[**[置顶] Android利用canvas画各种图形(点、直线、弧、圆、椭圆、文字、矩形、多边形、曲线、圆角矩形)**](http://blog.csdn.net/rhljiayou/article/details/7212620)

标签： [图形](http://www.csdn.net/tag/%e5%9b%be%e5%bd%a2)[android](http://www.csdn.net/tag/android)[path](http://www.csdn.net/tag/path)[float](http://www.csdn.net/tag/float)[class](http://www.csdn.net/tag/class)

2012-01-20 20:02 206082人阅读 [评论](http://blog.csdn.net/rhljiayou/article/details/7212620/#comments)(75) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/rhljiayou/article/details/7212620/#report)

分类：

android*（12）*

版权声明：本文为博主原创文章，未经博主允许不得转载。

目录[(?)[+]](http://blog.csdn.net/rhljiayou/article/details/7212620/)

1、首先说一下canvas类：

**Class Overview**

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).

**这个类相当于一个画布，你可以在里面画很多东西；**

**我们可以把这个Canvas理解成系统提供给我们的一块内存区域(但实际上它只是一套画图的API，真正的内存是下面的Bitmap)，而且它还提供了一整套对这个内存区域进行操作的方法，所有的这些操作都是画图API。也就是说在这种方式下我们已经能一笔一划或者使用Graphic来画我们所需要的东西了，要画什么要显示什么都由我们自己控制。**

**这种方式根据环境还分为两种：一种就是使用普通View的canvas画图，还有一种就是使用专门的SurfaceView的canvas来画图。两种的主要是区别就是可以在SurfaceView中定义一个专门的线程来完成画图工作，应用程序不需要等待View的刷图，提高性能。前面一种适合处理量比较小，帧率比较小的动画，比如说象棋游戏之类的；而后一种主要用在游戏，高品质动画方面的画图。**

下面是Canvas类常用的方法：

**drawRect(RectF rect, Paint paint) //绘制区域，参数一为RectF一个区域**

**drawPath(Path path, Paint paint) //绘制一个路径，参数一为Path路径对象**

**drawBitmap(Bitmap bitmap, Rect src, Rect dst, Paint paint)  //贴图，参数一就是我们常规的Bitmap对象，参数二是源区域(这里是bitmap)，参数三是目标区域(应该在canvas的位置和大小)，参数四是Paint画刷对象，因为用到了缩放和拉伸的可能，当原始Rect不等于目标Rect时性能将会有大幅损失。**

**drawLine(float startX, float startY, float stopX, float stopY, Paintpaint) //画线，参数一起始点的x轴位置，参数二起始点的y轴位置，参数三终点的x轴水平位置，参数四y轴垂直位置，最后一个参数为Paint 画刷对象。**

**drawPoint(float x, float y, Paint paint) //画点，参数一水平x轴，参数二垂直y轴，第三个参数为Paint对象。**

**drawText(String text, float x, floaty, Paint paint)  //渲染文本，Canvas类除了上面的还可以描绘文字，参数一是String类型的文本，参数二x轴，参数三y轴，参数四是Paint对象。**

[**drawOval**](file:///E:\android\android\android-sdk-windows\docs\reference\android\graphics\Canvas.html#drawOval%28android.graphics.RectF,%20android.graphics.Paint%29)**(**[**RectF**](file:///E:\android\android\android-sdk-windows\docs\reference\android\graphics\RectF.html) **oval,** [**Paint**](file:///E:\android\android\android-sdk-windows\docs\reference\android\graphics\Paint.html) **paint)//画椭圆，参数一是扫描区域，参数二为paint对象；**

[**drawCircle**](file:///E:\android\android\android-sdk-windows\docs\reference\android\graphics\Canvas.html#drawCircle%28float,%20float,%20float,%20android.graphics.Paint%29)**(float cx, float cy, float radius,**[**Paint**](file:///E:\android\android\android-sdk-windows\docs\reference\android\graphics\Paint.html) **paint)// 绘制圆，参数一是中心点的x轴，参数二是中心点的y轴，参数三是半径，参数四是paint对象；**

[**drawArc**](file:///E:\android\android\android-sdk-windows\docs\reference\android\graphics\Canvas.html#drawArc%28android.graphics.RectF,%20float,%20float,%20boolean,%20android.graphics.Paint%29)**(**[**RectF**](file:///E:\android\android\android-sdk-windows\docs\reference\android\graphics\RectF.html) **oval, float startAngle, float sweepAngle, boolean useCenter,** [**Paint**](file:///E:\android\android\android-sdk-windows\docs\reference\android\graphics\Paint.html) **paint)//画弧，**

**参数一是RectF对象，一个矩形区域椭圆形的界限用于定义在形状、大小、电弧，参数二是起始角(度)在电弧的开始，**

**参数三扫描角(度)开始顺时针测量的，参数四是如果这是真的话,包括椭圆中心的电弧,并关闭它,如果它是假这将是一个弧线,参数五是Paint对象；**

还要理解一个paint类：

**Class Overview**

The Paint class holds the style and color information about how to draw geometries, text and bitmaps.

paint类拥有风格和颜色信息如何绘制几何学,文本和位图。

Paint 代表了Canvas上的画笔、画刷、颜料等等；

Paint类常用方法:

**setARGB(int a, int r, int g, int b) // 设置 Paint对象颜色，参数一为alpha透明值**

**setAlpha(int a) // 设置alpha不透明度，范围为0~255**

**setAntiAlias(boolean aa) // 是否抗锯齿**

**setColor(int color)  // 设置颜色，这里**[**Android**](http://lib.csdn.net/base/android)**内部定义的有Color类包含了一些常见颜色定义**

**setTextScaleX(float scaleX)  // 设置文本缩放倍数，1.0f为原始**

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

**setUnderlineText(booleanunderlineText)  // 设置下划线**

2、直接看案例

看一下效果图：

在此案例中我们用到的是自定义view类;

CustomActivity.[**Java**](http://lib.csdn.net/base/java)

public class CustomActivity extends Activity {

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

init();

}

private void init() {

LinearLayout layout=(LinearLayout) findViewById(R.id.root);

final DrawView view=new DrawView(this);

view.setMinimumHeight(500);

view.setMinimumWidth(300);

//通知view组件重绘

view.invalidate();

layout.addView(view);

}

}

**重要的类自定义View组件要重写View组件的onDraw(Canvase)方法，接下来是在该 Canvas上绘制大量的几何图形，点、直线、弧、圆、椭圆、文字、矩形、多边形、曲线、圆角矩形，等各种形状！**

**DrawView.java**

public class DrawView extends View {

public DrawView(Context context) {

super(context);

}

@Override

protected void onDraw(Canvas canvas) {

super.onDraw(canvas);

/\*

\* 方法 说明 drawRect 绘制矩形 drawCircle 绘制圆形 drawOval 绘制椭圆 drawPath 绘制任意多边形

\* drawLine 绘制直线 drawPoin 绘制点

\*/

// 创建画笔

Paint p = new Paint();

p.setColor(Color.RED);// 设置红色

canvas.drawText("画圆：", 10, 20, p);// 画文本

canvas.drawCircle(60, 20, 10, p);// 小圆

p.setAntiAlias(true);// 设置画笔的锯齿效果。 true是去除，大家一看效果就明白了

canvas.drawCircle(120, 20, 20, p);// 大圆

canvas.drawText("画线及弧线：", 10, 60, p);

p.setColor(Color.GREEN);// 设置绿色

canvas.drawLine(60, 40, 100, 40, p);// 画线

canvas.drawLine(110, 40, 190, 80, p);// 斜线

//画笑脸弧线

p.setStyle(Paint.Style.STROKE);//设置空心

RectF oval1=new RectF(150,20,180,40);

canvas.drawArc(oval1, 180, 180, false, p);//小弧形

oval1.set(190, 20, 220, 40);

canvas.drawArc(oval1, 180, 180, false, p);//小弧形

oval1.set(160, 30, 210, 60);

canvas.drawArc(oval1, 0, 180, false, p);//小弧形

canvas.drawText("画矩形：", 10, 80, p);

p.setColor(Color.GRAY);// 设置灰色

p.setStyle(Paint.Style.FILL);//设置填满

canvas.drawRect(60, 60, 80, 80, p);// 正方形

canvas.drawRect(60, 90, 160, 100, p);// 长方形

canvas.drawText("画扇形和椭圆:", 10, 120, p);

/\* 设置渐变色 这个正方形的颜色是改变的 \*/

Shader mShader = new LinearGradient(0, 0, 100, 100,

new int[] { Color.RED, Color.GREEN, Color.BLUE, Color.YELLOW,

Color.LTGRAY }, null, Shader.TileMode.REPEAT); // 一个材质,打造出一个线性梯度沿著一条线。

p.setShader(mShader);

// p.setColor(Color.BLUE);

RectF oval2 = new RectF(60, 100, 200, 240);// 设置个新的长方形，扫描测量

canvas.drawArc(oval2, 200, 130, true, p);

// 画弧，第一个参数是RectF：该类是第二个参数是角度的开始，第三个参数是多少度，第四个参数是真的时候画扇形，是假的时候画弧线

//画椭圆，把oval改一下

oval2.set(210,100,250,130);

canvas.drawOval(oval2, p);

canvas.drawText("画三角形：", 10, 200, p);

// 绘制这个三角形,你可以绘制任意多边形

Path path = new Path();

path.moveTo(80, 200);// 此点为多边形的起点

path.lineTo(120, 250);

path.lineTo(80, 250);

path.close(); // 使这些点构成封闭的多边形

canvas.drawPath(path, p);

// 你可以绘制很多任意多边形，比如下面画六连形

p.reset();//重置

p.setColor(Color.LTGRAY);

p.setStyle(Paint.Style.STROKE);//设置空心

Path path1=new Path();

path1.moveTo(180, 200);

path1.lineTo(200, 200);

path1.lineTo(210, 210);

path1.lineTo(200, 220);

path1.lineTo(180, 220);

path1.lineTo(170, 210);

path1.close();//封闭

canvas.drawPath(path1, p);

/\*

\* Path类封装复合(多轮廓几何图形的路径

\* 由直线段\*、二次曲线,和三次方曲线，也可画以油画。drawPath(路径、油漆),要么已填充的或抚摸

\* (基于油漆的风格),或者可以用于剪断或画画的文本在路径。

\*/

//画圆角矩形

p.setStyle(Paint.Style.FILL);//充满

p.setColor(Color.LTGRAY);

p.setAntiAlias(true);// 设置画笔的锯齿效果

canvas.drawText("画圆角矩形:", 10, 260, p);

RectF oval3 = new RectF(80, 260, 200, 300);// 设置个新的长方形

canvas.drawRoundRect(oval3, 20, 15, p);//第二个参数是x半径，第三个参数是y半径

//画贝塞尔曲线

canvas.drawText("画贝塞尔曲线:", 10, 310, p);

p.reset();

p.setStyle(Paint.Style.STROKE);

p.setColor(Color.GREEN);

Path path2=new Path();

path2.moveTo(100, 320);//设置Path的起点

path2.quadTo(150, 310, 170, 400); //设置贝塞尔曲线的控制点坐标和终点坐标

canvas.drawPath(path2, p);//画出贝塞尔曲线

//画点

p.setStyle(Paint.Style.FILL);

canvas.drawText("画点：", 10, 390, p);

canvas.drawPoint(60, 390, p);//画一个点

canvas.drawPoints(new float[]{60,400,65,400,70,400}, p);//画多个点

//画图片，就是贴图

Bitmap bitmap = BitmapFactory.decodeResource(getResources(), R.drawable.ic\_launcher);

canvas.drawBitmap(bitmap, 250,360, p);

}

}