http://dev.10086.cn/cmdn/bbs/viewthread.php?tid=18736&page=1#pid89255

Android UI 开发专题(一) 之界面设计

近期很多网友对 Android 用户界面的设计表示很感兴趣，对于 Android UI 开发自绘控件和游戏制作而言掌握好绘图基础是必不可少的。本次专题分 10 节来讲述，有关 OpenGL ES 相关的可能将放到以后再透露。本次主要涉及以下四个包的相关内容: android.content.res 资源类

android.graphics 底层图形类

android.view 显示类

android.widget 控件类

一、android.content.res.Resources

对于 Android 平台的资源类 android.content.res.Resources 可能很多网友比较陌生，一起来看看 SDK 上是怎么介绍的吧，Contains classes for accessing application resources, such as raw asset files, colors, drawables, media or other other files in the package, plus important device configuration details (orientation, input types, etc.) that affect how the application may behave.平时用到的二进制源文件 raw、颜色 colors、图形 drawables 和多媒体文件 media 的相关资源均通过该类来管理。

int getColor(int id) 对应 res/values/colors.xml

Drawable getDrawable(int id) 对应 res/drawable/

XmlResourceParser getLayout(int id) 对应 res/layout/

String getString(int id) 和 CharSequence getText(int id) 对应 res/values/strings.xml

InputStream openRawResource(int id) 对应 res/raw/

void parseBundleExtra (String tagName, AttributeSet attrs, Bundle outBundle) 对应 res/xml/

String[] getStringArray(int id) res/values/arrays.xml

float getDimension(int id) res/values/dimens.xml

二、android.graphics.Bitmap

作为位图操作类，Bitmap 提供了很多实用的方法，常用的我们总结如下:

boolean compress(Bitmap.CompressFormat format, int quality, OutputStream stream) 压缩一个 Bitmap 对象根据相关的编码、画质保存到一个 OutputStream 中。其中第一个压缩格式目前有 JPG 和 PNG

void copyPixelsFromBuffer(Buffer src) 从一个 Buffer 缓冲区复制位图像素

void copyPixelsToBuffer(Buffer dst) 将当前位图像素内容复制到一个 Buffer 缓冲区

我们看到创建位图对象 createBitmap 包含了 6 种方法在目前的 Android 2.1 SDK 中，当然他们使用的是 API Level 均为 1，所以说从 Android 1.0 SDK 开始就支持了，所以大家可以放心使用。

static Bitmap createBitmap(Bitmap src)

static Bitmap createBitmap(int[] colors, int width, int height, Bitmap.Config config)

static Bitmap createBitmap(int[] colors, int offset, int stride, int width, int height, Bitmap.Config config)

static Bitmap createBitmap(Bitmap source, int x, int y, int width, int height, Matrix m, boolean filter)

static Bitmap createBitmap(int width, int height, Bitmap.Config config)

static Bitmap createBitmap(Bitmap source, int x, int y, int width, int height)

static Bitmap createScaledBitmap(Bitmap src, int dstWidth, int dstHeight, boolean filter) //创建一个可以缩放的位图对象

final int getHeight() 获取高度

final int getWidth() 获取宽度

final boolean hasAlpha() 是否有透明通道

void setPixel(int x, int y, int color) 设置某像素的颜色

int getPixel(int x, int y) 获取某像素的颜色，android 开发网提示这里返回的 int 型是 color 的定义

三、android.graphics.BitmapFactory

作为 Bitmap 对象的 I/O 类，BitmapFactory 类提供了丰富的构造 Bitmap 对象的方法，比如从一个字节数组、文件系统、资源 ID、以及输入流中来创建一个 Bitmap 对象，下面本类的全部成员，除了 decodeFileDescriptor 外其他的重载方法都很常用。

static Bitmap decodeByteArray(byte[] data, int offset, int length) //从字节数组创建

static Bitmap decodeByteArray(byte[] data, int offset, int length, BitmapFactory.Options opts)

static Bitmap decodeFile(String pathName, BitmapFactory.Options opts) //从文件创建，路径要写全

static Bitmap decodeFile(String pathName)

static Bitmap decodeFileDescriptor(FileDescriptor fd, Rect outPadding, BitmapFactory.Options opts) //从输入流句柄创建

static Bitmap decodeFileDescriptor(FileDescriptor fd)

static Bitmap decodeResource(Resources res, int id) //从 Android 的 APK 文件资源中创建，android123 提示是从/res/的 drawable 中

static Bitmap decodeResource(Resources res, int id, BitmapFactory.Options opts)

static Bitmap decodeResourceStream(Resources res, TypedValue value, InputStream is, Rect pad, BitmapFactory.Options opts)

static Bitmap decodeStream(InputStream is) //从一个输入流中创建

static Bitmap decodeStream(InputStream is, Rect outPadding, BitmapFactory.Options opts)

四、android.graphics.Canvas

从 J2ME MIDLET 时我们就知道 Java 提供了 Canvas 类，而目前在 Android 平台中，它主要任务为管理绘制过程，The Canvas class holds the "draw" calls. To draw something, you need 4 basic components: A Bitmap to hold the pixels, a Canvas to host the draw calls (writing into the bitmap), a drawing primitive (e.g. Rect, Path, text, Bitmap), and a paint (to describe the

colors and styles for the drawing).

该类主要提供了三种构造方法，分别为构造一个空的 Canvas、从 Bitmap 中构造和从 GL 对象中创建，如下

Canvas()

Canvas(Bitmap bitmap)

Canvas(GL gl)

同时 Canvas 类的一些字段保存着重要的绘制方法定义，比如 Canvas.HAS_ALPHA_LAYER_SAVE_FLAG 保存时需要 alpha 层，对于 Canvas 类提供的方法很多，每个都很重要，下面我们一一作介绍

boolean clipPath(Path path)

boolean clipPath(Path path, Region.Op op)

boolean clipRect(float left, float top, float right, float bottom)

boolean clipRect(Rect rect)

boolean clipRect(float left, float top, float right, float bottom, Region.Op op)

boolean clipRect(Rect rect, Region.Op op)

boolean clipRect(RectF rect)

boolean clipRect(RectF rect, Region.Op op)

boolean clipRect(int left, int top, int right, int bottom)

boolean clipRegion(Region region, Region.Op op)

boolean clipRegion(Region region)

void concat(Matrix matrix)

void drawARGB(int a, int r, int g, int b)

void drawArc(RectF oval, float startAngle, float sweepAngle, boolean useCenter, Paint paint)

void drawBitmap(Bitmap bitmap, Matrix matrix, Paint paint)

void drawBitmap(int[] colors, int offset, int stride, float x, float y, int width, int height, boolean hasAlpha, Paint paint)

void drawBitmap(Bitmap bitmap, Rect src, Rect dst, Paint paint)

void drawBitmap(Bitmap bitmap, float left, float top, Paint paint)

void drawBitmap(int[] colors, int offset, int stride, int x, int y, int width, int height, boolean hasAlpha, Paint paint)

void drawBitmap(Bitmap bitmap, Rect src, RectF dst, Paint paint)

void drawBitmapMesh(Bitmap bitmap, int meshWidth, int meshHeight, float[] verts, int vertOffset, int[] colors, int colorOffset, Paint paint)

void drawCircle(float cx, float cy, float radius, Paint paint)

void drawColor(int color)

void drawColor(int color, PorterDuff.Mode mode)

void drawLine(float startX, float startY, float stopX, float stopY, Paint paint)

void drawLines(float[] pts, Paint paint)

void drawLines(float[] pts, int offset, int count, Paint paint)

void drawOval(RectF oval, Paint paint)

void drawPaint(Paint paint)

void drawPath(Path path, Paint paint)

void drawPicture(Picture picture, RectF dst)

void drawPicture(Picture picture, Rect dst)

void drawPicture(Picture picture)
void drawPoint(float x, float y, Paint paint)
void drawPoints(float[] pts, int offset, int count, Paint paint)
void drawPoints(float[] pts, Paint paint)
void drawPosText(char[] text, int index, int count, float[] pos, Paint paint)
void drawPosText(String text, float[] pos, Paint paint)
void drawRGB(int r, int g, int b)
void drawRect(RectF rect, Paint paint)
void drawRect(float left, float top, float right, float bottom, Paint paint)
void drawRect(Rect r, Paint paint)
void drawRoundRect(RectF rect, float rx, float ry, Paint paint)
void drawText(String text, int start, int end, float x, float y, Paint paint)
void drawText(char[] text, int index, int count, float x, float y, Paint paint)
void drawText(String text, float x, float y, Paint paint)
void drawText(CharSequence text, int start, int end, float x, float y, Paint paint)
void drawTextOnPath(String text, Path path, float hOffset, float vOffset, Paint paint)
void drawTextOnPath(char[] text, int index, int count, Path path, float hOffset, float vOffset, Paint paint)
void drawVertices(Canvas.VertexMode mode, int vertexCount, float[] verts, int vertOffset, float[] texs, int texOffset, int[] colors, int colorOffset, short[] indices, int indexOffset, int indexCount, Paint paint)
static void freeGlCaches()
boolean getClipBounds(Rect bounds)
final Rect getClipBounds()
int getDensity()
DrawFilter getDrawFilter()
GL getGL()
int getHeight()
void getMatrix(Matrix ctm)
final Matrix getMatrix()
int getSaveCount()
int getWidth()
boolean isOpaque()
boolean quickReject(Path path, Canvas.EdgeType type)
boolean quickReject(float left, float top, float right, float bottom, Canvas.EdgeType type)
boolean quickReject(RectF rect, Canvas.EdgeType type)
void restore()
void restoreToCount(int saveCount)
final void rotate(float degrees, float px, float py)
void rotate(float degrees)
int save()
int save(int saveFlags)
int saveLayer(float left, float top, float right, float bottom, Paint paint, int saveFlags)
int saveLayer(RectF bounds, Paint paint, int saveFlags)

int saveLayerAlpha(float left, float top, float right, float bottom, int alpha, int saveFlags)
int saveLayerAlpha(RectF bounds, int alpha, int saveFlags)
final void scale(float sx, float sy, float px, float py)
void scale(float sx, float sy)
void setBitmap(Bitmap bitmap)
void setDensity(int density)
void setDrawFilter(DrawFilter filter)
void setMatrix(Matrix matrix)
void setViewport(int width, int height)
void skew(float sx, float sy)
void translate(float dx, float dy)

五、android.graphics.Color

有关 Android 平台上表示颜色的方法有很多种，Color 提供了常规主要颜色的定义比如 Color.BLACK 和 Color.GREEN 等等，我们平时创建时主要使用以下静态方法

static int argb(int alpha, int red, int green, int blue) 构造一个包含透明对象的颜色

static int rgb(int red, int green, int blue) 构造一个标准的颜色对象

static int parseColor(String colorString) 解析一种颜色字符串的值，比如传入 Color.BLACK

本类返回的均为一个整形类似 绿色为 0xff00ff00，红色为 0xffff0000。我们将这个 DWORD 型看做 AARRGGBB，AA 代表 Aphla 透明色，后面的就不难理解，每个分成 WORD 整好为 0-255。

Android UI 开发专题(二) 之绘图基础

今天我们继续介绍 Android 平台底层绘图类的相关内容，在 Android UI 开发专题(一) 之界面设计中我们介绍了有关 Android 平台资源使用以及 Bitmap 相关类的操作，接下来将会以实例的方式给大家演示各种类的用处以及注意点。今天我们继续了解 android.graphics 包中比较重要的绘图类。

一、 android.graphics.Matrix

有关图形的变换、缩放等相关操作常用的方法有:

void reset() // 重置一个 matrix 对象。

void set(Matrix src) //复制一个源矩阵，和本类的构造方法 Matrix(Matrix src) 一样

boolean isIdentity() //返回这个矩阵是否定义(已经有意义)

void setRotate(float degrees) //指定一个角度以 0,0 为坐标进行旋转

void setRotate(float degrees, float px, float py) //指定一个角度以 px,py 为坐标进行旋转

void setScale(float sx, float sy) // 缩放

void setScale(float sx, float sy, float px, float py) //以坐标 px,py 进行缩放

void setTranslate(float dx, float dy) //平移

void setSkew (float kx, float ky, float px, float py) //以坐标 px,py 进行倾斜

void setSkew (float kx, float ky) //倾斜

二、android.graphics.NinePatch

NinePatch 是 Android 平台特有的一种非矢量图形自然拉伸处理方法，可以帮助常规的图形在拉伸时不会缩放，实例中 Android 开发网提示大家对于 Toast 的显示就是该原理，同时 SDK 中提供了一个工具名为 Draw 9-Patch，有关该工具的使用方法可以参考我们经发布的 Draw 9-Patch 使用方法介绍一文。由于该类提供了高质量支持透明的缩放方式，所以图形格式为 PNG，文件命名方式为.9.png 的后缀比如 android123.9.png。

三、android.graphics.Paint

Paint 类我们可以理解为画笔、画刷的属性定义，本类常用的方法如下:

void reset() //重置

void setARGB(int a, int r, int g, int b) 或 void setColor(int color) 均为设置 Paint 对象的颜色

void setAntiAlias(boolean aa) //是否抗锯齿，需要配合 void setFlags (Paint.ANTI_ALIAS_FLAG) 来帮助消除锯齿使其边缘更平滑。

Shader setShader(Shader shader) //设置阴影，Shader 类是一个矩阵对象，如果为 NULL 将清除阴影。

void setStyle(Paint.Style style) //设置样式，一般为 FILL 填充，或者 STROKE 凹陷效果。

void setTextSize(float textSize) //设置字体大小

void setTextAlign(Paint.Align align) //文本对齐方式

Typeface setTypeface(Typeface typeface) //设置字体，通过 Typeface 可以加载 Android 内部的字体，一般为宋体对于中文，部分 ROM 可以自己添加比如雅黑等等

void setUnderlineText(boolean underlineText) //是否设置下划线，需要撇和 void setFlags (Paint.UNDERLINE_TEXT_FLAG) 方法。

四、android.graphics.Rect

Rect 我们可以理解为矩形区域，类似的还有 Point 一个点，Rect 类除了表示一个矩形区域位置描述外，android123 提示主要可以帮助我们计算图形之间是否碰撞(包含)关系，对于 Android 游戏开发比较有用，其主要的成员 contains 包含了三种重载方法，来判断包含关系

boolean contains(int left, int top, int right, int bottom)

boolean contains(int x, int y)

boolean contains(Rect r)

五、android.graphics.Region

Region 在 Android 平台中表示一个区域和 Rect 不同的是，它表示的是一个不规则的样子，可以是椭圆、多边形等等，而 Rect 仅仅是矩形。同样 Region 的 boolean contains(int x, int y) 成员可以判断一个点是否在该区域内

六、android.graphics.Typeface

Typeface 类是帮助描述一个字体对象，在 TextView 中通过使用 setTypeface 方法来制定一个输出文本的字体，其直接构造调用成员 create 方法可以直接指定一个字体名称和样式，比如

static Typeface create(Typeface family, int style)

static Typeface create(String familyName, int style)

同时使用 isBold 和 isItalic 方法可以判断出是否包含粗体或斜体的字型。

final boolean isBold()

final boolean isItalic()

该类的创建方法还有从 apk 的资源或从一个具体的文件路径，其具体方法为

static Typeface createFromAsset(AssetManager mgr, String path)

static Typeface createFromFile(File path)

static Typeface createFromFile(String path)

有关 Android 平台的图形、图像我们在前两节中已经整理出来，下次我们将首先讲述下 NinePatch 的实例应用。

Android UI 开发专题(三) 各种 Drawable

本次我们主要讲解 Android 平台下的各种 Drawable，这里在 SDK 的 android.graphics.drawable 包下面可以看到有各种 Drawable 类多达十几种，它们到底之间有什么关系和区别呢?

一、AnimationDrawable

顾名思义该类主要表示动画的图形类，可以实现逐帧播放的效果，下面代码示例如下

1. 定义一个 cwj_animation.xml 放到 res/drawable 目录下，其中定义的属性 duration 为延时，单位为毫秒，而 oneshot 属性表示是否仅播放一次，内容为：

```
1<animation-list android:id="selected" android:oneshot="false">
2<item android:drawable="@drawable/cwj0" android:duration="30"
/>
3<item android:drawable="@drawable/cwj1" android:duration="30"
/>
4<item android:drawable="@drawable/cwj2" android:duration="30"
/>
5<item android:drawable="@drawable/cwj3" android:duration="30"
/>
6<item android:drawable="@drawable/cwj4" android:duration="30"
/>
7<item android:drawable="@drawable/cwj5" android:duration="30"
/>
8</animation-list>
```

9
10

2.在 java 中调用也很简单

ImageView img = (ImageView)findViewById(R.id.cwj_image); //首先声明一个 ImageView 对象在 xml 布局文件中

img.setBackgroundResource(R.drawable.cwj_animation); //我们刚才的 animation 定义的 xml 文件

AnimationDrawable frameAnimation = (AnimationDrawable) img.getBackground(); //构造 AnimationDrawable 对象

frameAnimation.start() //开始播放动画

3. AnimationDrawable 类还提供了一些常用的方法如下:

void stop() 停止

void addFrame(Drawable frame, int duration) 添加一帧，类似 xml 中的布局

Drawable getFrame(int index) 返回某帧的 Drawable 图形

int getNumberOfFrames() 返回总共动画帧数

boolean isOneShot() 是否仅播放一次

boolean isRunning() 是否正在播放

二、BitmapDrawable

在 Android 平台中对于缩放、变形的 Bitmap 对象由 BitmapDrawable 类表示，其构造方法也很简单，由于该类继承于 android.graphics.drawable.Drawable，相对 Drawable 而言提供了更多的有关位图的操作方法，主要的构造方法如下:

BitmapDrawable() //直接构造一个空的对象，这样方式不推荐使用，SDK 标记为 deprecated.未来可能无法使用。

BitmapDrawable(Resources res) //从资源中构造

BitmapDrawable(Bitmap bitmap) //从 Bitmap 对象直接构造，但也是不推荐，而是希望用下一种

BitmapDrawable(Resources res, Bitmap bitmap) //从 bitmap 中创建设置初始的分辨率从 res 中

BitmapDrawable(String filepath) //从具体文件路径构造，也不推荐使用，而是下一种更好

BitmapDrawable(Resources res, String filepath) //同上

BitmapDrawable(InputStream is) //从输入流中构造，同样推荐下面的方法

BitmapDrawable(Resources res, InputStream is) //同上

在 BitmapDrawable 类中相对于 Drawable 类主要新增了以下几种方法，均比较实用:

final Bitmap getBitmap() 获取一个 Bitmap 对象

int getOpacity() //获取透明度

void setAntiAlias(boolean aa) //是否抗锯齿

void setTargetDensity(Canvas canvas) //设置目标 Canvas 密度

void setTargetDensity(DisplayMetrics metrics)

三、ClipDrawable

ColorDrawable
Drawable
GradientDrawable
InsetDrawable
LayerDrawable
LevelListDrawable
NinePatchDrawable
PaintDrawable
PictureDrawable
RotateDrawable
ScaleDrawable
ShapeDrawable
StateListDrawable
TransitionDrawable

以上的类型在常见的开发一般较少出现，主要是基类构造使用，Android 内部的多个 Widget 基础控件使用了，感兴趣的网友可以查看开源 GIT 中的相关内容。

Android UI 开发专题(四) View 自绘控件

很多时候想要设计漂亮的 Android UI，使用 Android 自带的控件无法满足我们的需要就要考虑自绘控件，在 Android 界面显示类 View，可以通过继承扩展重写相关方法来实现我们的图形绘制。

首先我们需要了解下 View 类的底层实现，在 SDK 中我们可以看到 View 直接继承于 Java 的基类 Object，实现了图形绘制和按键事件 Drawable.Callback KeyEvent.Callback 的相关方法，我们自绘时主要实现其内部的 onDraw 方法，相关的界面计算可以重写 onMeasure 方法，对于相关的按键可以重载 onKeyDown、onKeyUp 以及 onTouchEvent 等，下面 android 开发网就以一个实例来表示。

```
public class cwjView extendsView
{
public cwjView(Context context)
{
this(context,null);
}
public cwjView(Context context,AttributeSet attrs)
{
this(context,attrs,0);
}
public cwjView(Context context,AttributeSet attrs,int defStyle)
{
super(context,attrs,defStyle);
//这里是本类的构造，相关初始化可以在这里添加代码
}
```

```
@Override
protectd void onDraw(Canvas canvas)
{
super(canvas);
```

//绘图的关键，可以看到已经包含了一个 canvas 句柄，可以直接通过我们前面讲到的 Canvas 类进行相关的操作，完整的例子，大家可以参考 Android SDK 中例子 Snake 贪食蛇游戏的实现。

```
}
}
```

有关 View 类的更新，我们直接通过调用 invalidate(int l,int r,int r,int b)来更新一个 Rect 矩形区域，或更新全部，同时在线程中我们使用需要调用 postInvalidate 来更新界面。

Android UI 开发(五)Bitmap 和 Canvas 实例

在 Android UI 开发专题的前五节我们讲到的东西主要是基础和理论内容，从本次 Android123 将通过实例代码来演示，本次主要是 Bitmap 和 Canvas 类的使用，根据要求缩放 Bitmap 对象并返回新的 Bitmap 对象。centerToFit 方法一共有 4 个参数，返回一个 Bitmap 类型，第一个参数为原始的位图对象，width 和 height 分别为新的宽和高，而 Context 是用来加载资源的上下文实例。

```
1Bitmap centerToFit(Bitmap bitmap,int width,int height, Context context) {
2
3       finalint bitmapWidth= bitmap.getWidth();//获取原始 bitmap 的宽度
4
5       finalint bitmapHeight= bitmap.getHeight();
6
7       if (bitmapWidth< width || bitmapHeight< height) {
8
9       int color= context.getResources().getColor(R.color.window_background);//从资源读取背景色
10
11       Bitmap centered= Bitmap.createBitmap(bitmapWidth< width ? width : bitmapWidth,
12
13       bitmapHeight< height ? height : bitmapHeight, Bitmap.Config.RGB_565);
14
15       centered.setDensity(bitmap.getDensity());
16
17       Canvas canvas=
new Canvas(centered);
18
19       canvas.drawColor(color);//先绘制背景色
20
21       canvas.drawBitmap(bitmap, (width- bitmapWidth)/
2.0f, (height- bitmapHeight)/
```

```
2.0f,null);//通过 Canvas 绘制 Bitmap
22
23      bitmap= centered;
24
25      }
26
27      return bitmap;//返回新的 bitmap
28
29      }
30
31
```

本段代码从 Android 2.1 开始将会应用在全新的 Home 主屏上，同时相关的 ImageView 的适应屏幕大小的 setScaleType(fitCenter) 方法类似，仅仅是我们制定了未来的大小。

GraphableButton 类实现 Android UI 开发

从 Android 1.6 开始，系统设置中的电池使用记录提供了一种简单的自绘 Button 按钮演示-GraphableButton 类，通过 GraphableButton 我们可以很清晰的了解到前几次 Android123 讲到的 UI 开发要点。

```
1       public class GraphableButton extends Button {//从 Button 类继承
2
3       private static finalString TAG=
"GraphableButton";
4
5       static Paint[] sPaint=
new Paint[2];//定义两种颜色
6
7       static {
8
9       sPaint[0]=
new Paint();
10
11       sPaint[0].setStyle(Paint.Style.FILL);
12
13       sPaint[0].setColor(0xFF0080FF);
14
15       sPaint[1]=
new Paint();
16
17       sPaint[1].setStyle(Paint.Style.FILL);
18
```

```
19      sPaint[1].setColor(0xFFFF6060);
20
21      }
22
23      double[] mValues;
24
25      public GraphableButton(Context context, AttributeSet attrs) {
26
27      super(context, attrs);
28
29      }
30
31      public void setValues(double[] values,double maxValue) {//设置显示范围，下文提到
32
33      mValues= values.clone();
34
35      for (int i=
0; i< values.length; i++) {
36
37      mValues/=maxValue;
38
 39       }
40
 41       }
42
43      @Override
44
45      public void onDraw(Canvas canvas) {//重写 onDraw 直接绘制
46
47      Log.i(TAG,"onDraw: w ="
+ getWidth()+
", h ="
+ getHeight());
48
49      int xmin= getPaddingLeft();
50
51      int xmax= getWidth()- getPaddingRight();
52
53      int ymin= getPaddingTop();
54
55      int ymax= getHeight()- getPaddingBottom();
56
57      int startx= xmin;
58
```

```
59      for (int i=
0; i< mValues.length; i++) {
60
61      int endx= xmin+ (int) (mValues* (xmax- xmin));
62
63      canvas.drawRect(startx, ymin, endx, ymax, sPaint);//通过 canvas 绘制范围
64
65      // 该方法原型  drawRect(floatleft, float top, floatright, float bottom, Paint paint)
66
67      startx= endx;
68
69      }
70
71      super.onDraw(canvas);
72
73      }
74
75      }
76
```

调用方法很简单，和普通的 Button 没有什么区别，这里我们仅仅多定义了 setValues 方法，Android 开发网提醒网哟注意布局文件 xml 中如何定义，在最下文

```
1     private GraphableButton mButtons;
2
3     mButtons= (GraphableButton) findViewById(R.id.button0);
4
5     mButtons.setOnClickListener(this);//设置一个按下事件监听
6
7     mButtons.setVisibility(View.INVISIBLE);//设置当前按钮不可见
8
9     mButtons.setText("android123.com 欢迎您");
10
11     mButtons.setValues(0,100);
12
13     mButtons.setVisibility(View.VISIBLE);//设置按钮可见
14
```

下面在 layout.xml 中如何写呢，这里要写上自己程序完整的 package name 才能正确被 adt 识别，相关的具体定义如下:

```
1     android:id="@+id/button7"
2
3     android:layout_width="fill_parent"
4
5     android:layout_height="0dp"
```

```
6
7       android:layout_marginLeft="4dp"
8
9       android:layout_marginRight="4dp"
10
11       android:layout_marginBottom="4dp"
12
13       android:layout_weight="1"
/>
```