK;
  drawImage();
  dialog.dismiss();

 }
});

button = (RadioButton)
    optionView.findViewById(R.id.radioDashed);
button.setOnClickListener(new Button.OnClickListener() {

 public void onClick(View v) {
  penIndex = PEN_DASHED;
  drawImage();
  dialog.dismiss();

 }
});

button = (RadioButton)
    optionView.findViewById(R.id.radioSolid);
button.setOnClickListener(new Button.OnClickListener() {

 public void onClick(View v) {
  brushIndex = BRUSH_SOLID;
  drawImage();
  dialog.dismiss();

 }
});

button = (RadioButton)
    optionView.findViewById(R.id.radioGradient);
button.setOnClickListener(new Button.OnClickListener() {

 public void onClick(View v) {
  brushIndex = BRUSH_GRADIENT;
  drawImage();
  dialog.dismiss();
```

```
 }
});

button = (RadioButton)
    optionView.findViewById(R.id.radioTexture);
button.setOnClickListener(new Button.OnClickListener() {

 public void onClick(View v) {
  brushIndex = BRUSH_TEXTURE;
  drawImage();
  dialog.dismiss();

 }
});

button = (RadioButton)
    optionView.findViewById(R.id.radioIdentity);
button.setOnClickListener(new Button.OnClickListener() {

 public void onClick(View v) {
  transformIndex = TRANSFORM_IDENTITY;
  drawImage();
  dialog.dismiss();

 }
});

button = (RadioButton)
    optionView.findViewById(R.id.radioRotate);
button.setOnClickListener(new Button.OnClickListener() {

 public void onClick(View v) {
  transformIndex = TRANSFORM_ROTATE;
  drawImage();
  dialog.dismiss();

 }
});

button = (RadioButton)
    optionView.findViewById(R.id.radioScale);
button.setOnClickListener(new Button.OnClickListener() {

 public void onClick(View v) {
  transformIndex = TRANSFORM_SCALE;
```

```
 drawImage();
 dialog.dismiss();

 }
});

button = (RadioButton)
  optionView.findViewById(R.id.radioShear);
button.setOnClickListener(new Button.OnClickListener() {

 public void onClick(View v) {
  transformIndex = TRANSFORM_SHEAR;
  drawImage();
  dialog.dismiss();

 }
});

button = (RadioButton)
   optionView.findViewById(R.id.radioStroke);
button.setOnClickListener(new Button.OnClickListener() {

 public void onClick(View v) {
  renderingIndex = RENDERING_STROKE;
  drawImage();
  dialog.dismiss();

 }
});

button = (RadioButton)
   optionView.findViewById(R.id.radioFill);
button.setOnClickListener(new Button.OnClickListener() {

 public void onClick(View v) {
  renderingIndex = RENDERING_FILL;
  drawImage();
  dialog.dismiss();

 }
});

button = (RadioButton) optionView
  .findViewById(R.id.radioStrokeFill);
button.setOnClickListener(new Button.OnClickListener() {
```

```
  public void onClick(View v) {
   renderingIndex = RENDERING_STROKE_AND_FILL;
   drawImage();
   dialog.dismiss();

  }
 });
 return dialog;
}

 return null;

}

public void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.transform);
 graphic2dView = (GuidebeeGraphics2DView)
   findViewById(R.id.graphics2dview);
 btnOptions = (Button) findViewById(R.id.btnOption);
 btnOptions.setOnClickListener(this);
 shapes[0] = new Rectangle(0, 0, 100, 100);
 shapes[1] = new Ellipse(0, 0, 100, 100);
 IShape[] fontShapes = font.getGlyphArray(96,
       "TEXT".toCharArray(), 0,
  4, FontEx.TEXT_DIR_LR);
 shapes[2] = new Area(fontShapes[0]);
 for (int i = 1; i < fontShapes.length; i++) {
  ((Area) shapes[2]).add(new Area(fontShapes[i]));
 }
 Bitmap bitmap = BitmapFactory.decodeResource(
        getResources(),
  R.drawable.brick);
 bitmapWidth = bitmap.getWidth();
 bitmapHeight = bitmap.getHeight();

 rgbData = new int[bitmapWidth * bitmapHeight];
 bitmap.getPixels(rgbData, 0, bitmapWidth, 0, 0,
   bitmapWidth,
  bitmapHeight);
 w = SharedGraphics2DInstance.CANVAS_WIDTH;
 h = SharedGraphics2DInstance.CANVAS_HEIGHT;
 optionDialog = new OptionDialog(this);
```

```
}

private void setTrans(int transIndex) {
 // Sets the AffineTransform.
 switch (transIndex) {
 case 0:
  at.setToIdentity();
  at.translate(w / 2, h / 2);
  break;
 case 1:
  at.rotate(Math.toRadians(45));
  break;
 case 2:
  at.scale(0.5, 0.5);
  break;
 case 3:
  at.shear(0.5, 0.0);
  break;
 }
}

private void drawTransform() {
 graphics2D.clear(Color.WHITE);
 graphics2D.Reset();
 setTrans(transformIndex);
 graphics2D.setAffineTransform(at);

 // Initialize the transform.
 if (firstTime) {
  at.setToIdentity();
  at.translate(w / 2, h / 2);

  firstTime = false;
 }

 // Sets the Stroke.
 Pen pen = null;

 switch (penIndex) {
 case 0:
  pen = new Pen(Color.BLACK, 1);
  break;
 case 1:
  pen = new Pen(Color.BLACK, 8);
  break;
```

```
case 2: {
 int dash[] = { 10, 10 };
 pen = new Pen(Color.BLACK, 1, Pen.CAP_BUTT,
   Pen.JOIN_MITER, dash, 0);
}
 break;
}

// Sets the Paint.
Brush brush = null;

switch (brushIndex) {
case 0:
 brush = new SolidBrush(Color.BLUE);
 break;
case 1: {
 int[] fractions = new int[] { 13, 242 };
 Color[] colors = new Color[] { new Color(0xffff6600),
   new Color(0xffffff66) };
 brush = new LinearGradientBrush(50, 50, 150,
   125, fractions,
   colors, Brush.REPEAT);
}

 break;
case 2:
 try {

  brush = new TextureBrush(rgbData,
    bitmapWidth, bitmapHeight);
 } catch (Exception e) {
 }
 break;
}

// Sets the Shape.
IShape shape = shapes[primitiveIndex];
Rectangle r = shape.getBounds();
AffineTransform toCenterAt = new AffineTransform();
toCenterAt.concatenate(at);
toCenterAt.translate(-(r.width / 2), -(r.height / 2));

graphics2D.setAffineTransform(toCenterAt);
// Sets the rendering method.
switch (renderingIndex) {
```

```
 case 0:
  graphics2D.setDefaultPen(pen);
  graphics2D.draw(null, shape);
  break;
 case 1:
  graphics2D.setDefaultBrush(brush);
  graphics2D.fill(null, shape);
  break;
 case 2:
  graphics2D.setPenAndBrush(pen, brush);
  graphics2D.fill(null, shape);
  graphics2D.draw(null, shape);
  break;
 }
 graphic2dView.refreshCanvas();

}

@Override
public void onClick(View view) {
 // optionDialog.show();
 showDialog(OPTION_DIALOG);

}
}
```

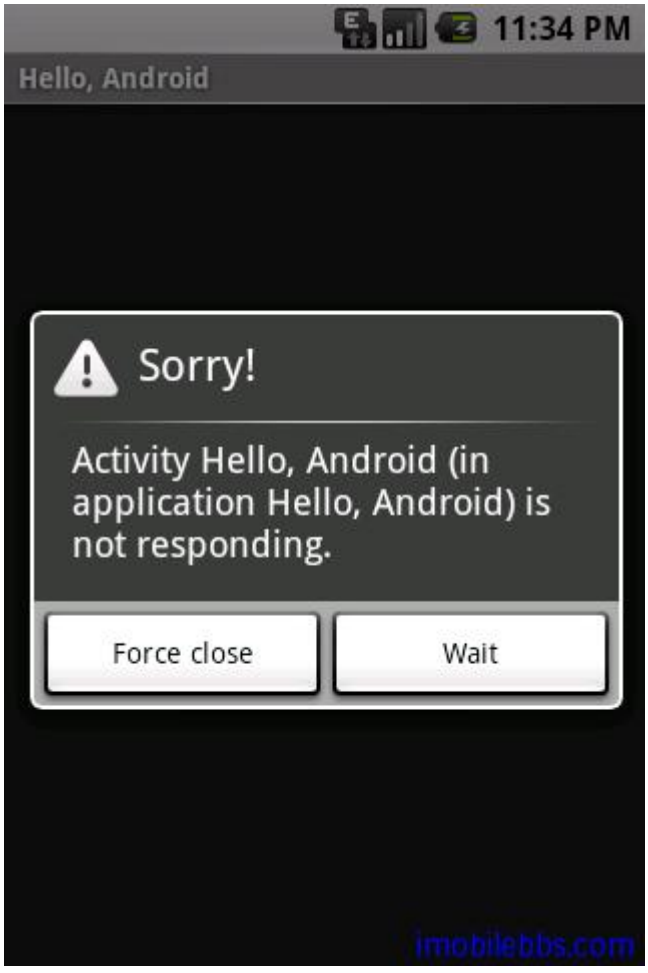

上面代码中包含了两种方法的代码，实际应用中可以任选其一

19

# 线程 Bezier 曲线

Android 中使用线程 Thread 的方法和 Java SE 相同。和大多数 OS 系统一样，Android 中也有称为 UI Thread 的主线程。UI Thread 主要用来给相应的 Widget 分发消息，包括绘制（Drawing）事件。UI Thread 也是用来处理用户交互事件的线程。比如：如果你按下屏幕上某个按钮，UI 线程则将Touch 事件通知对应的控件（Widgets)，Widget 则将其状态设置成“按下”，并把“重绘”（Invalidate）事件发到 Event Queue 中去。UI 线程从 Event Queue 中读取事件后通知Widgets 重画自身。

如果你的应用设计不好的话， UI 线程的这种单线程模式就会导致非常差的用户响应性能。特别是你将一些费时的操作如网络访问或数据库访问也放在 UI 线程中，这些操作会造成用户界面无反应，最糟糕的是，如果 UI 线程阻塞超过几秒（5秒），著名的 ANR 对话框就会出现：




所以在设计应用时，需要把一些费时的任务使用单独的工作线程来运行避免阻塞 UI 线程，但是如果在工作线程中想更新 UI 线程的话，不能直接在工作线程中更新 UI，这是因为 UI 线程不是“Thread Safe”。因此所有 UI 相关的操作一般必须在 UI Thread 中进行。

Android OS 提供了多种方法可以用在非 UI 线程访问 UI 线程。

- Activity.runOnUiThread(Runnable)
- View.post(Runnable)

- View.postDelayed(Runnable, long)
- Handler

Bezier 示例动态显示 Bezier 曲线，使用了 Activity.runOnUiThread 来更新屏幕，完整代码如下：

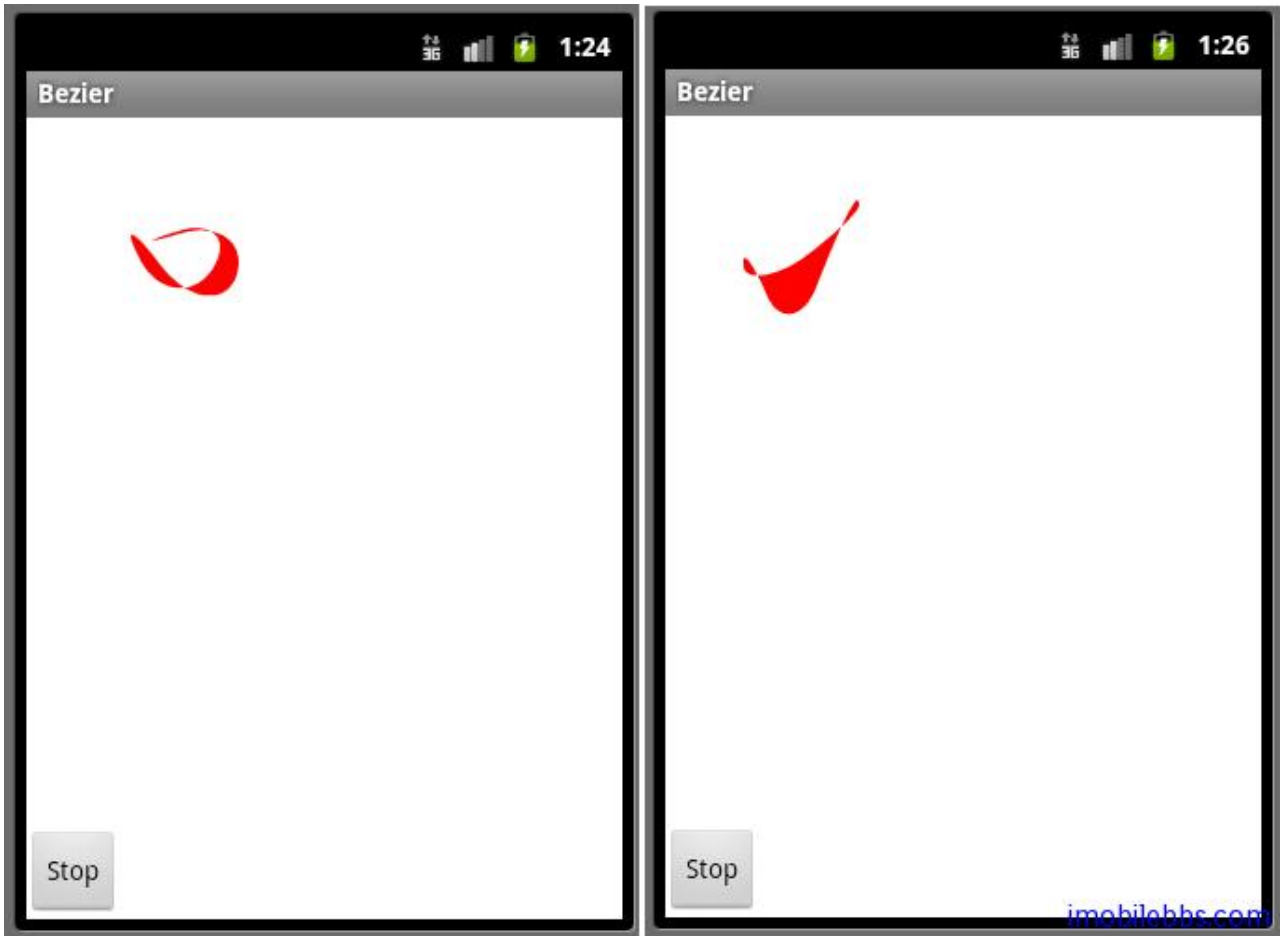
```
public class Bezier extends Graphics2DActivity
implements OnClickListener,Runnable{

 /**
    * The animation thread.
    */
   private Thread thread;
   private volatile boolean stopThread=false;
   private boolean stopOrNot=false;
   boolean drawn;
   /**
    * The random number generator.
    */
   static java.util.Random random = new java.util.Random();
   /**
    * The animated path
    */
   Path path = new Path();
   /**
    * Red brush used to fill the path.
    */
   SolidBrush brush = new SolidBrush(Color.RED);
   private static final int NUMPTS = 6;
   private int animpts[] = new int[NUMPTS * 2];
   private int deltas[] = new int[NUMPTS * 2];
   long startt, endt;

 private Button btnOptions;
 @Override
 protected void drawImage() {
   drawDemo(100, 100);

 }

 public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.beziers);
  graphic2dView
    = (GuidebeeGraphics2DView) findViewById(R.id.graphics2dview);
```

```
 btnOptions = (Button) findViewById(R.id.btnStopStart);
 btnOptions.setOnClickListener(this);
 reset(100,100);
 if (thread == null) {
          thread = new Thread(this);
          thread.start();
       }

}

@Override
public void onClick(View view) {

 if(!stopOrNot){
  btnOptions.setText("Start");
     stopThread=true;
 }
 else{
  stopThread=false;
  btnOptions.setText("Stop");
  if (thread == null) {
           thread = new Thread(this);
           thread.start();
        }
 }
 stopOrNot=!stopOrNot;

}
   /**
    * Generates new points for the path.
    */
   private void animate(int[] pts, int[] deltas,
     int i, int limit) {
       int newpt = pts[i] + deltas[i];
       if (newpt <= 0) {
           newpt = -newpt;
           deltas[i] = (random.nextInt() & 0x00000003)
           + 2;
       } else if (newpt >= limit) {
           newpt = 2 * limit - newpt;
           deltas[i] = -((random.nextInt() & 0x00000003)
             + 2);
       }
       pts[i] = newpt;
   }
```

```
  /**
   * Resets the animation data.
   */
  private void reset(int w, int h) {
      for (int i = 0; i < animpts.length; i += 2) {
          animpts[i + 0]
                  = (random.nextInt() & 0x00000003)
                  * w / 2;
          animpts[i + 1]
                  = (random.nextInt() & 0x00000003)
                  * h / 2;
          deltas[i + 0]
                  = (random.nextInt() & 0x00000003)
                  * 6 + 4;
          deltas[i + 1]
                  = (random.nextInt() & 0x00000003)
                  * 6 + 4;
          if (animpts[i + 0] > w / 2) {
              deltas[i + 0] = -deltas[i + 0];
          }
          if (animpts[i + 1] > h / 2) {
              deltas[i + 1] = -deltas[i + 1];
          }
      }
  }

   final Runnable updateCanvas = new Runnable() {
 public void run() {
  int offsetX = (graphic2dView.getWidth() -
    SharedGraphics2DInstance.CANVAS_WIDTH) / 2;
  int offsetY = (graphic2dView.getHeight()
    - SharedGraphics2DInstance.CANVAS_HEIGHT) / 2;
  graphic2dView.invalidate(offsetX,offsetY,
    offsetX+100,offsetY+100);
 }
};
    /**
     * Sets the points of the path and draws and fills the path.
     */
    private void drawDemo(int w, int h) {
        for (int i = 0; i < animpts.length; i += 2) {
            animate(animpts, deltas, i + 0, w);
            animate(animpts, deltas, i + 1, h);
        }
```

```
    //Generates the new pata data.
    path.reset();
    int[] ctrlpts = animpts;
    int len = ctrlpts.length;
    int prevx = ctrlpts[len − 2];
    int prevy = ctrlpts[len − 1];
    int curx = ctrlpts[0];
    int cury = ctrlpts[1];
    int midx = (curx + prevx) / 2;
    int midy = (cury + prevy) / 2;
    path.moveTo(midx, midy);
    for (int i = 2; i <= ctrlpts.length; i += 2) {
        int x1 = (curx + midx) / 2;
        int y1 = (cury + midy) / 2;
        prevx = curx;
        prevy = cury;
        if (i < ctrlpts.length) {
            curx = ctrlpts[i + 0];
            cury = ctrlpts[i + 1];
        } else {
            curx = ctrlpts[0];
            cury = ctrlpts[1];
        }
        midx = (curx + prevx) / 2;
        midy = (cury + prevy) / 2;
        int x2 = (prevx + midx) / 2;
        int y2 = (prevy + midy) / 2;
        path.curveTo(x1, y1, x2, y2, midx, midy);
    }
    path.closePath();
    // clear the clipRect area before production

    graphics2D.clear(Color.WHITE);
    graphics2D.fill(brush, path);

    this.runOnUiThread(updateCanvas);
}



public void run() {
    Thread me = Thread.currentThread();

    if (!drawn) {
        synchronized (this) {
```

```
            graphics2D.clear(Color.WHITE);
            graphics2D.fill(brush, path);
            graphic2dView.refreshCanvas();
            drawn = true;
          }
        }
        while (thread == me && !stopThread) {
          drawDemo(100,100);
        }
        thread = null;
    }
}
```




除了上述的方法外，Android 还提供了 AsyncTask 类以简化工作线程与 UI 线程之间的通信。这里不详述。此外，上面 Bezier 曲线动画在屏幕上显示时有闪烁的现象，这是动态显示图像的一个常见问题，后面将专门讨论。

20

# Broadcast Receiver 短信触发示例

Android 中 BroadCast Receiver 可以用来侦听广播事件。在使用 Broadcast 之前，必须使用代码或是在 AndroidManifest.xml 进行注册。

下面的例子实现使用短信来触发 AndroidGraphics2DTutorial 中的示例。短信格式为：@demo:xxxx ,xxxx为示例名称，比如， 启动 Colors 示例， 则向手机发送： @demo:Colors . 手机在收到短信后，先检测短信格式是否符合 @demo:xxxx, 若符合，这启动对应的示例。

在 AndroidGraphics2DTutorial 中添加一个自定义的 Broadcast Receiver SmsMessageReceiver 用于监测接受到的短信：

```
public class SmsMessageReceiver extends BroadcastReceiver {

  private static final String queryString="@demo:";
   @Override
   public void onReceive(Context context, Intent intent) {
      Bundle extras = intent.getExtras();
      if (extras == null)
         return;

      Object[] pdus = (Object[]) extras.get("pdus");

      for (int i = 0; i < pdus.length; i++) {
         SmsMessage message = SmsMessage.createFromPdu((byte[]) pdus[i]);
         String fromAddress = message.getOriginatingAddress();
         String fromDisplayName = fromAddress;
         String msg=message.getMessageBody();
         if(msg.startsWith(queryString)){
           // Trigger the main activity to fire up a dialog
       //that shows/reads the received messages
            Intent di = new Intent();
            di.setClass(context, AndroidGraphics2DTutorial.class);
            di.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK
                  | Intent.FLAG_ACTIVITY_SINGLE_TOP);
            di.putExtra(AndroidGraphics2DTutorial.SMS_FROM_ADDRESS_EXTRA,
                  fromAddress);
            di.putExtra(AndroidGraphics2DTutorial.SMS_FROM_DISPLAY_NAME_EXTRA,
                  fromDisplayName);
            di.putExtra(AndroidGraphics2DTutorial.SMS_MESSAGE_EXTRA, msg);
            context.startActivity(di);
         }


      }
```

```
  }
}
```

onReceive 会在 Broadcast 事件发生是执行，这里检测短信内容，如果是以@demo:开头的，则启动Android Graphics2DTutorial Main Activity。

修改 AndroidMainifest.xml

```
<receiver android:name=".SmsMessageReceiver" android:enabled="true">
        <intent-filter>
           <action android:name="android.provider.Telephony.SMS_RECEIVED" />
        </intent-filter>
 </receiver>
```

同时添加 permission ，和 Java ME 类似 Android 某些 API 需要指定对应的 Permission 才可以使用。

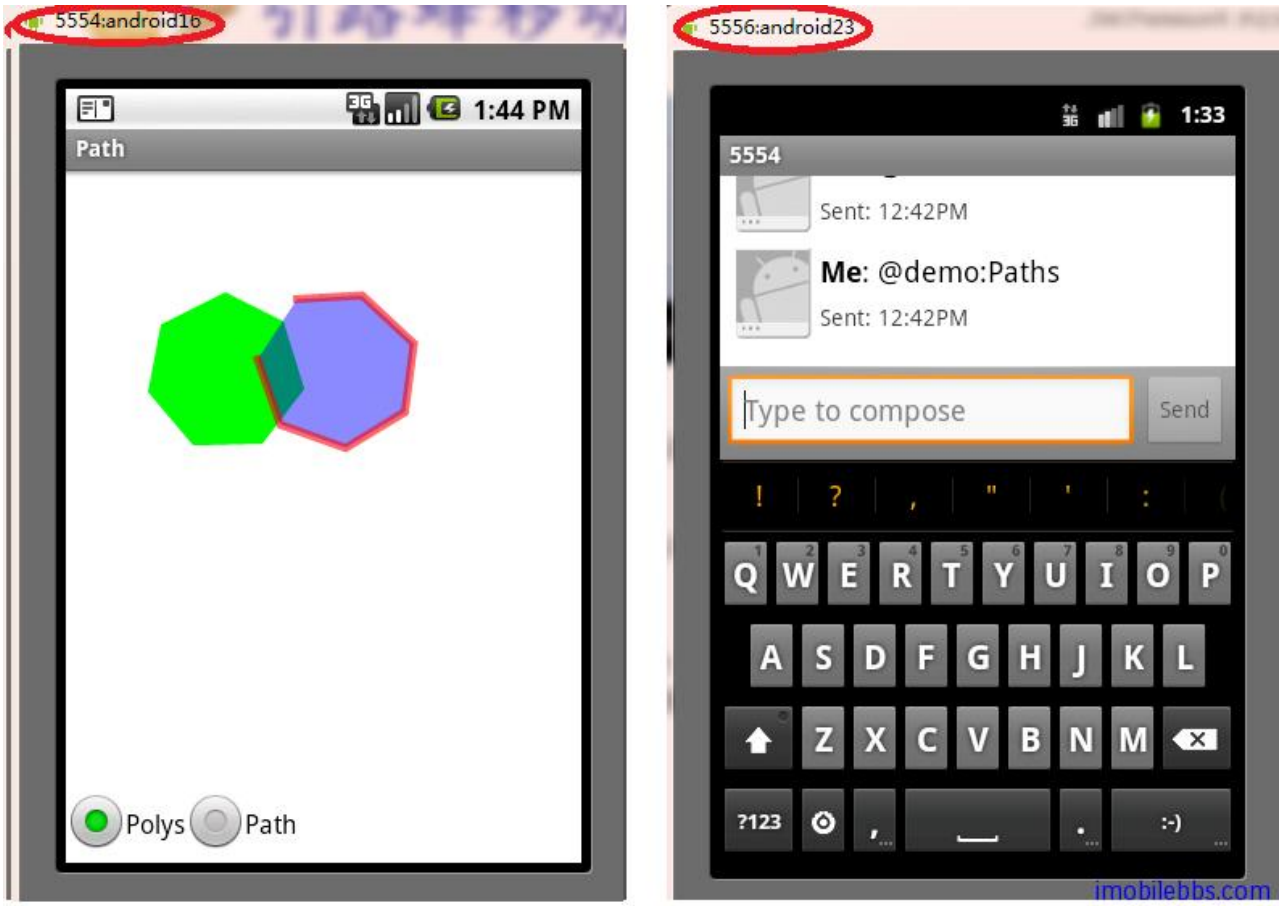
```
<uses-permission android:name="android.permission.SEND_SMS" />
<uses-permission android:name="android.permission.RECEIVE_SMS" />
```

修改 AndroidGraphics2DTutorial.java 来处理 SMS 消息：

```
public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      Resources res = getResources();
      String[] activity_Names = res.getStringArray(R.array.activity_name);
      String[] activity_Infos = res.getStringArray(R.array.activity_info);
      for(int i=0;i<activity_Names.length;i++){
         ActivityInfo activityInfo=new ActivityInfo();
         activityInfo.activityName=activity_Names[i];
         activityInfo.activityInfo=activity_Infos[i];
         activityInfo.iconIndex=R.drawable.icon1+i;
         activityInfos.add(activityInfo);
      }

      aa=new ActivityInfoAdapter(this,R.layout.activitylist,activityInfos);
      setListAdapter(aa);
      Bundle bundle=getIntent().getExtras();
      if(bundle!=null){
          mFromAddress = bundle.getString(SMS_FROM_ADDRESS_EXTRA);
          mMessage = bundle.getString(SMS_MESSAGE_EXTRA);
          int index=mMessage.indexOf(queryString);
          if(index>=0){
             String demoName=mMessage.substring(index+queryString.length());
             Intent intent = new Intent();
              intent.setClassName(this, packgeName+".example." +demoName);
              startActivity(intent);
```

```
        }

    }
}
```

下面来测试，如果使用设备，则给手机发送@demo:Colors . 如果使用模拟器，则可以启动两个模拟器：




模拟器左上角数字5554，5556为模拟器的号码。发送@demo:Paths ,则自动触发 Paths 示例，如果AndroidGraphics2DTutorial 没有运行，手机收到 SMS 短信后，会自动启动应用。

21

# 訪問 Internet 繪製在線地圖

在例子 Android 簡明開發教程十七：Dialog 顯示圖像中我們留了一個例子 DrawMap()沒有實現，這個例子顯示在線地圖，目前大部分地圖伺服器都是將地圖以圖片存儲以提高響應速度。 一般大小為256X256個像素。具體可以參見離線地圖下載方法解析。

比如： URL http://www.mapdigit.com/guidebeemap/maptile.php?type=MICROSOFTMAP&x=7&y=4&z=14顯示：


下面的例子訪問 Internet 下載地圖圖片，並拼接成地圖顯示,這種方法也是引路蜂地圖開發包實現的一個基本原則。

Android 應用訪問 Internet，首先需要賦予應用有訪問 Internet 的許可權：在AndroidManifest.xml 中添加：

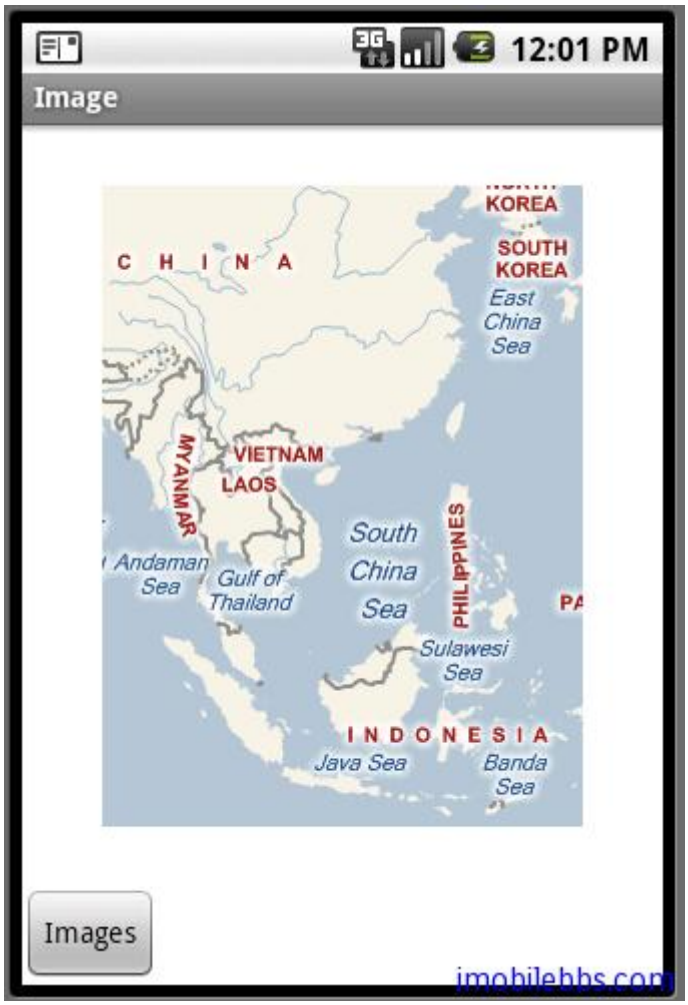
```
<uses-permission android:name=”android.permission.INTERNET” />
```

然後實現 DrawMap()如下：

```
private void drawMap(){
 try{

 graphics2D.clear(Color.WHITE);
 graphics2D.Reset();
 for(int x=6;x<8;x++)
 {
 for(int y=3;y<5;y++){
 String urlString="http://www.mapdigit.com/guidebeemap";
 urlString+="/maptile.php?type=MICROSOFTMAP";
 urlString+="&x="+x+"&y="+y+"&z=14";
 URL url=new URL(urlString);
 URLConnection connection=url.openConnection();
 HttpURLConnection httpConnection=(HttpURLConnection)connection;
 int responseCode=httpConnection.getResponseCode();
```

```
if(responseCode==HttpURLConnection.HTTP_OK){
InputStream stream=httpConnection.getInputStream();
Bitmap bitmap=BitmapFactory.decodeStream(stream);
int []buffer=new int[bitmap.getHeight()
* bitmap.getWidth()];
bitmap.getPixels(buffer, 0, bitmap.getWidth(), 0, 0,
bitmap.getWidth(), bitmap.getHeight());
graphics2D.drawImage(buffer,bitmap.getWidth(),
bitmap.getHeight(),(x-6)*256,(y-3)*256);

}
}
}
graphic2dView.refreshCanvas();

}catch(Exception e){

}
}
```

Android 中訪問 Internet 類主要定義在 java.net.* 和 android.net.*包中。上面顯示結果如下：




地圖沒有顯示滿屏是因為 Graphics2D 創建的 Canvas 大小沒有創建滿屏，創建的大小是240X320，如果創建滿屏的，則可以滿屏顯示地圖。

# 22

# 使用资源 Resources

在前面的例子中，我们忽略了一个重要的原则，在代码和 Layout 中，直接使用了字符串常量，比如：

```
<Button android:text=”Pattern”
android:id=”@+id/btnPattern”
android:layout_width=”wrap_content”
android:textColor=”@color/black”
android:checked=”true”
android:layout_height=”wrap_content”>
</Button>
```

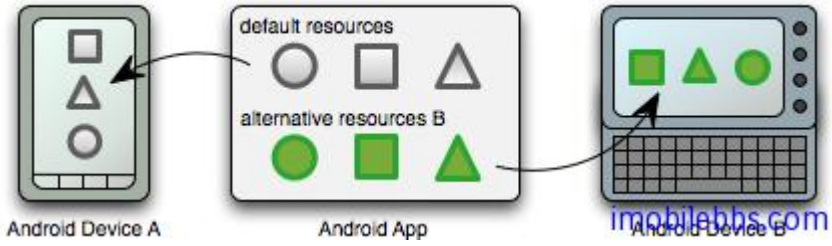
我们直接定义 Button 的显示内容为“Pattern”。如果你想你的应用支持多种设备，多种语言，那么直接使用字符串常量会给程序的移植带来很大的问题。因此设计应用是一个重要原则是尽可能的将 UI 相关的资源（如图像，文字等）以外部资源的形式来定义。

Android 支持多种资源类型，对应每一种资源，你可以定义一个缺省资源和多个可选资源（根据设备配置或语言类型等）。

缺省资源定义成与设备配置和语言无关，用在找不到与设备配置对应资源时使用。比如说你可以将缺省 UI Layout 定义在 res/layout 中， 而将屏幕横置（Landscape)定义在 res/layout-land 中。Android 在运行时会根据设备配置自动选择合适的资源。

下图显示两种不同配置的设备中没有定义可选资源时都使用缺省资源定义：


下图应用定义给两种不同设备定义了两种资源，一是缺省资源，一是为横屏显示时的资源：




在定义可选资源时，Android 对可选资源的命名方法有一定的规定，具体可以参见http://developer.android.com/guide/topics/resources/providing-resources.html

# 23

# 發布應用

到這裡基本介紹了 Android 開發的一些基本知識，在開發實際應用時最常用的幾個參考是：

- The Developer’s Guide
- Android References
- Android Resources 最後一個是Google.com： -）

寫好應用後，在設備上測試後，最後一步是發布你的應用。和 Java ME 平台類似的，Android 應用也需要進行數字簽名後才能發布。但和 Java ME 不同的，Android 用來簽名的數字證書並不需要經過 CA認證，這可以每年省下$400-$500的費用，iPhone 每年需交$100費用。Android 平台開發對於開發者來說是投資最小的，從長遠看也是最有發展前途的一個手機平台之一。

發布 Android 應用前，可以使用工具（如 keytool）創建一個私鑰來對應用進行數字簽名。Keytool 在 JDK 中。

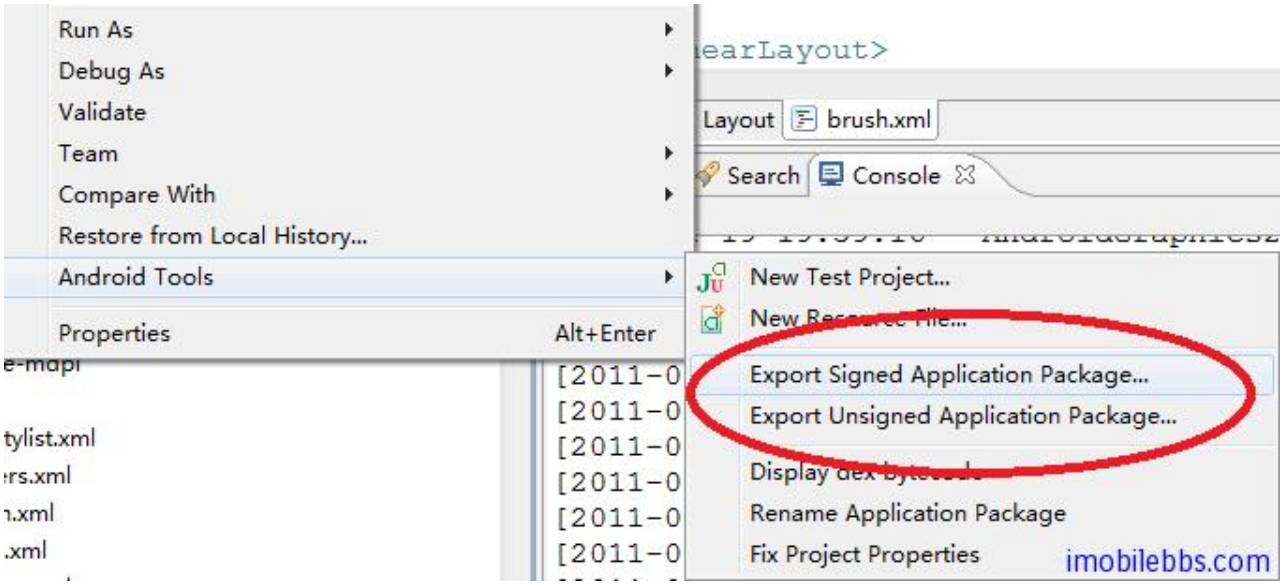
用法如下：

```
$ keytool -genkey -v -keystore my-release-key.keystore
-alias alias_name -keyalg RSA -keysize 2048 -validity 10000
```

具體含義可參見：http://developer.android.com/guide/publishing/app-signing.html

除了數字簽名之外，還可以對應用進行擾碼，如果使用 Eclipse 來開發 Android 應用，在創建的每個Android 應用中都有一個 proguard.cfg 文件，一般使用缺省設置即可。擾碼（或稱混淆）的好處是保護源碼和去除一些無用代碼可以是最後的發行包大大縮小。proguard 的詳細用法可以參見 http://proguard.sourceforge.net/

如果使用 Eclipse 來發布最後的 .apk 文件，可以通過 Android Tool 菜單嚮導來一步步來完成：



# 极客学院

jikexueyuan.com

中国最大的IT职业在线教育平台


更多信息请访问

http://wiki.jikexueyuan.com/project/android-development-tutorial/