**一、画线段**

**1.1 lineTo(float x, float y)**

@Override

protected void onDraw(Canvas canvas) {

super.onDraw(canvas);

mPaint.setColor(Color.RED);

mPaint.setStyle(Paint.Style.STROKE);

mPaint.setStrokeWidth(5);

// 实例化路径

mPath = new Path();

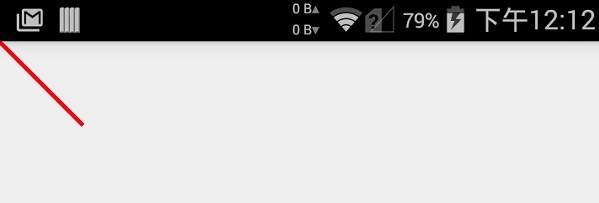
// 连接路径到点[100,100]

mPath.lineTo(100, 100);

// 绘制路径

canvas.drawPath(mPath, mPaint);

}



效果就是将起始点和（100，100）进行连接。这里因为没有设置起始点，所以默认是canvas的左上角，而这里默认的canvas和屏幕重合，所以就成了这个样子。

当然我们可以考虑多次调用lineTo方法来绘制更复杂的图形：

复制代码

@Override

protected void onDraw(Canvas canvas) {

super.onDraw(canvas);

mPaint.setColor(Color.RED);

mPaint.setStyle(Paint.Style.STROKE);

mPaint.setStrokeWidth(5);

// 实例化路径

mPath = new Path();

mPath.moveTo(100, 100);

// 连接路径到点

mPath.lineTo(300, 100);

mPath.lineTo(400, 200);

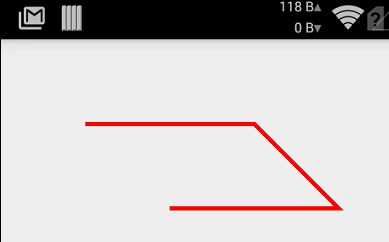
mPath.lineTo(200, 200);

// 绘制路径

canvas.drawPath(mPath, mPaint);

}

复制代码



**1.2 moveTo(float x, float y)**

我们可以通过moveTo(float x, float y) 来移动起始的坐标点，比如：

复制代码

@Override

protected void onDraw(Canvas canvas) {

super.onDraw(canvas);

mPaint.setColor(Color.RED);

mPaint.setStyle(Paint.Style.STROKE);

mPaint.setStrokeWidth(5);

// 实例化路径

mPath = new Path();

//移动点至[300,300]

mPath.moveTo(300, 300);

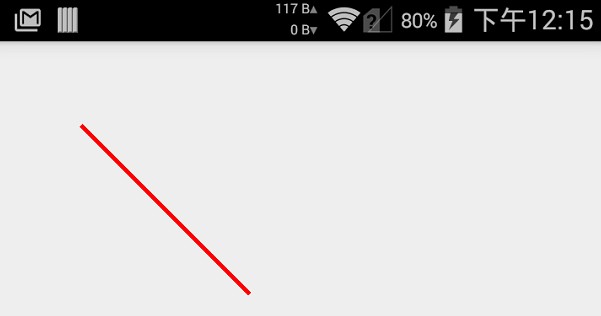
// 连接路径到点[100,100]

mPath.lineTo(100, 100);

// 绘制路径

canvas.drawPath(mPath, mPaint);

}



1.3 close()

protected void onDraw(Canvas canvas) {

super.onDraw(canvas);

mPaint.setColor(Color.RED);

mPaint.setStyle(Paint.Style.STROKE);

mPaint.setStrokeWidth(5);

// 实例化路径

mPath = new Path();

mPath.moveTo(100, 100);

// 连接路径到点

mPath.lineTo(300, 100);

mPath.lineTo(400, 200);

mPath.lineTo(200, 200);

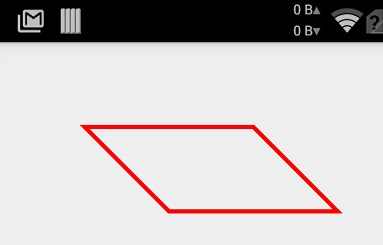
// 闭合曲线

mPath.close();

// 绘制路径

canvas.drawPath(mPath, mPaint);

}



**二、画贝赛尔曲线**

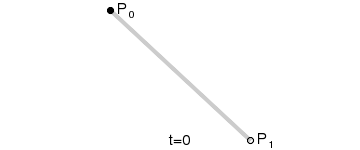
**2.1 贝赛尔曲线**

什么叫贝赛尔曲线？其实很简单，使用三个或多个点来确定的一条曲线，贝塞尔曲线在图形图像学中有相当重要的地位，Path中也提供了一些方法来给我们模拟低阶贝赛尔曲线。

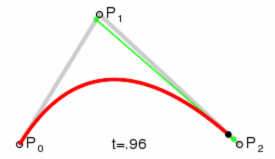
贝塞尔曲线的定义也比较简单，你只需要一个起点、一个终点和至少零个控制点则可定义一个贝赛尔曲线，当控制点为零时，只有起点和终点，此时的曲线说白了就是一条线段，我们称之为一阶贝赛尔曲线。

PS：以下图片和公式均来自维基百科和互联网

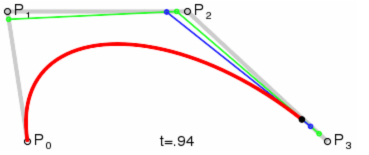
**一阶贝赛尔曲线：**



二阶贝赛尔曲线



三阶贝赛尔曲线：



**2.2 quadTo(float x1, float y1, float x2, float y2)**

二阶”贝赛尔曲线

@Override

protected void onDraw(Canvas canvas) {

super.onDraw(canvas);

mPaint.setColor(Color.RED);

mPaint.setStyle(Paint.Style.STROKE);

mPaint.setStrokeWidth(5);

// 实例化路径

mPath = new Path();

// 移动起点至[100,100]

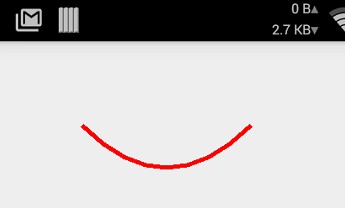
mPath.moveTo(100, 100);

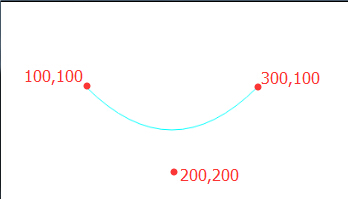
// 连接路径到点

mPath.quadTo(200, 200, 300, 100);

canvas.drawPath(mPath, mPaint);

}





2.3 cubicTo(float x1, float y1, float x2, float y2, float x3, float y3)

@Override

protected void onDraw(Canvas canvas) {

super.onDraw(canvas);

mPaint.setColor(Color.RED);

mPaint.setStyle(Paint.Style.STROKE);

mPaint.setStrokeWidth(5);

// 实例化路径

mPath = new Path();

// 移动起点至[100,100]

mPath.moveTo(100, 100);

// 连接路径到点

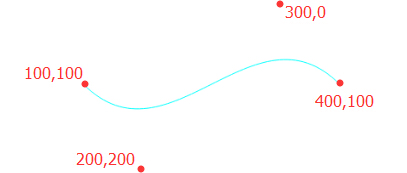
mPath.cubicTo(200, 200, 300, 0, 400, 100);

canvas.drawPath(mPath, mPaint);

}



  示意图：



**三画圆弧**

arcTo (RectF oval, float startAngle, float sweepAngle)

是一个画弧线的方法，其实说白了就是从圆或椭圆上截取一部分而已。

复制代码

@Override

protected void onDraw(Canvas canvas) {

super.onDraw(canvas);

mPaint.setColor(Color.RED);

mPaint.setStyle(Paint.Style.STROKE);

mPaint.setStrokeWidth(5);

// 实例化路径

mPath = new Path();

// 移动起点至[100,100]

mPath.moveTo(100, 100);

// 连接路径到点

RectF oval = new RectF(100, 100, 200, 200);

mPath.arcTo(oval, 0, 90);

canvas.drawPath(mPath, mPaint);

}

