安卓自定义 View 进阶：贝塞尔曲线（上）

2016-06-15 安卓应用频道

(点击上方公众号，可快速关注)

> 来源：伯乐在线专栏作者 - GcsSloop
> 链接：http://android.jobbole.com/83400/
> 点击 → 了解如何加入专栏作者

在上一篇文章 Path之基本图形 中我们了解了Path的基本使用方法，本次了解Path中非常非常非常重要的内容-贝塞尔曲线。

一.Path常用方法表

> 为了兼容性(偷懒) 本表格中去除了在API21(即安卓版本5.0)以上才添加的方法。忍不住吐槽一下，为啥看起来有些顺手就能写的重载方法要等到API21才添加上啊。宝宝此刻内心也是崩溃的。

作用	相关方法	备注
移动起点	moveTo	移动下一次操作的起点位置
设置终点	setLastPoint	重置当前path中最后一个点位置，如果在绘制之前调用，效果和moveTo相同
连接直线	lineTo	添加上一个点到当前点之间的直线到Path
闭合路径	close	连接第一个点连接到最后一个点，形成一个闭合区域
添加内容	addRect, addRoundRect, addOval, addCircle, addPath, addArc, arcTo	添加(矩形， 圆角矩形， 椭圆， 圆， 路径， 圆弧) 到当前Path (注意addArc和arcTo的区别)
是否为空	isEmpty	判断Path是否为空
是否为		

矩形	isRect	判断path是否是一个矩形
替换路径	set	用新的路径替换到当前路径所有内容
偏移路径	offset	对当前路径之前的操作进行偏移(不会影响之后的操作)
贝塞尔曲线	quadTo, cubicTo	分别为二次和三次贝塞尔曲线的方法
rXxx方法	rMoveTo, rLineTo, rQuadTo, rCubicTo	**不带r的方法是基于原点的坐标系(偏移量)，rXxx方法是基于当前点坐标系(偏移量)**
填充模式	setFillType, getFillType, isInverseFillType, toggleInverseFillType	设置,获取,判断和切换填充模式
提示方法	incReserve	提示Path还有多少个点等待加入**(这个方法貌似会让Path优化存储结构)**
布尔操作(API 19)	op	对两个Path进行布尔运算(即取交集、并集等操作)
计算边界	computeBounds	计算Path的边界
重置路径	reset, rewind	清除Path中的内容 **reset不保留内部数据结构，但会保留FillType.** **rewind会保留内部的数据结构，但不保留FillType**
矩阵操作	transform	矩阵变换

二.Path详解

上一次除了一些常用函数之外，讲解的基本上都是直线，本次需要了解其中的曲线部分,说到曲线，就不得不提大名鼎鼎的贝塞尔曲线。它的发明者是下面这个人(法国数学家PierreBézier)。

贝塞尔曲线能干什么？

贝塞尔曲线的运用是十分广泛的，可以说贝塞尔曲线奠定了计算机绘图的基础(因为它可以将任何复杂的图形用精确的数学语言进行描述)，在你不经意间就已经使用过它了。

你会使用Photoshop的话，你可能会注意到里面有一个钢笔工具，这个钢笔工具核心就是贝塞尔曲线。

你说你不会PS？ 没关系，你如果看过前面的文章或者用过2D绘图，肯定绘制过圆，圆弧，圆角矩形等这些东西。这里面的圆弧部分全部都是贝塞尔曲线的运用。

贝塞尔曲线作用十分广泛，简单举几个的栗子:

> - QQ小红点拖拽效果
> - 一些炫酷的下拉刷新控件
> - 阅读软件的翻书效果
> - 一些平滑的折线图的制作
> - 很多炫酷的动画效果
> - 如何轻松入门贝塞尔曲线？

虽然贝塞尔曲线用途非常广泛，然而目前貌似并没有适合的中文教程，能够搜索出来Android关于贝塞尔曲线的中文文章基本可以分为以下几种：

- 科普型(只是让人了解贝塞尔，并没有实质性的内容)
- 装逼型(摆出来一大堆公式，引用一堆英文原文)
- 基础型(仅仅是讲解贝塞尔曲线的两个函数用法)
- 实战型(根据实例讲解其中贝塞尔曲线的运用)

以上几种类型中比较有用的就是基础型和实战型，但两者各有不足，本文会综合两者内容，从零开始学习贝塞尔曲线。

第一步.理解贝塞尔曲线的原理

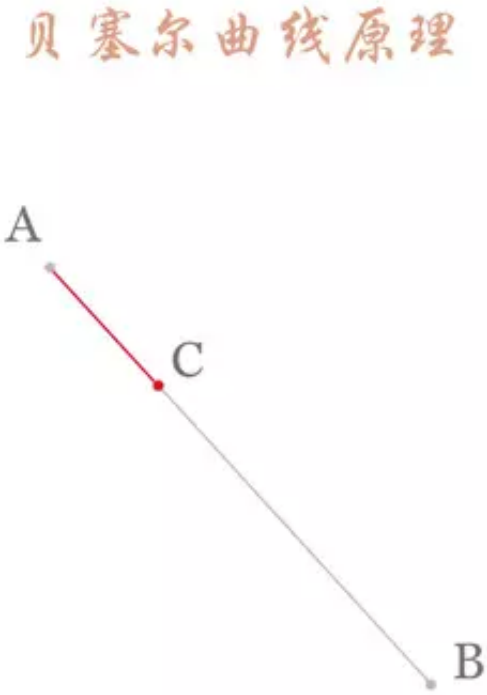
此处理解贝塞尔曲线并非是学会公式的推导过程(不是推倒(/*·ω·)/)，而是要了解贝塞尔曲线是如何生成的。 贝塞尔曲线是用一系列点来控制曲线状态的，我将这些点简单分为两类：

类型	作用
数据点	确定曲线的起始和结束位置
控制点	确定曲线的弯曲程度

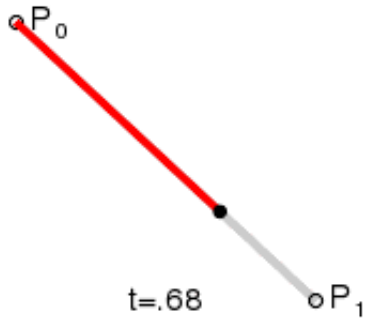
> 此处暂时仅作了解概念，接下来就会讲解其中详细的含义。

一阶曲线原理：

一阶曲线是没有控制点的，仅有两个数据点(A 和 B)，最终效果一个线段。



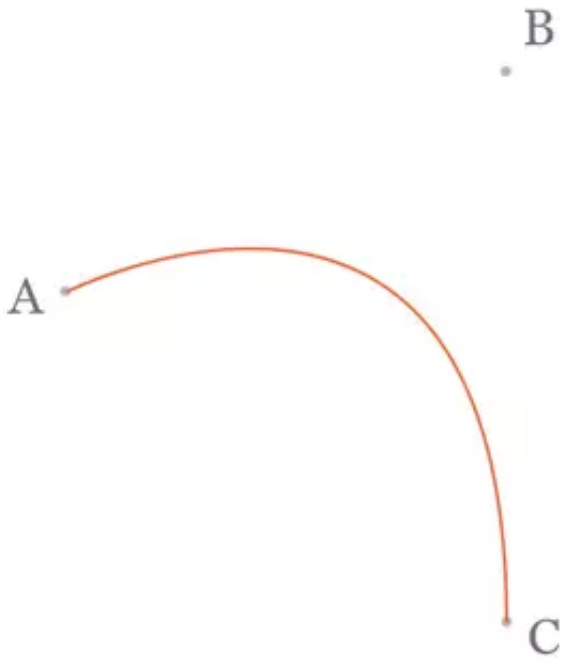
> 上图表示的是一阶曲线生成过程中的某一个阶段，动态过程可以参照下图(本文中贝塞尔曲线相关的动态演示图片来自维基百科)。



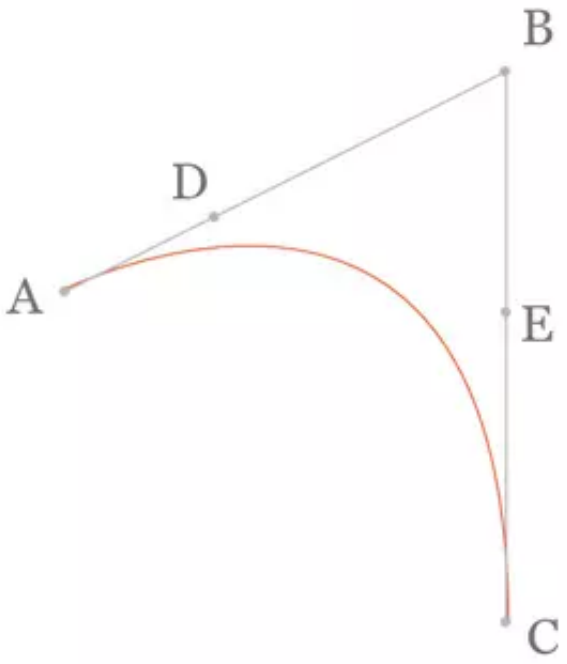
> PS：一阶曲线其实就是前面讲解过的lineTo。

二阶曲线原理：

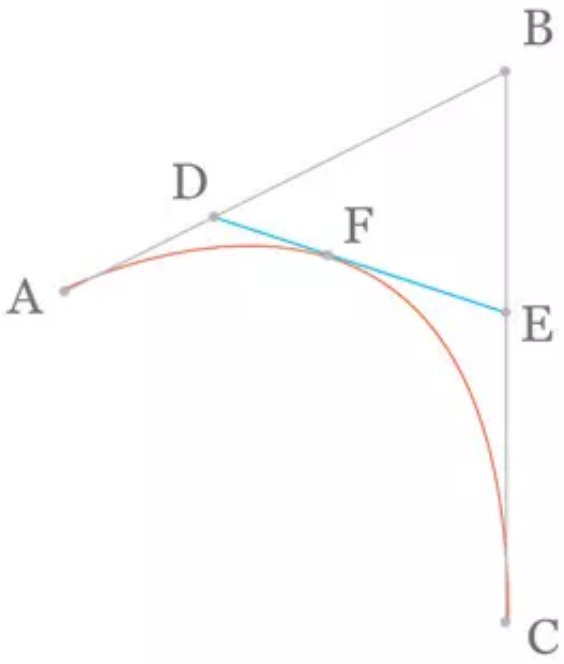
二阶曲线由两个数据点(A 和 C)，一个控制点(B)来描述曲线状态，大致如下：



上图中红色曲线部分就是传说中的二阶贝塞尔曲线，那么这条红色曲线是如何生成的呢？接下来我们就以其中的一个状态分析一下：



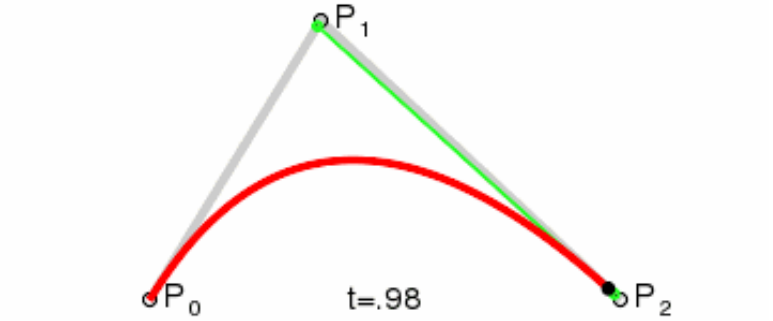
连接AB BC，并在AB上取点D，BC上取点E，使其满足条件：



连接DE，取点F，使得:

$$\frac{AD}{AB}=\frac{BE}{BC}=\frac{DF}{DE}$$

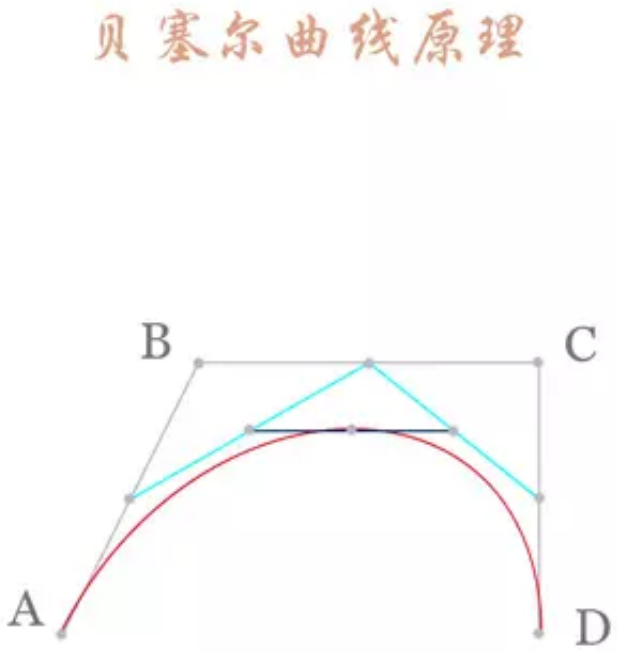
这样获取到的点F就是贝塞尔曲线上的一个点，动态过程如下：



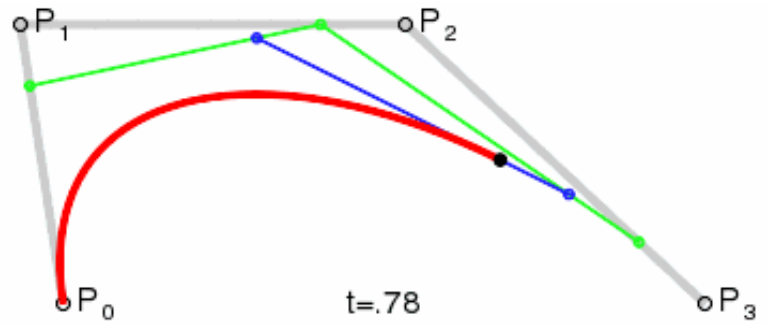
> PS: 二阶曲线对应的方法是quadTo

三阶曲线原理：

三阶曲线由两个数据点(A 和 D)，两个控制点(B 和 C)来描述曲线状态，如下：



三阶曲线计算过程与二阶类似，具体可以见下图动态效果：



> PS: 三阶曲线对应的方法是cubicTo

贝塞尔曲线速查表

强烈推荐点击这里练习贝塞尔曲线，可以加深对贝塞尔曲线的理解程度。

第二步.了解贝塞尔曲线相关函数使用方法

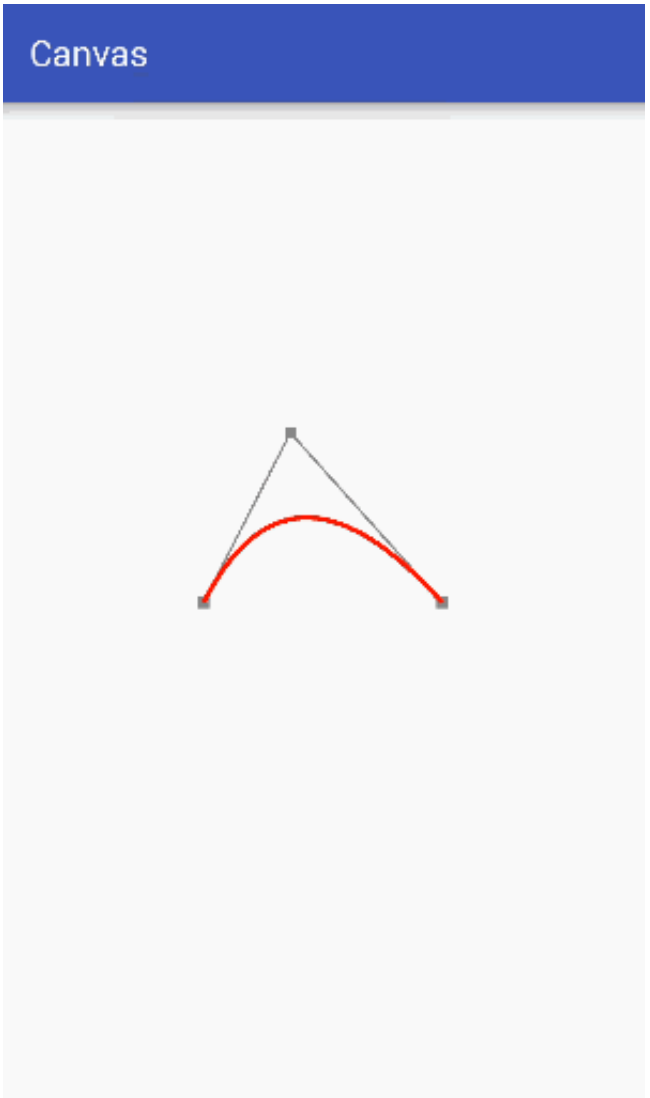
一阶曲线：

一阶曲线是一条线段，非常简单，可以参见上一篇文章Path之基本操作，此处就不详细讲解了。

二阶曲线：

通过上面对二阶曲线的简单了解，我们知道二阶曲线是由两个数据点，一个控制点构成，接下来我们就用一个实例来演示二阶曲线是如何运用的。

首先，两个数据点是控制贝塞尔曲线开始和结束的位置，比较容易理解，而控制点则是控制贝塞尔的弯曲状态，相对来说比较难以理解，所以本示例重点在于理解贝塞尔曲线弯曲状态与控制点的关系，废话不多说，先上效果图：



> 为了更加容易看出控制点与曲线弯曲程度的关系，上图中绘制出了辅助点和辅助线，从上

> 面的动态图可以看出，贝塞尔曲线在动态变化过程中有类似于橡皮筋一样的弹性效果，因此在制作一些弹性效果的时候很常用。

主要代码如下：

```
public class Bezier extends View {

    private Paint mPaint;
    private int centerX, centerY;

    private PointF start, end, control;

    public Bessel1(Context context) {
        super(context);
        mPaint = new Paint();
        mPaint.setColor(Color.BLACK);
        mPaint.setStrokeWidth(8);
        mPaint.setStyle(Paint.Style.STROKE);
        mPaint.setTextSize(60);

        start = new PointF(0,0);
        end = new PointF(0,0);
        control = new PointF(0,0);
    }

    @Override
    protected void onSizeChanged(int w, int h, int oldw, int oldh) {
        super.onSizeChanged(w, h, oldw, oldh);
        centerX = w/2;
        centerY = h/2;

        // 初始化数据点和控制点的位置
        start.x = centerX-200;
        start.y = centerY;
        end.x = centerX+200;
        end.y = centerY;
        control.x = centerX;
        control.y = centerY-100;
    }
```

```
    @Override
    public boolean onTouchEvent(MotionEvent event) {
        // 根据触摸位置更新控制点，并提示重绘
        control.x = event.getX();
        control.y = event.getY();
        invalidate();
        return true;
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);

        // 绘制数据点和控制点
        mPaint.setColor(Color.GRAY);
        mPaint.setStrokeWidth(20);
        canvas.drawPoint(start.x,start.y,mPaint);
        canvas.drawPoint(end.x,end.y,mPaint);
        canvas.drawPoint(control.x,control.y,mPaint);

        // 绘制辅助线
        mPaint.setStrokeWidth(4);
        canvas.drawLine(start.x,start.y,control.x,control.y,mPaint);
        canvas.drawLine(end.x,end.y,control.x,control.y,mPaint);

        // 绘制贝塞尔曲线
        mPaint.setColor(Color.RED);
        mPaint.setStrokeWidth(8);

        Path path = new Path();

        path.moveTo(start.x,start.y);
        path.quadTo(control.x,control.y,end.x,end.y);

        canvas.drawPath(path, mPaint);
    }
}
```

接下文

专栏作者简介（ 点击 → 加入专栏作者 ）

GcsSloop：搜索GcsSloop，发现更多精彩。

打赏支持作者写出更多好文章，谢谢！

阅读原文