[Android中缩放图片的方法] (https://blog.csdn.net/tugele/article/details/80751870)

简介

在Android项目中常常需要调整原始图片的尺寸大小以适应存储、传输和图片处理等需求。在Android API中提供了一些缩放图片的方法，在项目中发现，使用Android API中的Canvas、BitmapFactory和ThumbnailUtils等类的相关方法缩放图片，锯齿感明显，图像质量不高；另外还有一些第三方的开源库专门用于在Android平台缩放图片；在FFmpeg中也提供了缩放图片和视频的方法，可以编译FFmpeg在Android平台调用相关方法。本文将总结在项目中使用上述方法的操作和实现的效果。

一、使用Canvas

1. **drawBitmap**

绘制bitamp方法说明：

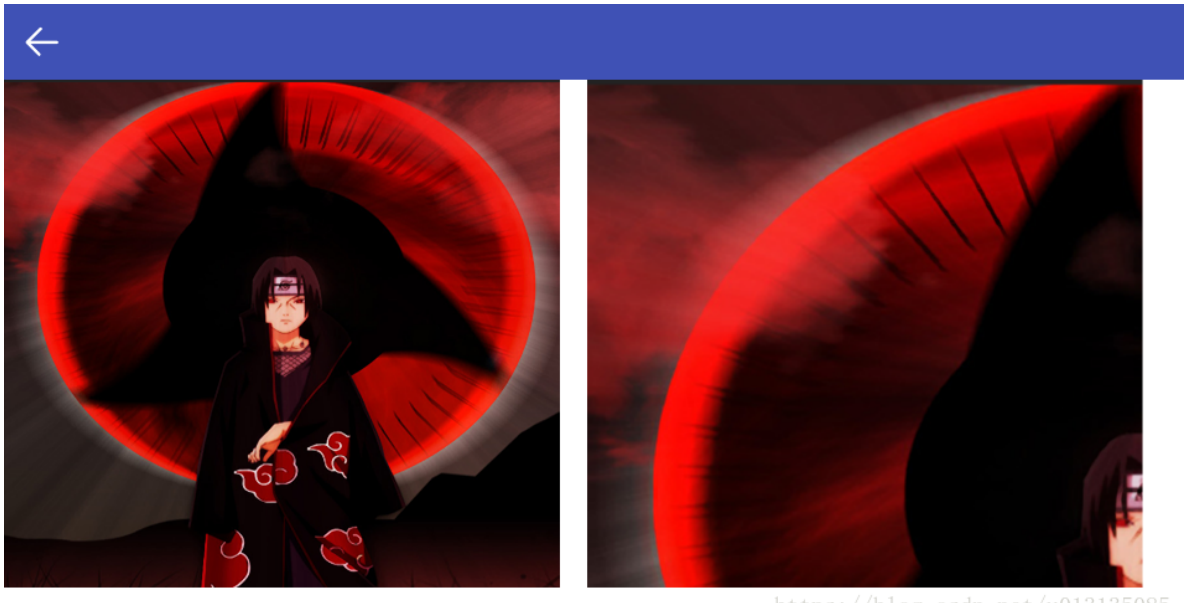
```
/*
* @param
*  bitmap 位图
*  left 绘制区域距离左边界偏移量
*  top 绘制区域距离上边界偏移量
*  paint 画笔
* 在View中指定位置绘制bitmap
* 注：传入的参数中的偏移量是指对于View的偏移。
*/
public void drawBitmap(@NonNull Bitmap bitmap, float left, float top,
            @Nullable Paint paint)

/*
* @param
*  bitmap 位图
*  src bitmap需要绘制的面积，若src的面积小于bitmap时会对bitmap进行裁剪，
*      一般来说需要绘制整个bitmap时可以为null
*  dst 在画布中指定绘制bitmap的位置，当这个区域的面积与bitmap要显示的面积不匹配时，
*      会进行缩放，不可为null
*  paint 画笔
* 在指定位置绘制指定大小的bitmap
*/
public void drawBitmap(@NonNull Bitmap bitmap, @Nullable Rect src, @NonNull
RectF dst,
            @Nullable Paint paint)
public void drawBitmap(@NonNull Bitmap bitmap, @Nullable Rect src, @NonNull Rect
dst,
            @Nullable Paint paint)

/*
* @param
*  bitmap 位图
*  matrix 当绘制位图时需要转变时使用的矩阵
*  paint 画笔
* 使用指定的矩阵绘制位图
```

```
*/
public void drawBitmap(@NonNull Bitmap bitmap, @NonNull Matrix matrix,
            @Nullable Paint paint)
```

java代码：

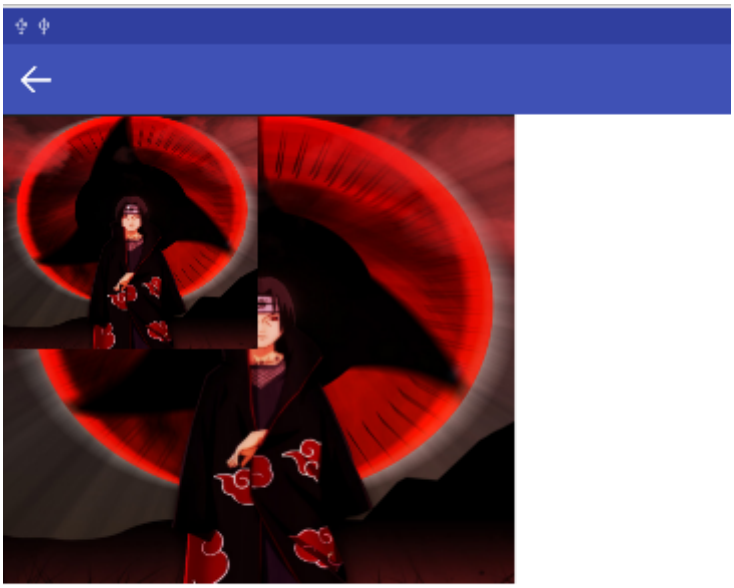
```
    public CustomView(Context context, AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);
        setLayerType(LAYER_TYPE_SOFTWARE,null);
        mBitmap = BitmapFactory.decodeResource(getResources(), R.mipmap.you);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        // 画出原图像
        canvas.drawBitmap(mBitmap, 0, 0, null);
        // 指定图片绘制区域(左上角的四分之一)
        Rect src = new Rect(0,0,mBitmap.getWidth()/2,mBitmap.getHeight()/2);
        // 指定图片在屏幕上显示的区域(原图大小)
        Rect dst = new
Rect(mBitmap.getWidth()+50,0,mBitmap.getWidth()+50+mBitmap.getWidth(),mBitmap.ge
tHeight());
        canvas.drawBitmap(mBitmap, src,dst,null);

    }
```

效果：

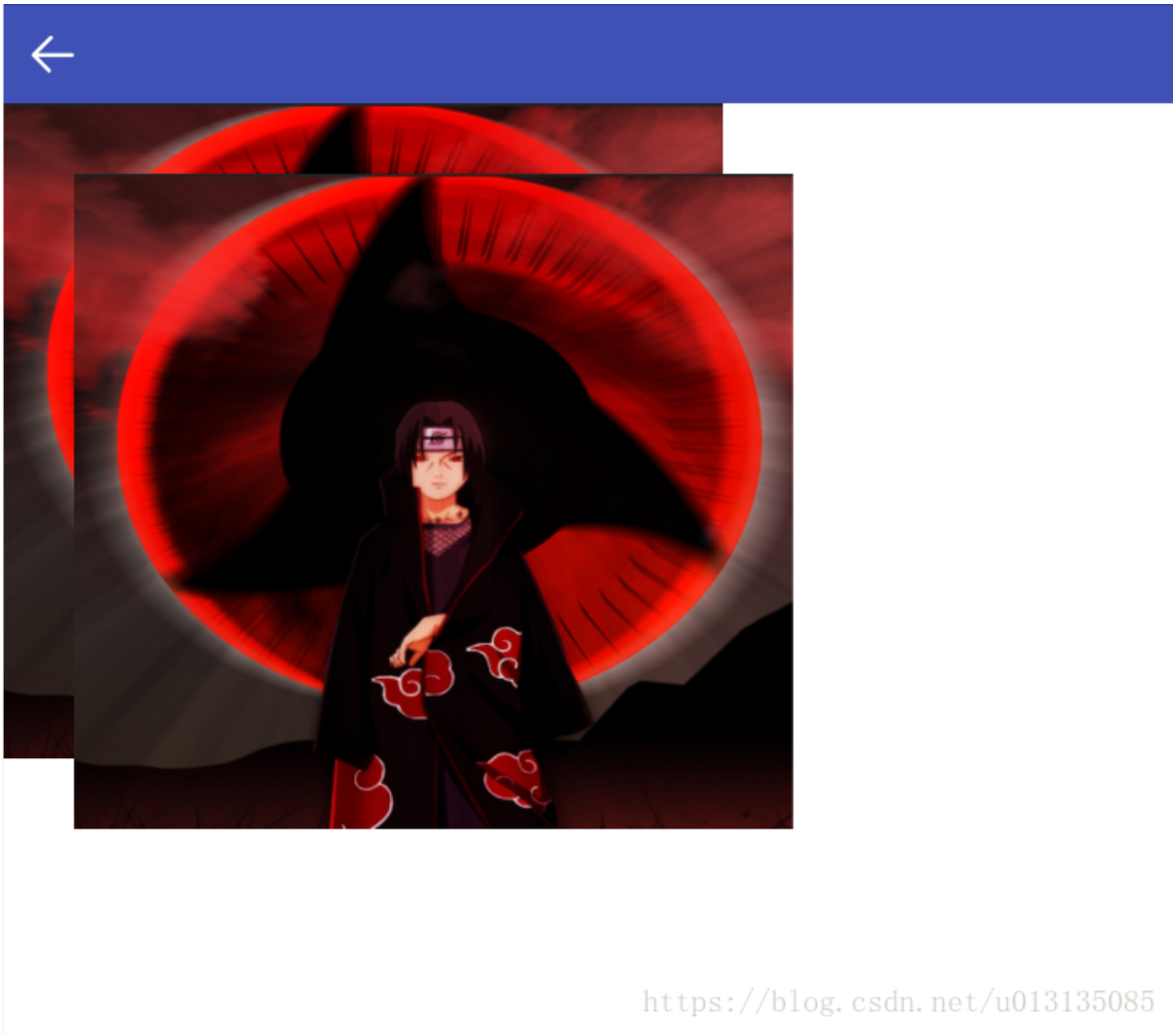


drawBitmap(Bitmap bitmap, Matrix matrix, Paint paint)方法是通过矩阵对图片进行一些变换处理。如下：

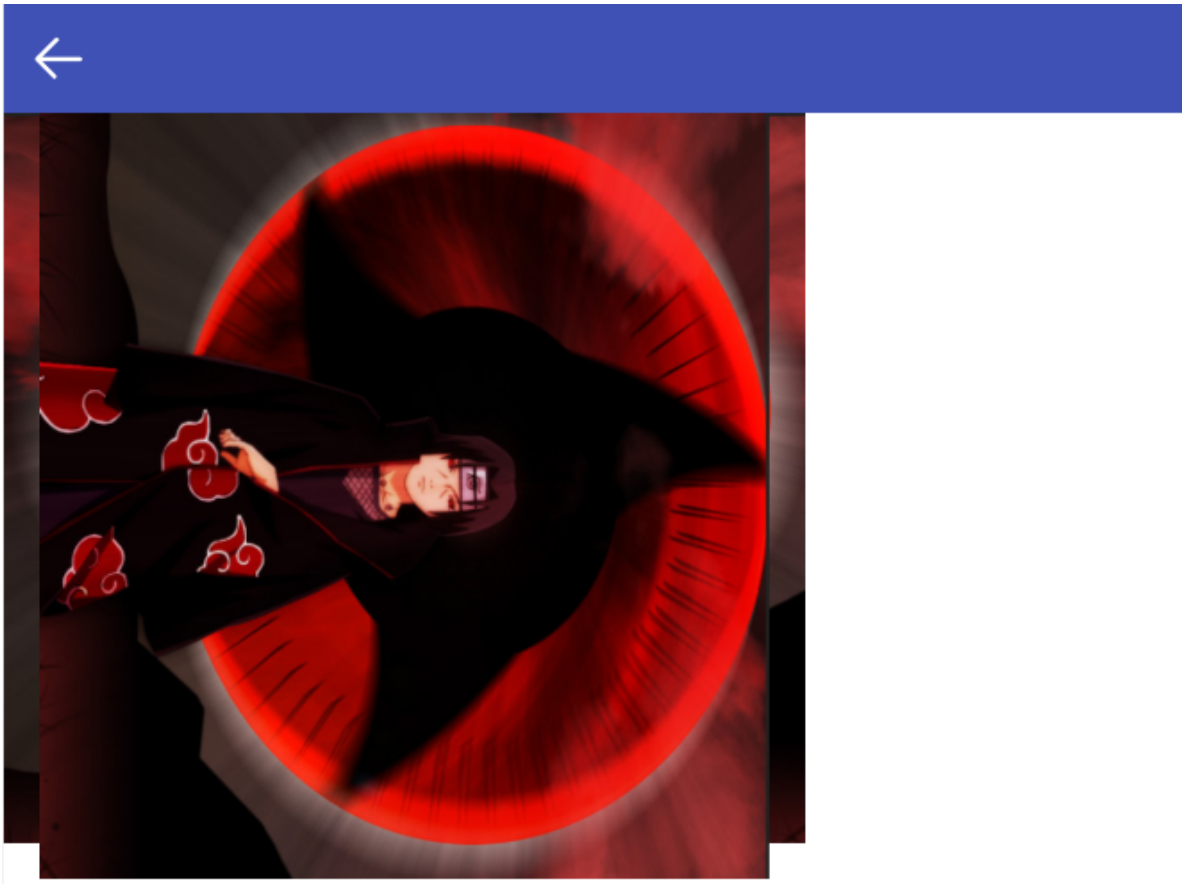
```
    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        // 画出原图像
        canvas.drawBitmap(mBitmap, 0, 0, null);
        Matrix matrix = new Matrix();
        matrix.postScale(0.5f, 0.5f);
        canvas.drawBitmap(mBitmap, matrix,null);
    }
```



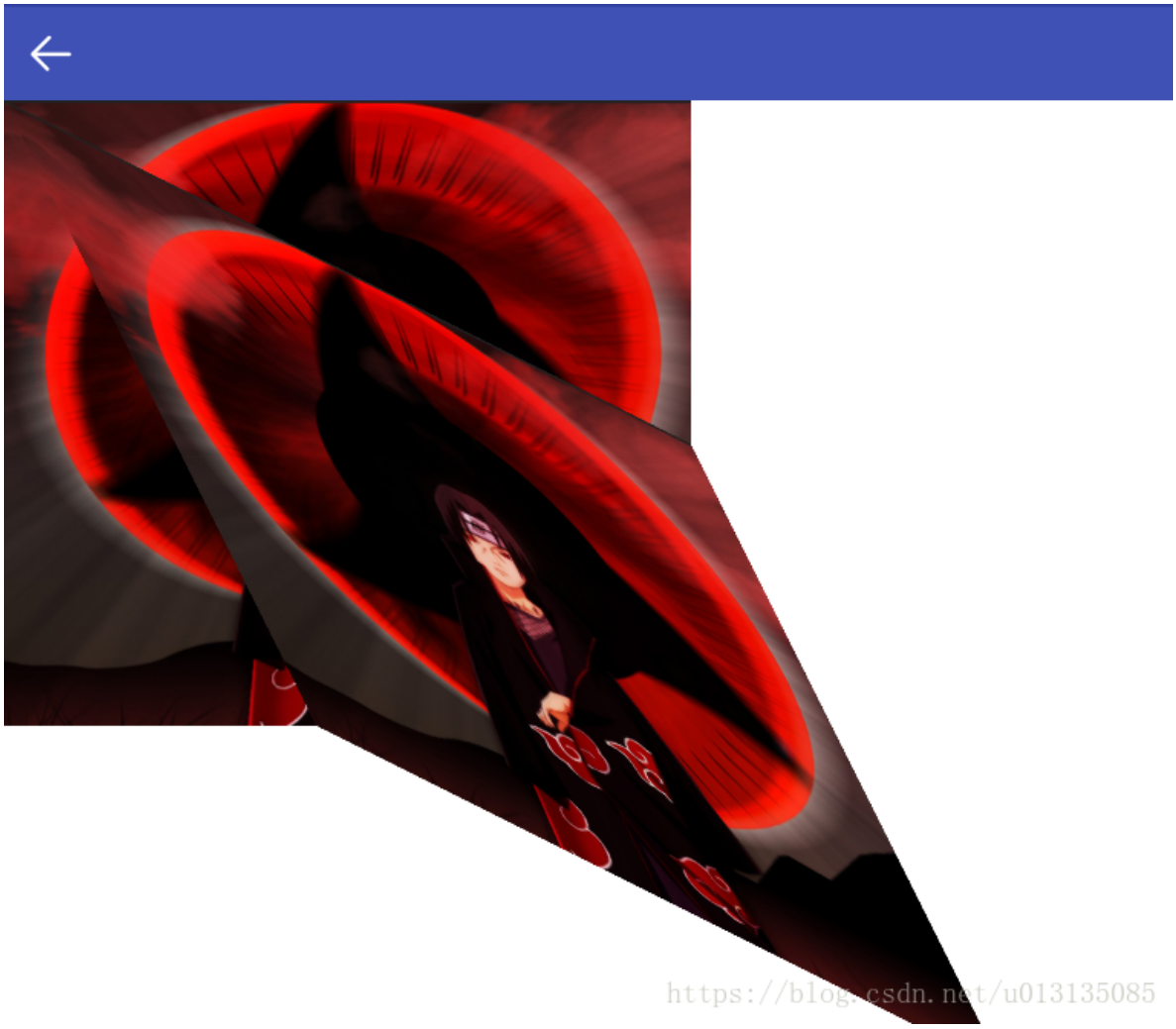

```
  @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        // 画出原图像
        canvas.drawBitmap(mBitmap, 0, 0, null);
        Matrix matrix = new Matrix();
        matrix.postTranslate(100f, 100f);
        canvas.drawBitmap(mBitmap, matrix,null);
    }
```




```
@Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        // 画出原图像
        canvas.drawBitmap(mBitmap, 0, 0, null);
        Matrix matrix = new Matrix();
        matrix.postRotate(90, mBitmap.getWidth()*0.5f,mBitmap.getHeight()*0.5f);
        canvas.drawBitmap(mBitmap, matrix,null);
    }
```




```
@Override
  protected void onDraw(Canvas canvas) {
      super.onDraw(canvas);
      // 画出原图像
      canvas.drawBitmap(mBitmap, 0, 0, null);
      Matrix matrix = new Matrix();
      matrix.postSkew(0.5f,0.5f);
      canvas.drawBitmap(mBitmap, matrix,null);
  }
```



二、 BitmapFactory

在把图片转化为bitmap时，遇到大一些的图片，我们经常会遇到OOM(Out Of Memory)的问题。因此需要把图片进行缩放。

Options

这就用到了我们上面提到的BitmapFactory.Options这个类，下面逐一介绍Options的参数 。

Options.inJustDecodeBounds

BitmapFactory.Options这个类，有一个字段叫做 inJustDecodeBounds ，如果我们把它设为true，那么BitmapFactory.decodeFile(String path, Options opt)并不会真的返回一个Bitmap给你，它仅仅会把它的宽，高取回来给你，这样就不会占用太多的内存，也就不会那么频繁的发生OOM了。

```
BitmapFactory.Options options = new BitmapFactory.Options();

options.inJustDecodeBounds = true;

Bitmap bmp = BitmapFactory.decodeFile(path, options);/* 这里返回的bmp是null */
```

之后，options.outWidth 和 options.outHeight就是我们想要的宽和高了。可以根据需要进行图片的缩略，在文章最后附上图片缩放代码 .

Options.inSampleSize

图片的缩放比例，该参数需要自己通过计算得到，一般通过options.outHeight和 options. outWidth 获取的宽高和自己想要到得图片宽高计算出缩放比例。下面会给出缩放代码 .

为了节约内存我们还可以使用下面的几个字段：

```
options.inDither=false;     /*不进行图片抖动处理*/
options.inPreferredConfig=null;  /*设置让解码器以最佳方式解码*/

/* 下面两个字段需要组合使用 */

options.inPurgeable = true;

options.inInputShareable = true;
```

BitmapFactory的一些方法

这些方法可以用于从不同的数据源解析、创建Bitmap对象

1. BitmapFactory.decodeByteArray(byte[] data, int offset, int length)
   从指定字节数组的offset位置开始，将长度为length的字节数据解析成Bitmap对象；
2. BitmapFactory.decodeFile(String path)
   该方法将指定路径的图片转成Bitmap;
3. BitmapFactory.decodeFile(String path, Options options)
   该方法使用options的变量信息，将指定路径的图片转成Bitmap；
4. decodeResource()
   可以将/res/drawable/内预先存入的图片转换成Bitmap对象；
5. decodeStream()
   方法可以将InputStream对象转换成Bitmap对象。；

图片缩放代码：

```
private Bitmap decodeThumbBitmapForFile(String path, int viewWidth, int
viewHeight){
        BitmapFactory.Options options = new BitmapFactory.Options();
        //设置为true,表示解析Bitmap对象，该对象不占内存
        options.inJustDecodeBounds = true;
        BitmapFactory.decodeFile(path, options);
        //设置缩放比例
        options.inSampleSize = computeScale(options, viewWidth, viewHeight);

        //设置为false,解析Bitmap对象加入到内存中
        options.inJustDecodeBounds = false;
        Log.e(TAG, "get Iamge form file,  path = " + path);
        //返回Bitmap对象
        return BitmapFactory.decodeFile(path, options);
    }
```

computeScale() 计算缩放比例：

```
private int computeScale(BitmapFactory.Options options, int viewWidth, int
viewHeight){
        int inSampleSize = 1;
        if(viewWidth == 0 || viewWidth == 0){
            return inSampleSize;
        }
```

```
        int bitmapWidth = options.outWidth;
        int bitmapHeight = options.outHeight;

       //假如Bitmap的宽度或高度大于我们设定图片的View的宽高，则计算缩放比例
        if(bitmapWidth > viewWidth || bitmapHeight > viewWidth){
            int widthScale = Math.round((float) bitmapWidth / (float)
viewWidth);
            int heightScale = Math.round((float) bitmapHeight / (float)
viewWidth);

        //为了保证图片不缩放变形，我们取宽高比例最小的那个
            inSampleSize = widthScale < heightScale ? widthScale : heightScale;
        }
        return inSampleSize;
    }
```