**Drawable**

Bitmap是Android对图像的定义描述，而Drawable则是对图像的展现描述，在View视图中显示图像都是通过Drawable来实现的。其中有关Bitmap的介绍参见《[Android开发笔记（九十四）图片的基本加工](http://blog.csdn.net/aqi00/article/details/51282669)》，有关Drawable的介绍参见《[Android开发笔记（七）初识Drawable](http://blog.csdn.net/aqi00/article/details/50132481)》。虽然ImageView提供了setImageBitmap方法，但查看该方法的源码，会发现内部还是调用setImageDrawable方法，同时利用BitmapDrawable完成Bitmap与Drawable的转换。

    **public** **void** setImageBitmap(Bitmap bm) {

        // if this is used frequently, may handle bitmaps explicitly

        // to reduce the intermediate drawable object

        setImageDrawable(**new** BitmapDrawable(mContext.getResources(), bm));

    }

一般我们要自定义图像控件，通常基于ImageView来自定义，例如ImageButton。其实对于一些简单的图像处理，我们可以自定义Drawable来实现，比如说裁剪图片、给图片添加文本、给图片添加简单动画等等。

**圆形、椭圆、圆角矩形的Drawable**

对图片进行简单形状的裁剪，这是很常见的操作，例如手机桌面上的APP图标是圆角正方形样式，例如csdn客户端的用户头像是圆形图片等等。这些简单的裁剪，可直接使用Canvas类的相关方法来实现，比如调用drawCircle方法完成圆形裁剪，调用drawOval方法完成椭圆形裁剪，调用drawRoundRect方法完成圆角矩形裁剪，更多有关Canvas的介绍参见《[Android开发笔记（十三）视图绘制的几个方法](http://blog.csdn.net/aqi00/article/details/50221679)》。

因为裁剪图片一般是处理位图，所以我们可基于BitmapDrawable来自定义相关Drawable，这样只需自己实现少数方法（构造函数、draw函数等等）。需要注意的是，由于我们要画的是裁剪后的图片，因此不能直接调用drawBitmap方法，而要把Bitmap对象塞入BitmapShader对象中，然后调用Paint的setShader方法，把图像作为阴影来绘制，从而实现裁剪显示的功能。

下面是自定义圆形、椭圆、圆角矩形的Drawable效果图



下面是圆形裁剪图像（CircleDrawable）的代码例子：

**import** android.content.res.Resources;

**import** android.graphics.Bitmap;

**import** android.graphics.BitmapShader;

**import** android.graphics.Canvas;

**import** android.graphics.Paint;

**import** android.graphics.Shader.TileMode;

**import** android.graphics.drawable.BitmapDrawable;

**public** **class** CircleDrawable **extends** BitmapDrawable {

    **private** Paint mPaint;

    **public** CircleDrawable(Resources res, Bitmap bitmap) {

        **super**(res, bitmap);

        BitmapShader bitmapShader = **new** BitmapShader(bitmap,

                TileMode.CLAMP, TileMode.CLAMP);

        mPaint = **new** Paint();

        mPaint.setAntiAlias(**true**);  //抗锯齿

        mPaint.setShader(bitmapShader);

    }

    @Override

    **public** **void** draw(Canvas canvas) {

        **int** width = getBitmap().getWidth();

        **int** height = getBitmap().getHeight();

        **int** radius = Math.min(width, height) / 2;

        **int** x\_pos = (width>radius+radius)?width/2:radius;

        **int** y\_pos = (height>radius+radius)?height/2:radius;

        canvas.drawCircle(x\_pos, y\_pos, radius, mPaint);

    }

}

下面是椭圆形裁剪图像（OvalDrawable）的代码例子：

**import** android.content.res.Resources;

**import** android.graphics.Bitmap;

**import** android.graphics.BitmapShader;

**import** android.graphics.Canvas;

**import** android.graphics.Paint;

**import** android.graphics.RectF;

**import** android.graphics.Shader.TileMode;

**import** android.graphics.drawable.BitmapDrawable;

**public** **class** OvalDrawable **extends** BitmapDrawable {

    **private** Paint mPaint;

    **public** OvalDrawable(Resources res, Bitmap bitmap) {

        **super**(res, bitmap);

        BitmapShader bitmapShader = **new** BitmapShader(bitmap,

                TileMode.CLAMP, TileMode.CLAMP);

        mPaint = **new** Paint();

        mPaint.setAntiAlias(**true**);  //抗锯齿

        mPaint.setShader(bitmapShader);

    }

    @Override

    **public** **void** draw(Canvas canvas) {

        **int** width = getBitmap().getWidth();

        **int** height = getBitmap().getHeight();

        RectF oval = **new** RectF(0, 0, width, height);

        canvas.drawOval(oval, mPaint);

    }

}

下面是圆角矩形裁剪图像（RoundDrawable）的代码例子：

**import** android.content.res.Resources;

**import** android.graphics.Bitmap;

**import** android.graphics.BitmapShader;

**import** android.graphics.Canvas;

**import** android.graphics.Paint;

**import** android.graphics.RectF;

**import** android.graphics.Shader.TileMode;

**import** android.graphics.drawable.BitmapDrawable;

**public** **class** RoundDrawable **extends** BitmapDrawable {

    **private** Paint mPaint;

    **private** RectF mRect;

    **private** **int** mCornerRadius = 10;

    **public** RoundDrawable(Resources res, Bitmap bitmap) {

        **super**(res, bitmap);

        BitmapShader bitmapShader = **new** BitmapShader(bitmap,

                TileMode.CLAMP, TileMode.CLAMP);

        mPaint = **new** Paint();

        mPaint.setAntiAlias(**true**);  //抗锯齿

        mPaint.setShader(bitmapShader);

    }

    **public** **void** setCornerRadius(**int** corner\_radius) {

        mCornerRadius = corner\_radius;

    }

    **public** **int** getCornerRadius() {

        **return** mCornerRadius;

    }

    @Override

    **public** **void** setBounds(**int** left, **int** top, **int** right, **int** bottom) {

        **super**.setBounds(left, top, right, bottom);

        mRect = **new** RectF(left, top, right, bottom);

    }

    @Override

    **public** **void** draw(Canvas canvas) {

        canvas.drawRoundRect(mRect, mCornerRadius, mCornerRadius, mPaint);

    }

}

**添加水印的Drawable**

给图片添加水印文字，这也是一种常见的图片加工操作。通过自定义Drawable，可以不用修改原图片，直接在展示时添加水印文本，更方便更快捷。添加文本操作可调用Canvas类的drawText方法，至于文本颜色、文本大小等属性的设置，可通过Paint类来实现。比如设置文本颜色，调用的是Paint类的setColor；设置文本大小，调用的是Paint类的setTextSize；设置文本对齐方式，调用的是Paint类的setTextAlign。

下面是自定义添加水印的Drawable效果图



下面是添加水印图像（MarkDrawable）的代码例子：

**import** android.content.res.Resources;

**import** android.graphics.Bitmap;

**import** android.graphics.Canvas;

**import** android.graphics.Color;

**import** android.graphics.Paint;

**import** android.graphics.Paint.FontMetrics;

**import** android.graphics.Point;

**import** android.graphics.drawable.BitmapDrawable;

**public** **class** MarkDrawable **extends** BitmapDrawable {

    **private** **static** **final** String TAG = "MarkDrawable";

    **private** Paint mPaint;

    **private** String mText;

    **private** **int** mTextColor;

    **private** **float** mTextSize;

    **private** **int** mTextAlign;

    **public** **static** **int** ALIGN\_TOP = 1;

    **public** **static** **int** ALIGN\_CENTER = 2;

    **public** **static** **int** ALIGN\_BOTTOM = 3;

    **public** MarkDrawable(Resources res, Bitmap bitmap) {

        **super**(res, bitmap);

        mTextColor = Color.GREEN;

        mTextSize = 40f;

        mTextAlign = ALIGN\_CENTER;

    }

    **public** **void** setTextColor(**int** text\_color) {

        mTextColor = text\_color;

    }

    **public** **int** getTextColor() {

        **return** mTextColor;

    }

    **public** **void** setTextSize(**float** text\_size) {

        mTextSize = text\_size;

    }

    **public** **float** getTextSize() {

        **return** mTextSize;

    }

    **public** **void** setTextAlign(**int** text\_align) {

        mTextAlign = text\_align;

    }

    **public** **int** getTextAlign() {

        **return** mTextAlign;

    }

    **public** **void** setText(String text) {

        mText = text;

        mPaint = **new** Paint();

        mPaint.setAntiAlias(**true**);

        mPaint.setColor(mTextColor);

        mPaint.setTextSize(mTextSize);

        mPaint.setTextAlign(Paint.Align.CENTER);

    }

    @Override

    **public** **void** draw(Canvas canvas) {

        **super**.draw(canvas);

        **if** (mPaint != **null**) {

            Point point = getFontScope(mText, mTextSize);

            **int** width = getBitmap().getWidth();

            **int** height = getBitmap().getHeight();

            **int** x\_pos = (width>point.x)?(width/2):0;

            **int** y\_pos = 0;

            **if** (mTextAlign == ALIGN\_TOP) {

                y\_pos = point.y;

            } **else** **if** (mTextAlign == ALIGN\_CENTER) {

                y\_pos = height / 2;

            } **else** **if** (mTextAlign == ALIGN\_BOTTOM) {

                y\_pos = height - point.y/2;

            }

            canvas.drawText(mText, x\_pos, y\_pos, mPaint);

        }

    }

    //根据字体大小获得文字宽度和高度

    **private** Point getFontScope(String text, **float** size) {

        Point point = **new** Point();

        Paint paint = **new** Paint();

        paint.setTextSize(size);

        FontMetrics fm = paint.getFontMetrics();

        point.x = (**int**) paint.measureText(text, 0, text.length());

        point.y = (**int**) Math.ceil(fm.descent - fm.ascent);

        **return** point;

    }

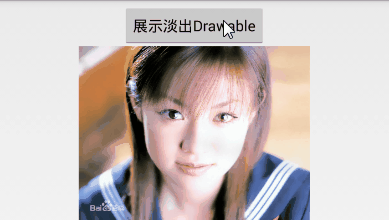
}

**灰度动画的Drawable**

通过自定义Drawable，我们还能够实现简单的图形动画。提到透明度动画，大家肯定马上想到AlphaAnimation，这个透明度动画其实也能用Drawable实现。具体的说，便是采用Handler+Runnable机制，间隔很短的时间依次执行处理任务。设置图形的灰度可调用setAlpha并刷新图形，由于setAlpha方法内部已经调用了invalidateSelf方法，所以我们不必再次刷新画面。这样随着时间流逝，依次展现渐变的灰度便实现动画效果了。有关Runnable的介绍参见《[Android开发笔记（四十七）Runnable接口实现多线程](http://blog.csdn.net/aqi00/article/details/50515561)》。

在前面的博文《[Android开发笔记（十五）淡入淡出动画](http://blog.csdn.net/aqi00/article/details/50251541)》中，博主提到可以使用AlphaAnimation和TransitionDrawable，现在又多了第三个办法，就是自定义的AlphaDrawable。同一个功能有多个实现方式，这就是Android的魅力所在呀。

下面是自定义灰度动画的Drawable效果图



下面是灰度动画图像（AlphaDrawable）的代码例子：

**import** android.content.res.Resources;

**import** android.graphics.Bitmap;

**import** android.graphics.Canvas;

**import** android.graphics.drawable.BitmapDrawable;

**import** android.os.Handler;

**public** **class** AlphaDrawable **extends** BitmapDrawable {

    **private** Handler mHandler = **new** Handler();

    **private** **int** mPeriod = 5;

    **private** **int** mCount = 100;

    **private** **int** mGap = 0;

    **public** AlphaDrawable(Resources res, Bitmap bitmap) {

        **super**(res, bitmap);

    }

    **public** **void** setPeriod(**int** period) {

        mPeriod = period;

    }

    **public** **int** getPeriod() {

        **return** mPeriod;

    }

    @Override

    **public** **void** draw(Canvas canvas) {

        **super**.draw(canvas);

        **if** (mGap == 0) {

            mGap = mPeriod\*1000 / mCount;

        }

        mHandler.postDelayed(mRefresh, mGap);

    }

    **private** Runnable mRefresh = **new** Runnable() {

        @Override

        **public** **void** run() {

            mCount--;

            **if** (mCount >= 0) {

                setAlpha((**int**) (255 \* (100-mCount)/100.0));

            }

        }

    };

}

[点击下载本文用到的自定义Drawable的工程代码](http://download.csdn.net/detail/aqi00/9617828)

[点此查看Android开发笔记的完整目录](http://blog.csdn.net/aqi00/article/details/50012511)