Android 自定义圆形渐变色进度条的从实现到开源

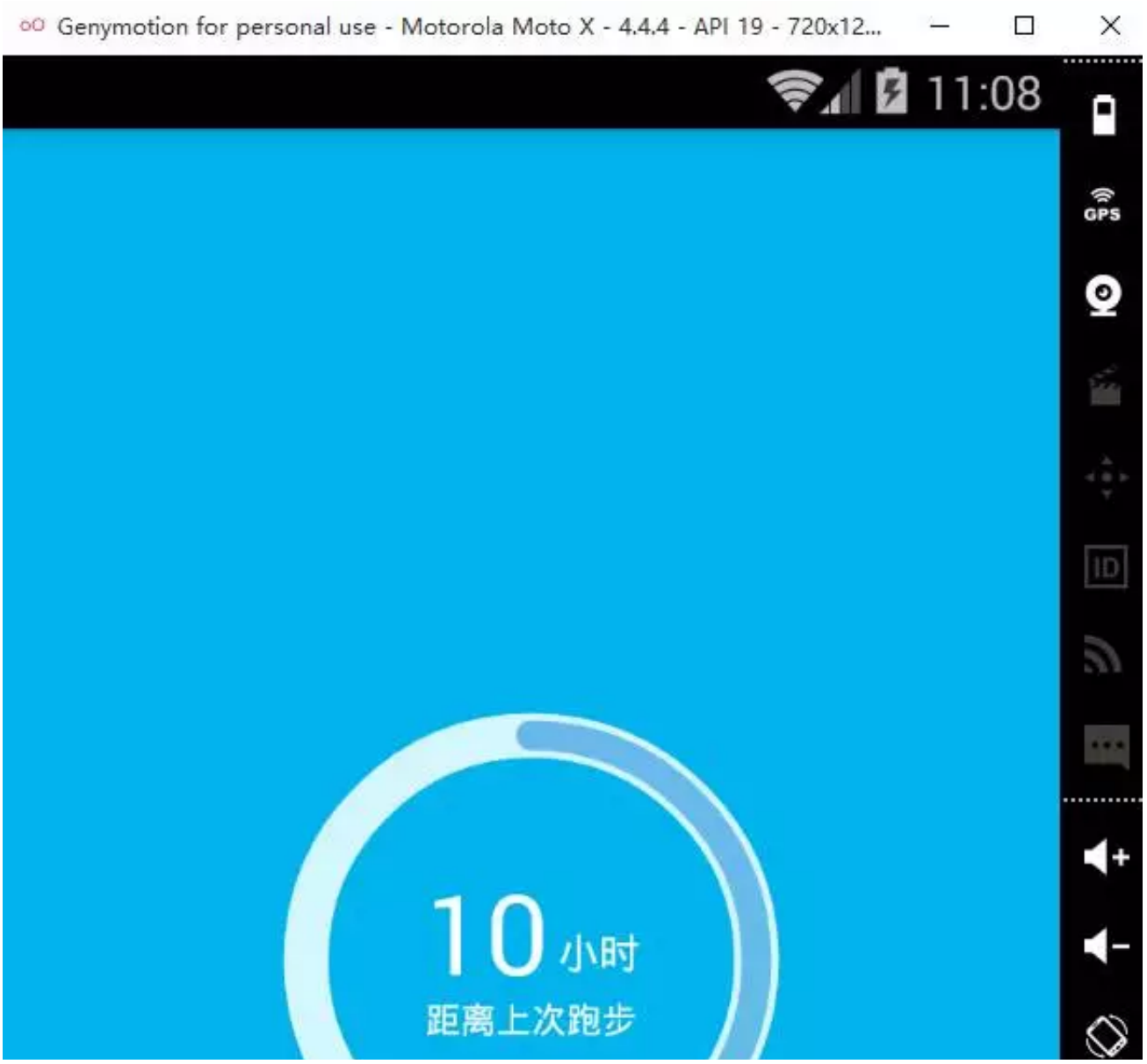
原创 2016-07-25 伯乐专栏/任磊 安卓应用频道

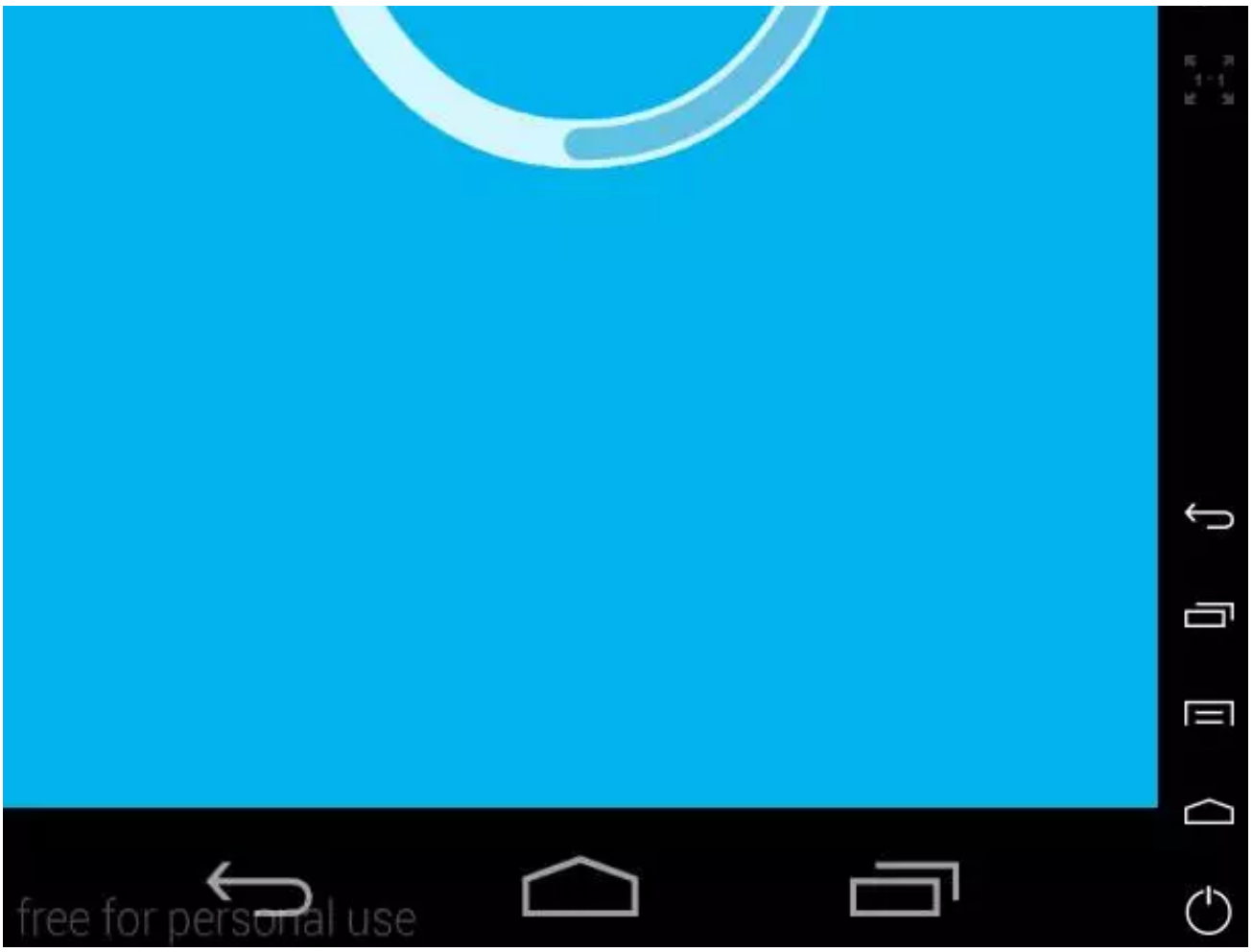
(点击上方公众号，可快速关注)

> 来源：伯乐在线专栏作者 - PleaseCallMeCoder
> 链接：http://android.jobbole.com/84025/
> 点击 → 了解如何加入专栏作者

写在前面的话

3月初我在自定义控件概述中挖下的几个坑，前一段时间已经基本填完了，自定义控件的几种实现方式也分别写了demo来进行说明。今天我们来聊一聊如何把**自己封装一个圆形渐变色进度条控件开源到github，并且上传到jcenter方便别人远程依赖。**先看下效果图：





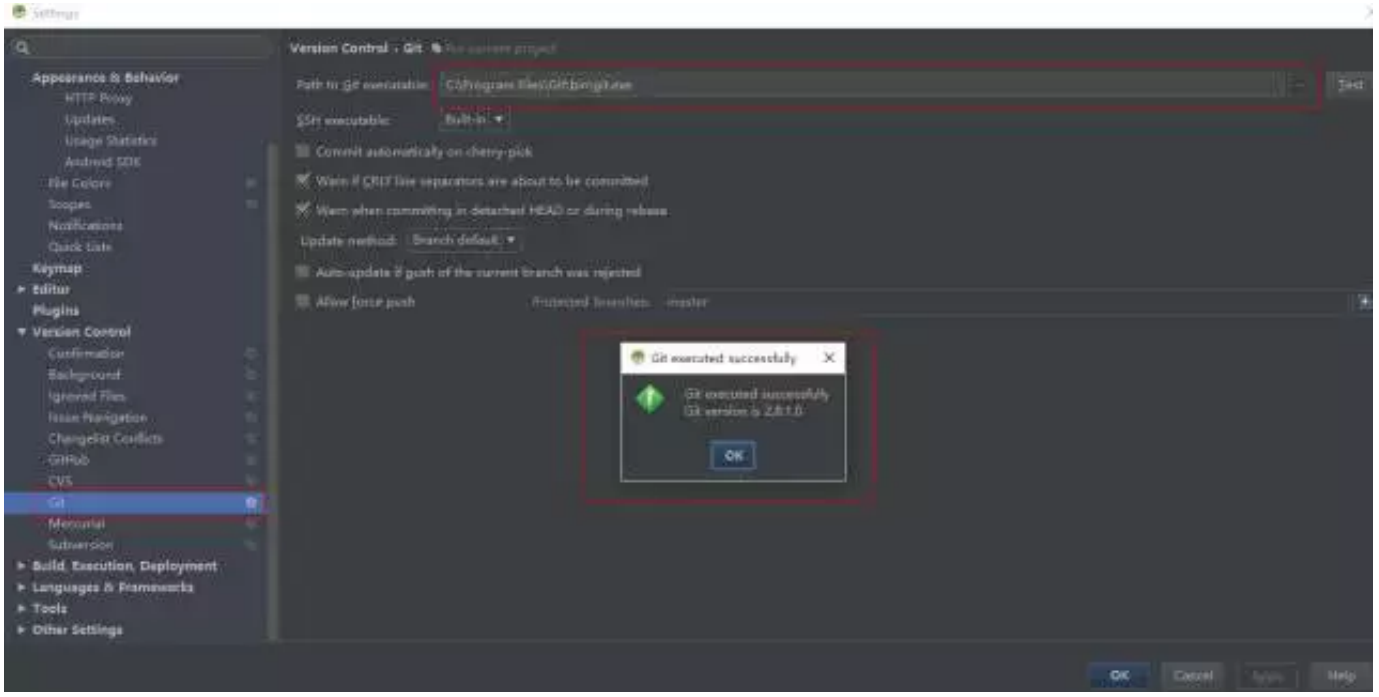
连接github并提交新项目

前提条件：

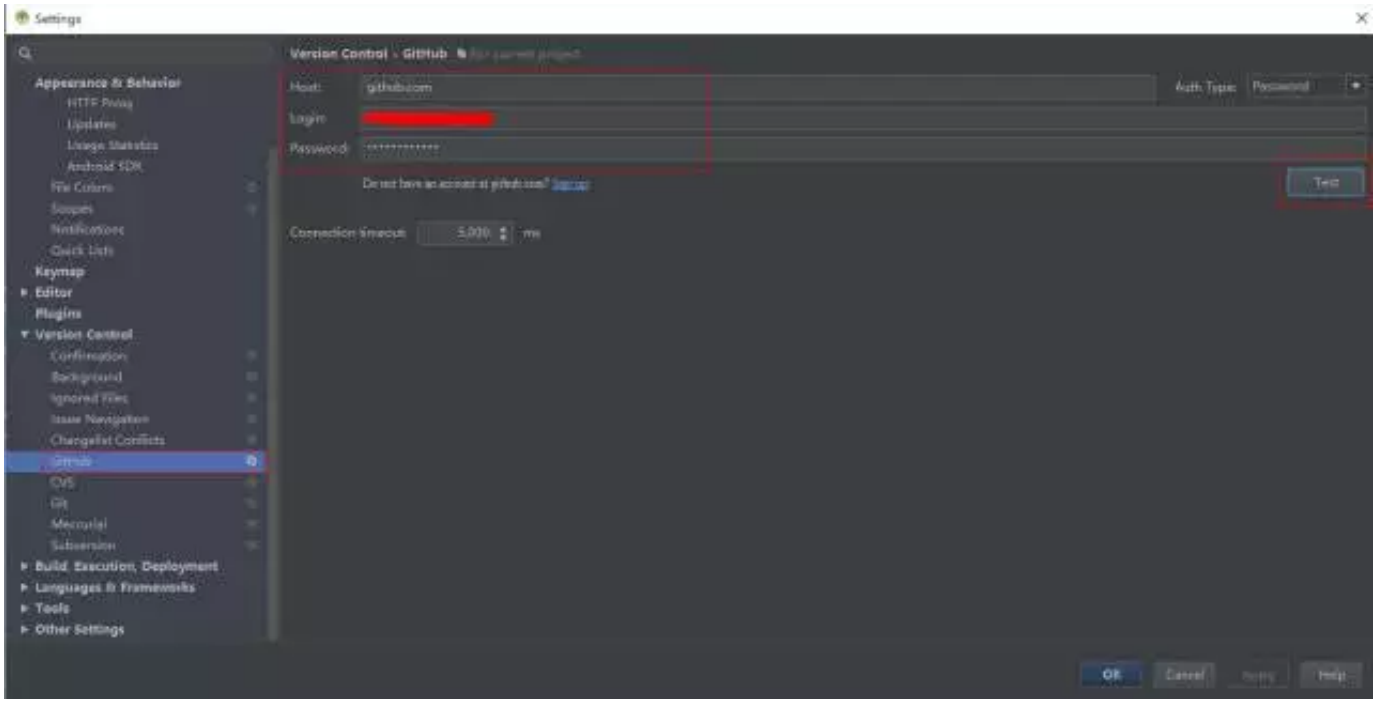
1. 安装Git客户端（下载地址）
2. 有GitHub账号

创建新项目并提交到Github：

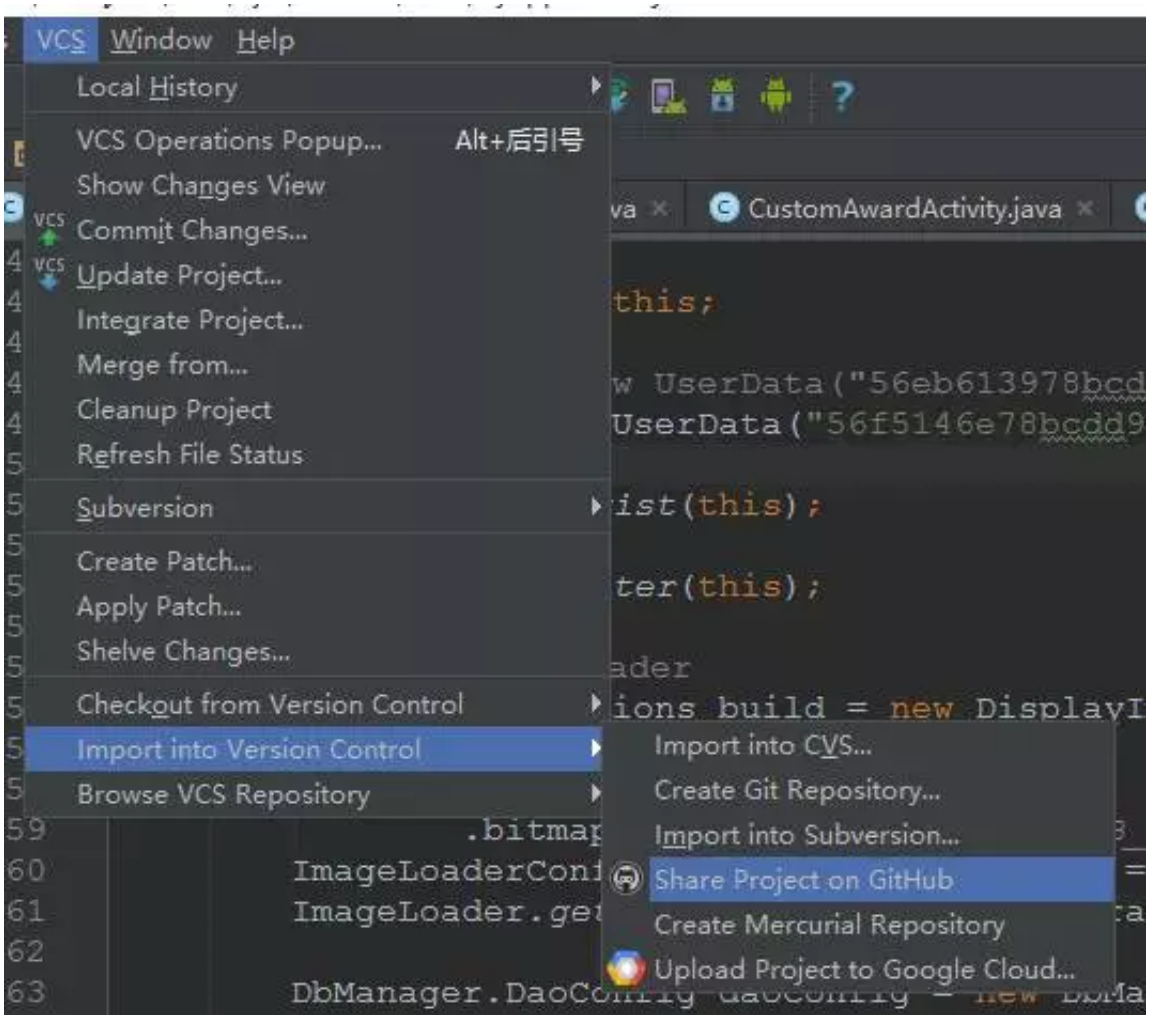
- 在AndroidStudio中新建一个项目
- 配置Git:Settings -> Version Control -> Git ,设置git目录,点击Test测试,如果成功会有Success提示



- 关联自己的Github账号：Settring -> VersionControl ->GitHub,设置自己的github账号,密码,点击Test测试,如果成功会有Success提示



- 分享项目:VCS -> Import into Version Control -> Share Project on GitHub, 点击之后开始设置 repository name(如果你有设置过MasterPassword 会弹出对话框让填入密码)



点击Share之后, 你就可以在GitHub上看到新的仓库，同时在AndroidStudio中CVS下也可以看到版本控制Git，这样就已经设置成功。

- 如果你想解除关联,只需要Settrings -> Version Control删掉关联就可以了。

完成circlebar代码

接下来我们来实现我们的圆形渐变色进度条，需要的技能：

- Canvas绘图基础
- Shader绘制渐变色
- 绘制进度条的原理
- Canvas绘图基础

关于Canvas绘图，网上的教程很多，这里大概的说一下都有哪些点需要了解：

- Canvas坐标系与绘图坐标系。
- drawARGB：Canvas中的drawARGB可以用来对整个Canvas以某种统一的颜色整体绘制，四个参数分别是Alpha、Red、Green、Blue，取值都是0-255。

- drawText：Canvas中用drawText方法绘制文字。
- drawPoint：Canvas中用drawPoint方法绘制点。
- drawLine：Canvas通过drawLine方法绘制一条线段，通过drawLines方法绘制多段线。
- drawRect：Canvas通过drawRect方法绘制矩形。
- drawCircle：Canvas中用drawCircle方法绘制圆形。
- drawOval：Canvas中提供了drawOval方法绘制椭圆。
- drawArc：Canvas中提供了drawArc方法用于绘制弧，这里的弧指两种：弧面和弧线，弧面即用弧围成的填充面，弧线即为弧面的轮廓线。
- drawPath：Canvas通过drawPath方法可以绘制Path。那Path是什么呢？Path致以过来是路径的意思，在Android中，Path是一种线条的组合图形，其可以由直线、二次曲线、三次曲线、椭圆的弧等组成。Path既可以画线条，也可以画填充面。
- drawBitmap：Canvas中提供了drawBitmap方法用于绘制Bitmap。

Paint和Shader

Paint

- 画笔Paint控制着所绘制的图形的具体外观，Paint默认的字体大小为12px，在绘制文本时我们往往要考虑密度density设置合适的字体大小。画笔的默认颜色为黑色，默认的style为FILL，默认的cap为BUTT，默认的线宽为0.
- 在画面状的图形时，如果Paint的style是FILL，那么绘制的就是填充面；如果是STROKE，那么绘制的就是轮廓线。

Shader

androd 提供了Shader类专门用来渲染图像以及一些几何图形。Shader下面包括几个直接子类，分别是BitmapShader、 ComposeShader、LinearGradient、RadialGradient、SweepGradient。BitmapShader主要用来渲染图像，LinearGradient 用来进行梯度渲染，RadialGradient 用来进行环形渲染，SweepGradient 用来进行梯度渲染，ComposeShader则是一个 混合渲染，可以和其它几个子类组合起来使用。

Shader类的使用，都需要先构建Shader对象，然后通过Paint的setShader方法设置渲染对象，然后设置渲染对象，然后再绘制时使用这个Paint对象即可。

绘制进度条

talk is cheap，show you my code。下面说一下绘制圆形渐变进度条的过程。

首先先跟大家说下原理，我们的主要绘制过程其实非常简单，**调用drawArc方法绘制圆弧。**

先来说下drawArc方法。

```
/**
 * 绘制弧
 * drawArc (RectF oval, float startAngle, float sweepAngle, boolean useCenter, Paint paint)
 * oval是RecF类型的对象，其定义了椭圆的形状
 * startAngle指的是绘制的起始角度，钟表的3点位置对应着0度，如果传入的startAngle小于0或者大于等于360，那么用startAngle对360进行取模后作为起始绘制角度。
 * sweepAngle指的是从startAngle开始沿着钟表的顺时针方向旋转扫过的角度。如果sweepAngle大于等于360，那么会绘制完整的椭圆环。如果sweepAngle小于0，那么会用sweepAngle对360进行取模后作为扫过的角度。
 * useCenter是个boolean值，如果为true，表示在绘制完环之后，用椭圆的中心点连接环上的起点和终点以闭合环；如果值为false，表示在绘制完环之后，环的起点和终点直接连接，不经过椭圆的中心点。
 */
```

里边需要传一个定义好的矩形

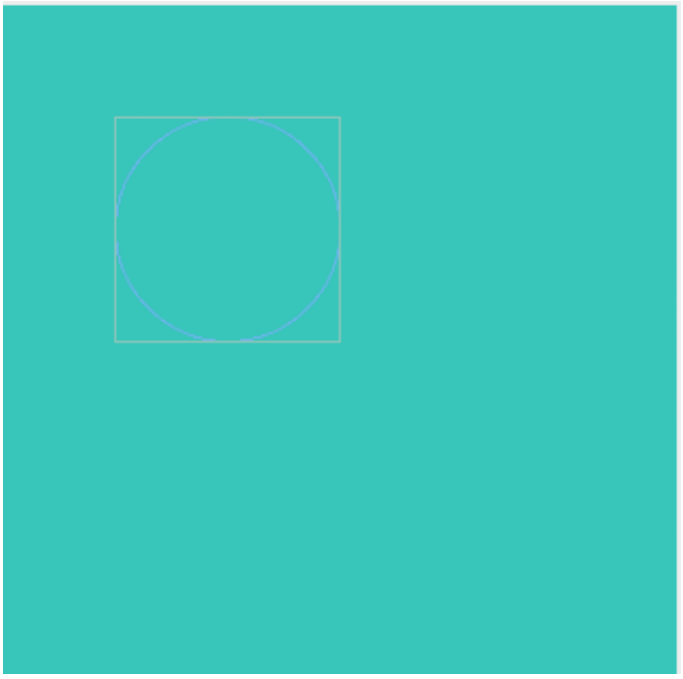
```
/**
 *RectF rectF = new RectF(100, 100, 300, 300);
 * 这四个参数分别代表的意思是：left  top  right  bottom  左 上 右 下
 * left ： 矩形左边的X坐标
 * top:  矩形顶部的Y坐标
 * right： 矩形右边的X坐标
 * bottom： 矩形底部的Y坐标
 * 其实就是矩形的左上角和右下角的坐标值
 */
```

接下来我们来看一个小例子，自定义view，在ondraw中调用如下方法：

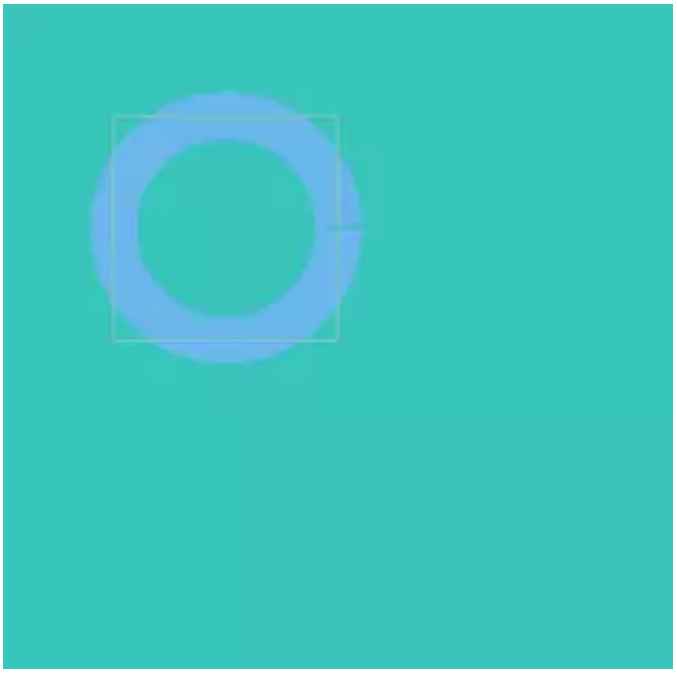
```
//绘制矩形框和圆弧
private void drawArc(Canvas canvas) {


    canvas.drawARGB(255, 56, 197, 186);
```

```
    RectF rectF = new RectF(100, 100, 300, 300);

    paint.setStrokeWidth(1 * density);//设置线宽
    paint.setColor(0xFF6BB7ED);//设置颜色
    paint.setStyle(Paint.Style.FILL);//默认设置画笔为填充模式

    //绘制椭圆
    paint.setStyle(Paint.Style.STROKE);//设置画笔为线条模式
    canvas.drawArc(rectF, 0, 359, false, paint);

    paint.setStrokeWidth(1 * density);//设置线宽
    paint.setColor(0xff8bc5ba);//设置颜色
    paint.setStyle(Paint.Style.STROKE);//默认设置画笔为填充模式

    canvas.drawRect(rectF, paint);
}
```

效果如下图：



我们看到当我们以同一个矩形rectF为基准画了一个圆弧和矩形，圆弧正好为矩形的内切圆。这时候我们增大圆弧的线宽为21*density。效果如下图：

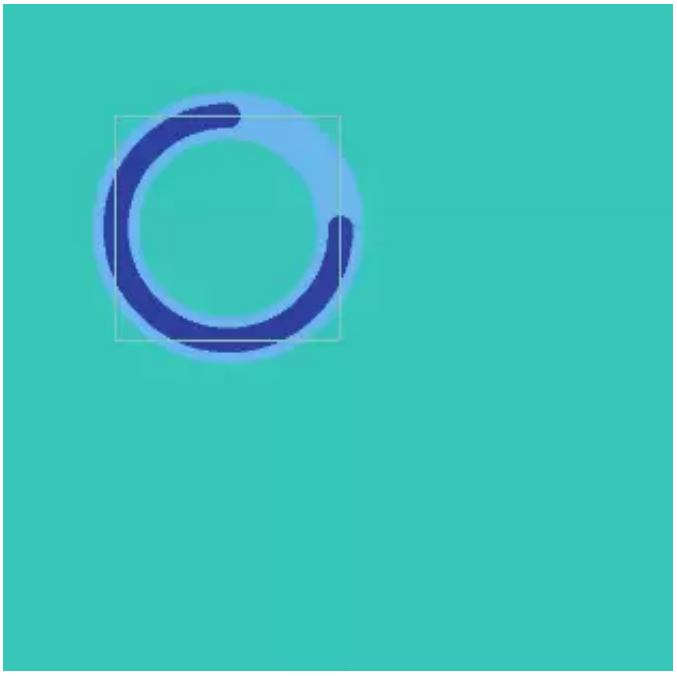


可以看到圆弧以矩形为基准宽度向矩形外和矩形内各增大了10*density。

这时候我们以rectF为基准再多画一个圆弧。

```
paint.setStrokeWidth(11 * density);//设置线宽
paint.setColor(0xFF303f9f);//设置颜色
paint.setStyle(Paint.Style.FILL);//默认设置画笔为填充模式
paint.setStrokeCap(Paint.Cap.ROUND);

paint.setStyle(Paint.Style.STROKE);//设置画笔为线条模式
canvas.drawArc(rectF, 0, 270, false, paint);
```

效果如下：



perfect！可以看到我们要的效果基本已经出来了。

然后说一下具体的一些细节。

onmeasure，我们在里边去设置控件的大小为正方形：

```
@Override
protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {

    int height = getDefaultSize(getSuggestedMinimumHeight(), heightMeasureSpec);
    int width = getDefaultSize(getSuggestedMinimumWidth(), widthMeasureSpec);
    int min = Math.min(width, height);// 获取View最短边的长度
    setMeasuredDimension(min, min);// 强制改View为以最短边为长度的正方形

    circleStrokeWidth = Textscale(35, min);// 圆环的宽度
    pressExtraStrokeWidth = Textscale(2, min);// 圆环离矩形的距离
    /**
     * 这四个参数分别代表的意思是：left  top  right  bottom  左 上 右 下
     * left ： 矩形左边的X坐标
     * top:  矩形顶部的Y坐标
     * right： 矩形右边的X坐标
     * bottom： 矩形底部的Y坐标
     * 其实就是矩形的左上角和右下角的坐标值
     */
    mColorWheelRectangle.set(circleStrokeWidth + pressExtraStrokeWidth,
```

```
            circleStrokeWidth + pressExtraStrokeWidth, min
                    - circleStrokeWidth - pressExtraStrokeWidth, min
                    - circleStrokeWidth - pressExtraStrokeWidth);// 设置圆环内圆的外接正方形
    mColorWheelPaint.setStrokeWidth(circleStrokeWidth - 5);
    mColorWheelPaintCentre.setStrokeWidth(circleStrokeWidth + 5);
    mDefaultWheelPaint.setStrokeWidth(circleStrokeWidth - Textscale(2, min));
    mDefaultWheelPaint.setShadowLayer(Textscale(10, min), 0, 0, Color.rgb(127, 127, 127));// 设置阴影
}
```

定义了一个setShaderColor方法来设置渐变色，这里我们用梯度渐变。

```
/**
 * 设置渐变色
 *
 * @param shaderColor
 */
public void setShaderColor(int[] shaderColor) {
    this.mColors = shaderColor;
    Shader newShader = new SweepGradient(0, 0, mColors, null);
    mColorWheelPaint.setShader(newShader);
}
```

最后继承Animation类自定义一个动画效果，即根据进度计算角度，来慢慢绘制我们的进度条。

```
/**
 * 进度条动画
 *
 * @author Administrator
 */
public class BarAnimation extends Animation {
    public BarAnimation() {

    }

    /**
     * 每次系统调用这个方法时， 改变mSweepAnglePer，mPercent，stepnumbernow的值，
     * 然后调用postInvalidate()不停的绘制view。
```

```
     */
    @Override
    protected void applyTransformation(float interpolatedTime,
                                       Transformation t) {
        super.applyTransformation(interpolatedTime, t);
        if (interpolatedTime < 1.0f) {
            mPercent = Float.parseFloat(fnum.format(interpolatedTime
                    * stepnumber * 100f / stepnumbermax));// 将浮点值四舍五入保留一位小数
            mSweepAnglePer = interpolatedTime * stepnumber * 360
                    / stepnumbermax;
            stepnumbernow = (int) (interpolatedTime * stepnumber);
        } else {
            mPercent = Float.parseFloat(fnum.format(stepnumber * 100f
                    / stepnumbermax));// 将浮点值四舍五入保留一位小数
            mSweepAnglePer = stepnumber * 360 / stepnumbermax;
            stepnumbernow = stepnumber;
        }
        postInvalidate();
    }
}
```

好了，到这里我们的自定义圆形渐变色进度条就完全搞定了。

上传到jcenter

上传到jcenter，我用的是bintray-release这个插件。这里可以参考这篇：
http://blog.csdn.net/lmj623565791/article/details/51148825

步骤如下：

- 注册bintray.com账号，注册地址
- 引入bintray-release，在需要上传的module里面填写相关publish的信息
- 调用上传的命令
- Add to Jcenter提交审核

需要注意的是，这里我遇到了一个问题，在这里跟大家分享一下，即当我上传的moudle中带有中文注释，编码为utf-8的时候，上传会抛出异常，然后上传失败。从网上也没有找到太好的解决办法，最后我把中文中是全部删掉才上传成功。如果你有好的解决办法，请告诉我0.0

结语

代码我已经上传到了github，github中有说明文档，欢迎Star、Fork。链接：https://github.com/PleaseCallMeCoder/CircleProgressBar

专栏作者简介（ 点击 → 加入专栏作者 ）

PleaseCallMeCoder：我是 PleaseCallMeCoder，一个小小的90后程序员。热衷于移动开发，喜欢研究新技术，奔跑在成为大神的路上。

打赏支持作者写出更多好文章，谢谢！

阅读原文