Android程序设计

可绘制类

2019.6.11

isszym　sysu.edu.cn

官方文档（中文） 官方文档（英文） runoob cnblogs

主目录

Drawable类概述

参考1 参考2 参考3 参考4

- 通过Drawable类，可以用XML文件绘制图像或处理一个已知位图。
- BitmapDrawable可以增加位图属性，并在填充背景时定义重复填充方法，ScaleDrawable则对位图进行缩放，TransitionDrawable可以用于从一个位图过渡到另一个位图，一幅淡出另一幅淡入。
- RotateDrawable可以旋转位图，InsetDrawable可以在位图边留空，LayerDrawable可以把多幅位图叠放分层叠放在一起。
- 使用预设图像边沿的九宫格文件NinePatchDrawable (.9.png)还可以平滑拉伸图像的边沿。
- ClipDrawable可以剪取位图一部分，ColorDrawable可以定义颜色， ShapeDrawable可以定义四种形状。
- 设置不同级别(level)可以定义 拉伸（ScaleDrawable）、旋转（RotateDrawable）、剪切（ClipDrawable）的程度。
- AnimationDrawable可以使用多幅图像定义逐帧动画。除了利用图像文件形成可绘制对象，使用形状可绘制对象ShapeDrawable 还可以直接按照一个几何形状生成新的可绘制对象。 AnimationDrawable在动画部分给出。

BitmapDrawable

参考

如果希望对一幅图像在采用XML文件中进行修饰，包括抗锯齿、做抖动处理、做过滤操作，以及在图像小于应用背景时进行各种平铺操作，例如：重复（repeat）、镜像（mirror）、边沿拉伸（clamp），就可以使用BitmapDrawable。

项目名：DrawableOnCanvas

```
<ImageView
    android:layout_width="match_parent"
    android:layout_height="80dp"
    android:background="@drawable/bitmap"
    android:scaleType="centerInside" />
```

ImageView

drawable/bitmap.xml

```
<?xml version="1.0" encoding="utf-8"?>
<bitmap xmlns:android="http://schemas.android.com/apk/res/android"
    android:antialias="true"
    android:src="@drawable/bk"
    android:dither="true"
    android:filter="true"
    android:gravity="center"
    android:tileMode="repeat">
</bitmap>
```

bk.png

ScaleDrawable

参考

如果希望对图像进行缩放，就可以使用ScaleDrawable。scaleHeight和scaleWidth均为缩小的百分比，取值20%表示为原来大小的80%。只有在Java程序中设置Level值后该Drawable才有作用，Level值越大图像越大。

```
<ImageView
    android:id="@+id/scaleView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/scale" />
```

girl.png

scale.xml

drawable/scale.xml

```
<?xml version="1.0" encoding="utf-8"?>
<scale xmlns:android="http://schemas.android.com/apk/res/android"
    android:drawable="@drawable/girl"
    android:scaleGravity="top|right"
    android:scaleHeight="30%"
    android:scaleWidth="30%" />
```

```
ImageView imgScale=(ImageView) findViewById(R.id.scaleView);
final Drawable scaleDrawable = imgScale.getDrawable();
scaleDrawable.setLevel(1);  //值越大图形越大（0~10000）
```

TransitionDrawable

参考

TransitionDrawable可以用于从一幅图像过渡到另一幅图像。

```
<ImageView
    android:id="@+id/transView"
    android:layout_width="80dp"
    android:layout_height="80dp"
    android:src="@drawable/transition"
 />
```

mipmap/ic_bg_meizi1.jpg

meizi1逐渐变淡

meizi2逐渐显现

mipmap/ic_bg_meizi2.jpg

drawable/transition.xml

```
<?xml version="1.0" encoding="utf-8"?>
<transition xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:drawable="@mipmap/ic_bg_meizi1"/>
    <item android:drawable="@mipmap/ic_bg_meizi2"/>
</transition>
```

```
ImageView transView = (ImageView) findViewById(R.id.transView);
TransitionDrawable td = (TransitionDrawable) transView.getDrawable();
td.startTransition(5000);
```

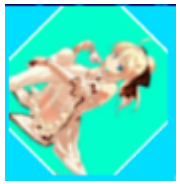
RotateDrawable

参考

如果希望对一幅图进行旋转操作，就可以使用RotateDrawable。

```
<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true"
    android:src="@drawable/rotate" />
```

girl.png



rotate.xml

drawable/rotate.xml

```
<?xml version="1.0" encoding="utf-8"?>
<rotate xmlns:android="http://schemas.android.com/apk/res/android"
        android:drawable="@drawable/girl"
        android:visible="true"
        android:fromDegrees="45"
        android:toDegrees="90"
        android:pivotX="50%"
        android:pivotY="50%">
</rotate>
```

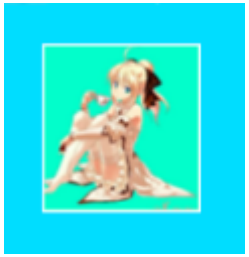
通过改变Level值可以使旋转角度从45度变为90度，改变方法类似ScaleDrawable。

InsetDrawable

参考

采用InsetDrawable可以把图像嵌入到一个View的背景中，也就是在图像周围加入一些空白。

```
<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true"
    android:src="@drawable/inset" />
```



inset.xml

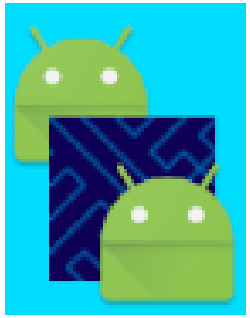
drawable/inset.xml

```
<?xml version="1.0" encoding="utf-8"?>
<inset xmlns:android="http://schemas.android.com/apk/res/android"
    android:drawable="@drawable/girl"
    android:insetBottom="20dp"
    android:insetLeft="20dp"
    android:insetRight="20dp"
    android:insetTop="20dp">
</inset>
```

LayerDrawable

参考

LayerDrawable可以把多张图片叠在一起当作一张图片使用。



```
<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/layer" />
```

drawable/layer.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
   <item>
      <bitmap android:src="@mipmap/ic_launcher"
         android:gravity="center" />
   </item>
   <item android:top="60dp" android:left="30dp">
      <bitmap android:src="@drawable/bk2"
         android:gravity="center" />
   </item>
   <item android:top="80dp" android:left="50dp">
      <bitmap android:src="@mipmap/ic_launcher"
         android:gravity="center" />
   </item>
</layer-list>
```

多变的layer-list

```
<!-- 定义一个拖动条，并改变轨道外观 -->
<!-- LayerDrawable -->
<SeekBar
    android:layout_width="200dp"
    android:layout_height="wrap_content"
    android:max="100"
    android:progressDrawable="@drawable/layer_bar" />
```


ok.gif grow.gif

drawable/layer_bar.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <!-- 定义轨道的背景 -->
    <item android:id="@android:id/background"
        android:drawable="@drawable/grow" />
    <!-- 定义轨道上已完成部分的外观-->
    <item android:id="@android:id/progress"
        android:drawable="@drawable/ok" />
</layer-list>
```

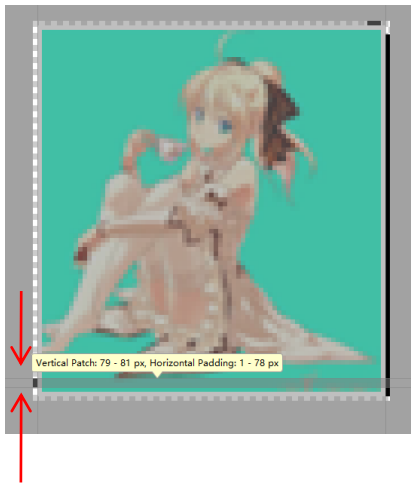
* background和progress也可以使用Shape和加了Clip的Shape，具体见安卓程序设计（一）的“SeekBar”和“ProgressBar”。

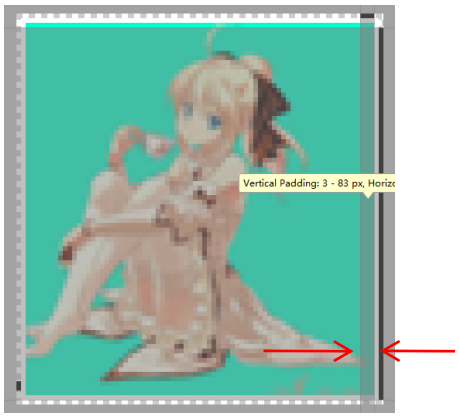
NinePatchDrawable

参考

NinePatchDrawable使用了一幅NinePatch图像（.9.png），这种图像要定义拉伸时填充的内容。可以执行SDK/tools/draw9patch.bat或在Android Studio中点击相应的图进行定义。

vertical patch





horizontal patch

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@drawable/girl1"
    android:text="9Pitch" />
```

ClipDrawable

参考

```
<ImageView
    android:id="@+id/image"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:scaleType="fitStart"
    android:src="@drawable/clip" />
```

drawable/clip.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<clip xmlns:android="http://schemas.android.com/apk/res/android"
   android:drawable="@drawable/shuangta"
   android:clipOrientation="horizontal"
   android:gravity="center">
</clip>
```

周期性修改level

```
ImageView imageview = (ImageView) findViewById(R.id.image);
final ClipDrawable clipDrawable = (ClipDrawable)imageview.getDrawable();
final Handler handler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
        if (msg.what == 0x123) {
            clipDrawable.setLevel(clipDrawable.getLevel() + 500);
        }
    }
};
final Timer timer = new Timer();
timer.schedule(new TimerTask() {
    @Override
    public void run() {
        Message msg = new Message();
        msg.what = 0x123;
        handler.sendMessage(msg);
        if (clipDrawable.getLevel() >= 10000) {
            timer.cancel();
        }
    }
}, 0, 300);
```

StateListDrawable

drawable/state.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <!-- 指定获得焦点时的颜色 -->
    <item android:state_focused="true"
          android:color="#f44"/>
    <!-- 指定失去焦点时的颜色 -->
    <item android:state_focused="false"
          android:color="#00f"/>
</selector>
```

```
<!-- 使用StateListDrawable资源 -->
<EditText
    android:layout_width="100dp"
    android:layout_height="wrap_content"
    android:textColor="@drawable/state" />
<EditText
    android:layout_width="100dp"
    android:layout_height="wrap_content"
    android:textColor="@drawable/state" />
```

ColorDrawable

参考

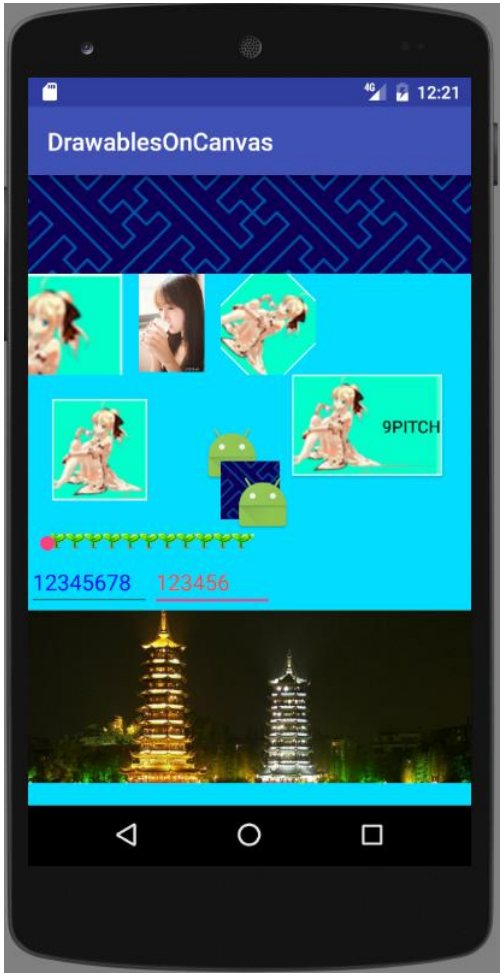
主要用于定义颜色。因为可以直接使用颜色资源等其他定义方法，这个方式很少使用。在Java程序定义：

```
ColorDrawable drawable = new ColorDrawable(0xffff2200);
txtShow.setBackground(drawable);
```

在XML文件中定义：

```
<?xml version="1.0" encoding="utf-8"?>
<color
    xmlns:android="…"
    android:color="#FF0000"
/>
```

源码例子见DrawablesOnCanvas



ShapeDrawable

参考

- ShapeDrawable可以用XML文件给出正方形、椭圆、线条和环形的描述，并得到相应的图形。这些图形的应用场所和位图一样。
- 对于ShapeDrawable的形状，还可以在XML文件中定义圆角、渐变、内边距、尺寸、内部填充色，以及边界色、边界虚线、边界粗细。

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape=["rectangle"|"oval"|"line"|"ring"]
    android:innerRadius="16dp"…> //环形的四个属性
    <corners        … />        //圆角
    <gradient       … />        //渐变方式：线性渐变(默认)/放射渐变/扫描式渐变
    <padding        … />        //内边距
    <size           … />        //尺寸
    <solid          … />        //内部填充色
    <stroke         … />        //边界色、虚线、粗细
</shape>
```

- **Shape的具体格式：**

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape= ["rectangle"|"oval"|"line"|"ring"] //矩形（默认）/椭圆形/直线形/环形
    android:innerRadius="dimension"      // 内环半径
    android:innerRadiusRatio="float"     // 内环半径相对于环的宽度的比例
    android:thickness="dimension"        // 环的厚度
    android:thicknessRatio="float"       // 环的厚度相对于环的宽度的比例
    android:useLevel="boolean">          //如果当做是LevelListDrawable使用时值为true，否则为false.
    <corners
        android:radius="dimension"              //全部的圆角半径
        android:bottomLeftRadius="dimension"    //左下角的圆角半径
        android:bottomRightRadius="dimension"   //右下角的圆角半径
        android:topLeftRadius="dimension"       //左上角的圆角半径
        android:topRightRadius="dimension"      //右上角的圆角半径
    />
    <gradient   //定义圆角
        android:type=["linear"|"radial"|"sweep"] //线性渐变（默认）/放射渐变/扫描式渐变
        android:angle="integer"           // 渐变角度，必须为45的倍数，0为从左到右，90为从上到下
        android:startColor="color"        // 渐变开始点的颜色
        android:centerColor="color"       // 渐变中间点的颜色，在开始与结束点之间
        android:endColor="color"          // 渐变结束点的颜色
        android:centerX="float"           // 渐变中心X的相当位置，范围为0～1
        android:centerY="float"           // 渐变中心Y的相当位置，范围为0～1
        android:gradientRadius="float"    // 渐变的半径，只有当渐变类型为radial时才能使用
        android:useLevel="boolean" />     // 使用LevelListDrawable时就要设置为true，
                                          //                设为false时才有渐变效果
```

```
    <padding                                    //内部边距
        android:bottom="dimension"
        android:left="dimension"
        android:right="dimension"
        android:top="dimension" />
    <size                                       //尺寸
        android:width="dimension"
        android:height="dimension" />
    <solid
        android:color="color"                   //内部填充颜色
     />
    <stroke    //边框
        android:width="dimension"               //边框宽度
        android:color="color"                   //边框颜色
        android:dashGap="dimension"             //边框的虚线间隔，0表示实线
        android:dashWidth="dimension" //边框的虚线线段长度
    />
</shape>
```

* innerRadiusRatio是指内环半径相对于环的宽度的比例，比如，环的宽度为50，比例为2.5，那么内环半径为20。

- 圆角矩形和扫描式渐变

drawable/rectangle.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle"
    android:useLevel="false">

    <corners
        android:bottomLeftRadius="10dp"
        android:bottomRightRadius="10dp"
        android:topLeftRadius="10dp"
        android:topRightRadius="10dp" />

    <gradient
        android:centerColor="@android:color/holo_blue_dark"
        android:endColor="@android:color/holo_blue_bright"
        android:startColor="@android:color/holo_green_dark"
        android:type="sweep"
        android:useLevel="false" />

    <size
        android:width="60dp"
        android:height="60dp" />
</shape>
```

EditView
的背景

- 圆形和线性渐变

drawable/circle.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="oval"
    android:useLevel="false" >

    <gradient
        android:type="linear"
        android:angle="45"
        android:startColor="@android:color/holo_green_dark"
        android:centerColor="@android:color/holo_blue_dark"
        android:endColor="@android:color/holo_red_dark"
        android:useLevel="false" />

    <size android:width="60dp" android:height="60dp" />

    <stroke android:width="1dp"
        android:color="@android:color/white" />

</shape>
```

TextView
的背景

- 环形和放射型渐变

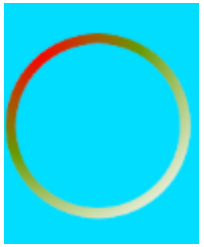
drawable/ring.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="ring"
    android:useLevel="false"
    android:innerRadius="40dp"
    android:thickness="5dp">

    <gradient android:type="radial"
        android:gradientRadius="300"
        android:centerY="0.1"
        android:centerX="0.2"
        android:startColor="#FF00"
        android:centerColor="@android:color/holo_green_dark"
        android:endColor="@android:color/white" />

    <size android:width="90dp"
        android:height="90dp" />

</shape>
```



ImageView 的背景

- 虚线

drawable/line.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="line" >

    <size android:width="60dp"
        android:height="60dp" />

    <stroke android:width="4dp"
        android:color="@android:color/holo_purple"
        android:dashWidth="10dp"
        android:dashGap="5dp" />
</shape>
```

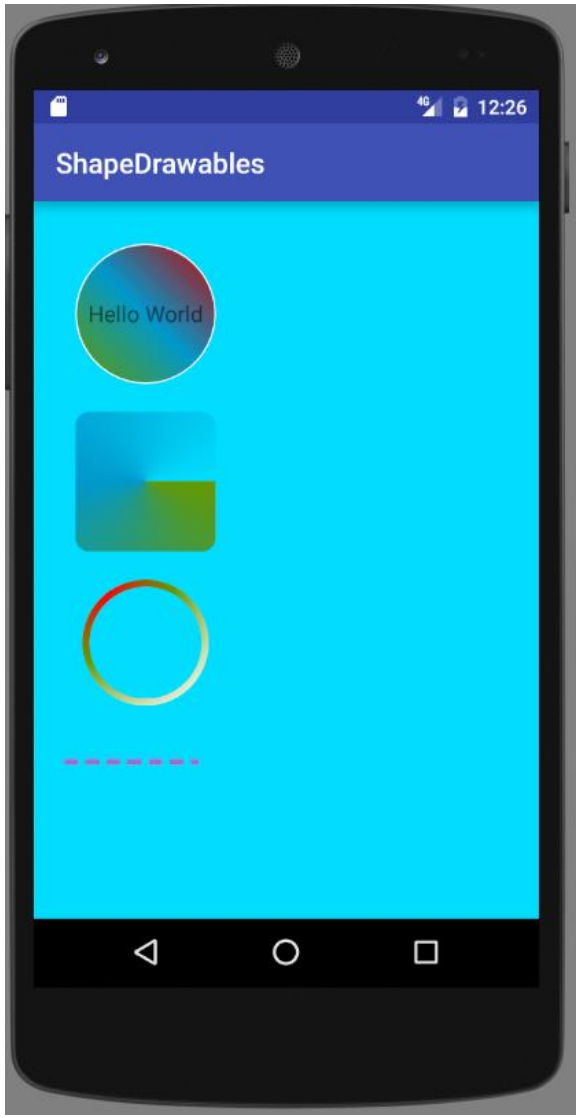
ImageView的背景

```
<ImageView
    android:layerType="software"
    android:layout_width="100dp"
    android:layout_height="wrap_content"
    android:background="@drawable/line" />
```

* 虚线需要去除硬件加速才能显示出来

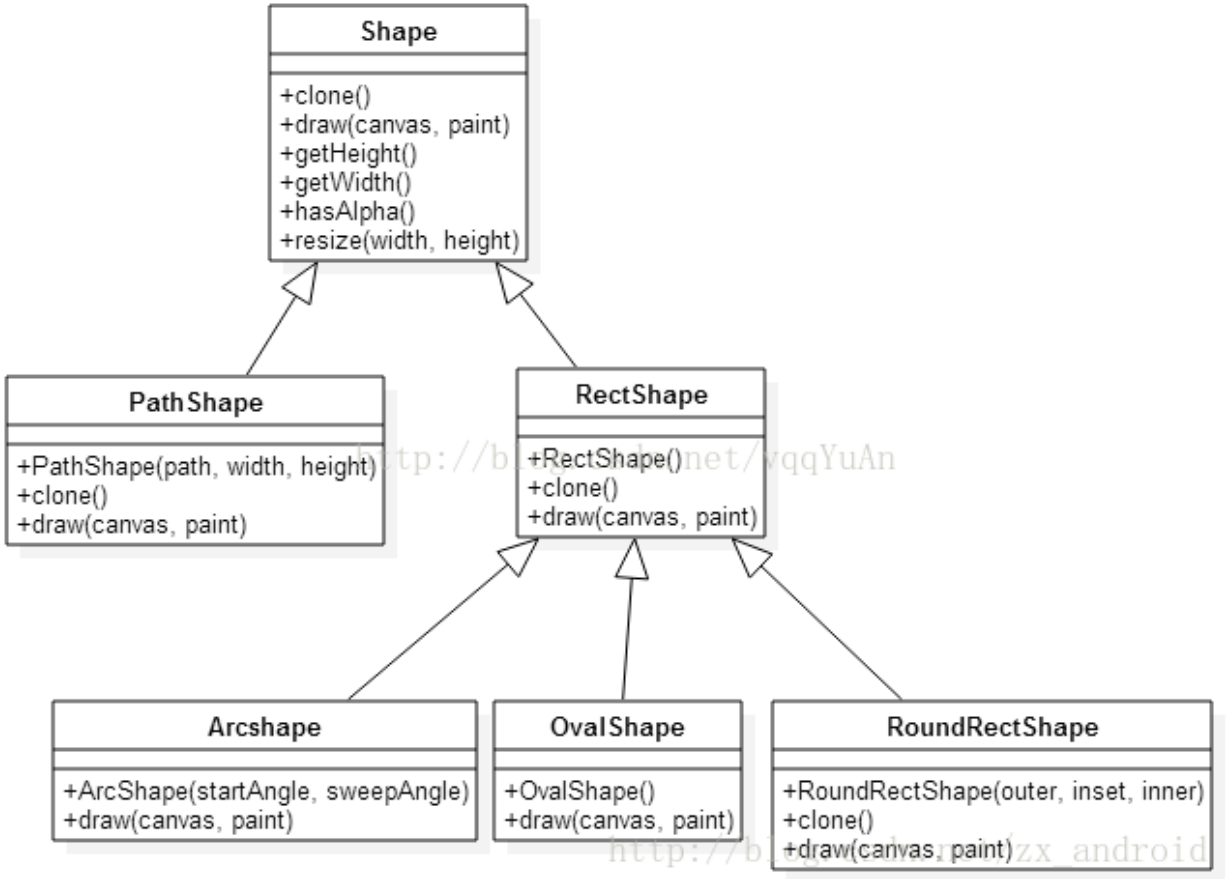
activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="20dp"
    android:background="@android:color/holo_blue_bright"
    android:orientation="vertical">
    <TextView
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:layout_margin="10dp"
        android:background="@drawable/circle"
        android:gravity="center"
        android:text="Hello World"
        android:textSize="16sp" />
    <EditText
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:layout_margin="10dp"
        android:background="@drawable/rectangle"
        android:padding="10dp" />
    <ImageView
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:background="@drawable/ring" />
    <ImageView
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:background="@drawable/line"
        android:layerType="software" />
</LinearLayout>
```



- **Shape类的Java编程**

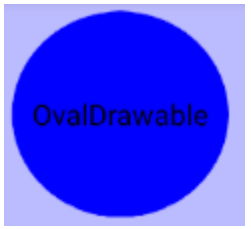
Shape类可以直接采用Java产生drawable形状。

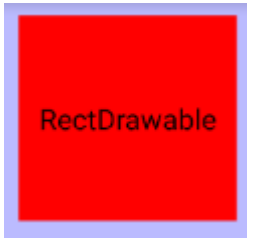


例子

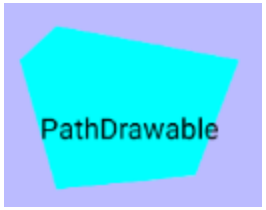
- **Shape类在Java中作为背景的编程**

```
//椭圆形形状
OvalShape ovalShape = new OvalShape();
ShapeDrawable drawable1 = new ShapeDrawable(ovalShape);
drawable1.getPaint().setColor(Color.BLUE);
drawable1.getPaint().setStyle(Paint.Style.FILL);
findViewById(R.id.textView1).setBackground(drawable1);
//矩形形状
RectShape rectShape = new RectShape();
ShapeDrawable drawable2 = new ShapeDrawable(rectShape);
drawable2.getPaint().setColor(Color.RED);
drawable2.getPaint().setStyle(Paint.Style.FILL);
findViewById(R.id.textView2).setBackground(drawable2);
//一个继承自ShapeDrawable更为通用、可以直接使用的形状
PaintDrawable drawable3 = new PaintDrawable(Color.GREEN);
drawable3.setCornerRadius(30);
findViewById(R.id.textView3).setBackground(drawable3);
//扇形、扇面形状（顺时针,开始角度45，  扫描的弧度跨度270）
ArcShape arcShape = new ArcShape(45, 270);
ShapeDrawable drawable4 = new ShapeDrawable(arcShape);
drawable4.getPaint().setColor(Color.YELLOW);
drawable4.getPaint().setStyle(Paint.Style.FILL);
findViewById(R.id.textView4).setBackground(drawable4);
```



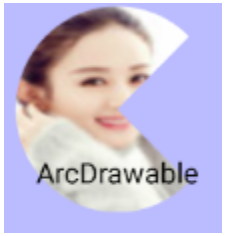


```
//路径
Path path = new Path();
path.moveTo(50, 0);
path.lineTo(0, 50);
path.lineTo(50, 240);
path.lineTo(240, 220);
path.lineTo(300, 50);
path.lineTo(50, 0);
PathShape pathShape = new PathShape(path, 300, 300);
ShapeDrawable drawable5 = new ShapeDrawable(pathShape);
drawable5.getPaint().setColor(Color.CYAN);
drawable5.getPaint().setStyle(Paint.Style.FILL);
findViewById(R.id.textView5).setBackground(drawable5);
```

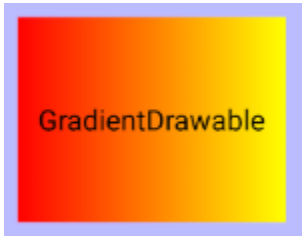



```
//圆角矩形形状（无内矩形）圆角半径（左上、右上、右下、左下）
float[] outerRadii = {20, 20, 40, 40, 60, 60, 80, 80}; //外矩形
RectF inset = new RectF(100, 100, 200, 200);           //内矩形
float[] innerRadii = {20, 20, 20, 20, 20, 20, 20, 20};
RoundRectShape roundRectShape = new RoundRectShape(outerRadii, null, innerRadii);
ShapeDrawable drawable6 = new ShapeDrawable(roundRectShape);
drawable6.getPaint().setColor(Color.MAGENTA);
drawable6.getPaint().setAntiAlias(true);
drawable6.getPaint().setStrokeWidth(1);
drawable6.getPaint().setStyle(Paint.Style.STROKE);//描边
findViewById(R.id.textView6).setBackground(drawable6);
```

```
// Shader
ArcShape arcShape2 = new ArcShape(45, 270);
ShapeDrawable drawable7 = new ShapeDrawable(arcShape2);
drawable7.getPaint().setColor(Color.YELLOW);
drawable7.getPaint().setStyle(Paint.Style.FILL);
Bitmap bitmap = ((BitmapDrawable)
getResources().getDrawable(R.drawable.zly)).getBitmap();
BitmapShader bitmapShader = new BitmapShader(bitmap,
                Shader.TileMode.REPEAT, Shader.TileMode.REPEAT);
Matrix matrix = new Matrix();
matrix.setTranslate(-480,-20);
matrix.preScale(400.00f / bitmap.getWidth(),
            400.00f / bitmap.getHeight());//view:w=600,h=600
bitmapShader.setLocalMatrix(matrix);
drawable7.getPaint().setShader(bitmapShader);
findViewById(R.id.textView7).setBackground(drawable7);
```




```
// 渐变
GradientDrawable gradientDrawable =
      new GradientDrawable(GradientDrawable.Orientation.LEFT_RIGHT,
      new int[]{Color.RED,Color.YELLOW});
findViewById(R.id.textView8).setBackground(gradientDrawable);
```



- 在自定义View中直接创建ShapeDrawable对象

```
public class MyView extends View {
    private ShapeDrawable mDrawable;
    public MyView(Context context, AttributeSet attrs){
        super(context, attrs);
    }
    @Override
    protected void onDraw(Canvas canvas){
        int x = 2;
        int y = 2;
        int width = 200;
        int height = 200;

        mDrawable = new ShapeDrawable(new OvalShape());
        mDrawable.getPaint().setColor(0xff74AC23);
        mDrawable.setBounds(x, y, x + width, y + height);
        mDrawable.draw(canvas);
    }
}
```