[Android] 给图像添加相框、圆形圆角显示图片、图像合成知识

原创 Eastmount 最后发布于2014-10-31 03:09:44 阅读数 9059 ☆ 收藏 展开

前一篇文章讲述了Android触屏setOnTouchListener实现突破缩放、移动、绘制和添加水印,继续我的"随手拍"项目完成给图片添加相框、圆形圆角显示图片和图像合成的功能介绍.希望文章对大家有所帮助.

一. 打开图片和显示assets文件中图片

首先,对XML中activity_main.xml进行布局,通过使用RelativeLayout相对布局完成(XML代码后面附).然后,在Mainctivity.java中public class MainActivity extends Activity函数添加代码如下,添加点击按钮监听事件:

```
//控件
private Button openImageBn;             //打开图片
private Button showImageBn;             //显示assets资源图片
private Button showImageBn1;            //模式1加成
private Button showImageBn2;            //模式2加成
private Button roundImageBn;            //圆角图片
private ImageView imageShow;            //显示图片
//自定义变量
private Bitmap bmp;                     //原始图片
private final int IMAGE_OPEN = 0;       //打开图片
private Canvas canvas;                  //画布
private Paint paint;                    //画刷

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
        //打开图片
        openImageBn = (Button)findViewById(R.id.button1);
        imageShow = (ImageView) findViewById(R.id.imageView1);
        openImageBn.setOnClickListener(new OnClickListener() {
                @Override
                public void onClick(View v) {
                        Intent intent = new Intent(Intent.ACTION_PICK,
                                        android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
                        startActivityForResult(intent, IMAGE_OPEN);
                }
        });
    if (savedInstanceState == null) {
        getFragmentManager().beginTransaction()
                .add(R.id.container, new PlaceholderFragment())
                .commit();
    }
}
//打开图片
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
         super.onActivityResult(requestCode, resultCode, data);
         if(resultCode==RESULT_OK && requestCode==IMAGE_OPEN) {
        Uri imageFileUri = data.getData();
        DisplayMetrics dm = new DisplayMetrics();
```

```
        getWindowManager().getDefaultDisplay().getMetrics(dm);
        int width = dm.widthPixels;    //手机屏幕水平分辨率
        int height = dm.heightPixels;  //手机屏幕垂直分辨率
                                                                try {
                //载入图片尺寸大小没载入图片本身 true
                BitmapFactory.Options bmpFactoryOptions = new BitmapFactory.Options();
                bmpFactoryOptions.inJustDecodeBounds = true;
                bmp = BitmapFactory.decodeStream(getContentResolver().openInputStream(imageFileUri), null,
bmpFactoryOptions);
                int heightRatio = (int)Math.ceil(bmpFactoryOptions.outHeight/(float)height);
                int widthRatio = (int)Math.ceil(bmpFactoryOptions.outWidth/(float)width);
                //inSampleSize表示图片占原图比例 1表示原图                if(heightRatio>1&&widthRatio>1) {
                    if(heightRatio>widthRatio) {
                        bmpFactoryOptions.inSampleSize = heightRatio;
                    }
                    else {
                        bmpFactoryOptions.inSampleSize = widthRatio;
                    }
                }
                //图像真正解码 false
                bmpFactoryOptions.inJustDecodeBounds = false;
                bmp = BitmapFactory.decodeStream(getContentResolver().openInputStream(imageFileUri), null,
bmpFactoryOptions);
                imageShow.setImageBitmap(bmp);             }
            catch(FileNotFoundException e) {
                e.printStackTrace();
            }
        }  //end if
}
```

上面点击"打开"按钮能实现打开图片,而在讲述为图片添加边框时,它其实就是通过两张或多张图片的合成实现的.

在jacpy.may《Android图片处理总结》文档中建议图片不要放在drawable目录下,因为屏幕分辨率会影响图片的大小.最好放在assets目录里,它代表应用无法直接访问的原生资源(通常加载PNG透明图实现边框合成),只能以流的方式读取并且小于1M.

读取assets文件夹中图片的方法如下,首先手动添加PNG图片至assets目录,然后在omCreate函数中添加如下代码:

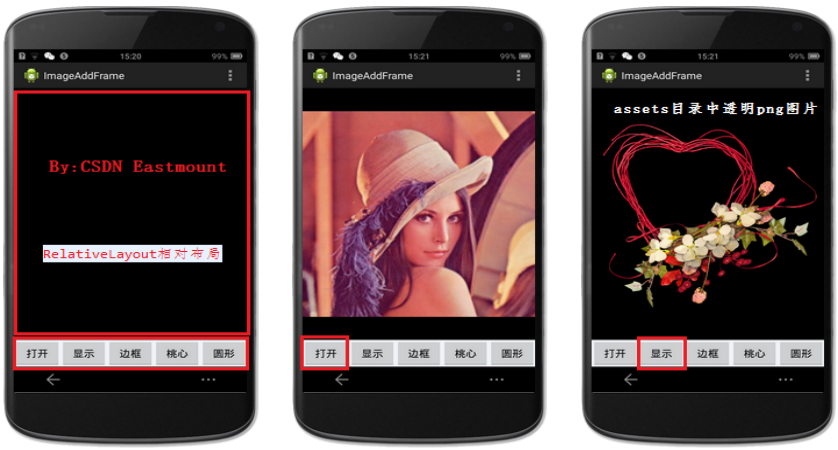
```
//显示assets中图片
showImageBn = (Button)findViewById(R.id.button2);
showImageBn.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
                Bitmap bitmap = getImageFromAssets("image01.png");
                imageShow.setImageBitmap(bitmap);
        }
});
```

再通过自定义函数getImageFromAssets实现获取图片"image01.png":

```
//获取assets中资源并转换为Bitmap
private Bitmap getImageFromAssets(String fileName)
{
        //Android中使用assets目录存放资源,它代表应用无法直接访问的原生资源
        Bitmap imageAssets = null;
        AssetManager am = getResources().getAssets();
        try {
                InputStream is = am.open(fileName);
                imageAssets = BitmapFactory.decodeStream(is);
```

```
                is.close();
                                } catch(IOException e) {
                e.printStackTrace();
        }
        return imageAssets;
}
```

显示效果如下图所示：



其中XML代码如下：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.touchimagetest.MainActivity"
    tools:ignore="MergeRootFrame" >
    <!-- 底部添加按钮 -->
    <RelativeLayout
        android:id="@+id/MyLayout_bottom"
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="50dp"
        android:layout_alignParentBottom="true"
        android:gravity="center">
            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:orientation="horizontal"
                android:layout_alignParentBottom="true" >
                        <Button
                                android:id="@+id/button1"
                                android:layout_width="wrap_content"
                                android:layout_height="match_parent"
                                android:layout_weight="1"
                                android:text="打开" />
                        <Button
                                android:id="@+id/button2"
                                android:layout_width="wrap_content"
                                android:layout_height="match_parent"
                                android:layout_weight="1"
                                android:text="显示" />
                        <Button
                                android:id="@+id/button3"
                                android:layout_width="wrap_content"
```

```
                android:layout_height="match_parent"
                android:layout_weight="1"
                android:text="边框" />
                                                        <Button
                android:id="@+id/button4"
                android:layout_width="wrap_content"
                android:layout_height="match_parent"
                android:layout_weight="1"
                android:text="桃心" />
            <Button
                android:id="@+id/button5"
                android:layout_width="wrap_content"
                android:layout_height="match_parent"
                android:layout_weight="1"
                android:text="圆形" />
        </LinearLayout>
    </RelativeLayout>
    <!-- 顶部显示图片 -->
    <RelativeLayout
        android:id="@+id/Content_Layout"
        android:orientation="horizontal"
        android:layout_above="@id/MyLayout_bottom"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:background="#000000"
        android:gravity="center">
        <ImageView
                android:id="@+id/imageView1"
                android:layout_width="fill_parent"
                android:layout_height="fill_parent"
                android:layout_gravity="center_horizontal" />
    </RelativeLayout>
</RelativeLayout>
```

二. 添加相框与图片合成

然后开始完成图片合成的工作,这里我采用两种方法完成.继续在onCreate函数中添加代码:

```
//模式1合成图片
showImageBn1 = (Button)findViewById(R.id.button3);
showImageBn1.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
                Bitmap bitmap = getImageFromAssets("image01.png");
                addFrameToImage(bitmap);
        }
});
```

通过自定义函数addFrameToImage实现加载图片合成.首先创建一个空的可变为图对象,它的大小和配置与打开的图像相同,随后构建一个Canvas对象和一个Paint对象,在画布上绘制第一个位图对象,它成为了合成操作的目标.

现在设置Paint对象上的过渡模式,通过传入一个定义操作模式的常量,实例化一个新的PorterDuffXfermode对象.然后在Canvas对象上绘制第二个位图对象,并将ImageView设置为新的位图对象.代码如下:

```
//图片合成1
private void addFrameToImage(Bitmap bm) //bm原图(前景) bm资源图片(背景)
{
        Bitmap drawBitmap =Bitmap.createBitmap(bmp.getWidth(),
```

```
                        bmp.getHeight(), bmp.getConfig());          canvas = new Canvas(drawBitmap);
        paint = new Paint();
        canvas.drawBitmap(bmp, 0, 0, paint);
        paint.setXfermode(new PorterDuffXfermode(android.
                        graphics.PorterDuff.Mode.LIGHTEN));
        //对边框进行缩放
    int w = bm.getWidth();
    int h = bm.getHeight();
    //缩放比 如果图片尺寸超过边框尺寸 会自动匹配
    float scaleX = bmp.getWidth()*1F / w;
    float scaleY = bmp.getHeight()*1F / h;
    Matrix matrix = new Matrix();
    matrix.postScale(scaleX, scaleY);   //缩放图片
    Bitmap copyBitmap =  Bitmap.createBitmap(bm, 0, 0, w, h, matrix, true);
        canvas.drawBitmap(copyBitmap, 0, 0, paint);
        imageShow.setImageBitmap(drawBitmap);
}
```

第二种方法是参照《Android多媒体开发高级编程》,但是它图片十四合成效果不是很好:

```
//模式2合成图片
showImageBn2 = (Button)findViewById(R.id.button4);
showImageBn2.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
                Bitmap bitmap = getImageFromAssets("image07.png");
                //第二种合成方法
                imageShow.setImageBitmap(addFrameToImageTwo(bitmap));
        }
});
```

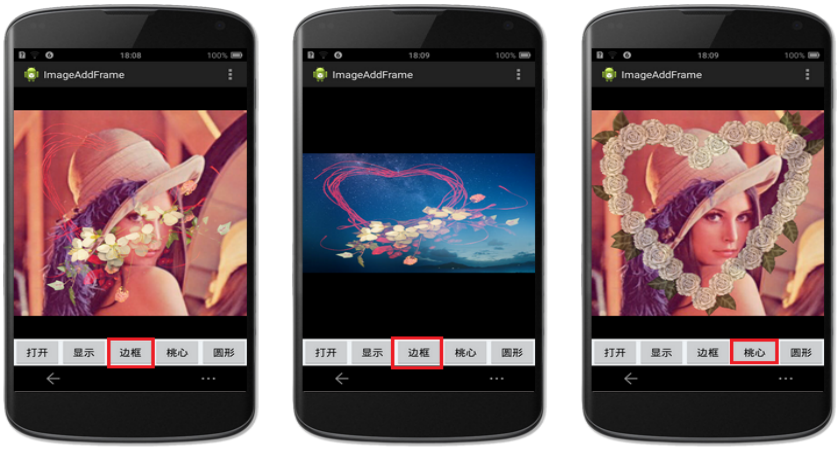
然后通过自定义函数实现:

```
//图片合成
private Bitmap addFrameToImageTwo(Bitmap frameBitmap) //bmp原图 frameBitmap资源图片(边框)
{
        //bmp原图 创建新位图
        int width = bmp.getWidth();
    int height = bmp.getHeight();
    Bitmap drawBitmap =Bitmap.createBitmap(width, height, Config.RGB_565);
        //对边框进行缩放
    int w = frameBitmap.getWidth();
    int h = frameBitmap.getHeight();
    float scaleX = width*1F / w;        //缩放比 如果图片尺寸超过边框尺寸 会自动匹配
    float scaleY = height*1F / h;
    Matrix matrix = new Matrix();
    matrix.postScale(scaleX, scaleY);   //缩放图片
    Bitmap copyBitmap =  Bitmap.createBitmap(frameBitmap, 0, 0, w, h, matrix, true);

    int pixColor = 0;
    int layColor = 0;
    int newColor = 0;

    int pixR = 0;
    int pixG = 0;
    int pixB = 0;
    int pixA = 0;

    int newR = 0;
```

```
    int newG = 0;       int newB = 0;
    int newA = 0;

    int layR = 0;
    int layG = 0;
    int layB = 0;
    int layA = 0;

    float alpha = 0.8F;
    float alphaR = 0F;
    float alphaG = 0F;
    float alphaB = 0F;

    for (int i = 0; i < width; i++)
    {
        for (int k = 0; k < height; k++)
        {
            pixColor = bmp.getPixel(i, k);
            layColor = copyBitmap.getPixel(i, k);
            // 获取原图片的RGBA值
            pixR = Color.red(pixColor);
            pixG = Color.green(pixColor);
            pixB = Color.blue(pixColor);
            pixA = Color.alpha(pixColor);
            // 获取边框图片的RGBA值
            layR = Color.red(layColor);
            layG = Color.green(layColor);
            layB = Color.blue(layColor);
            layA = Color.alpha(layColor);
            // 颜色与纯黑色相近的点
            if (layR < 20 && layG < 20 && layB < 20)
            {
                alpha = 1F;
            }
            else
            {
                alpha = 0.3F;
            }
            alphaR = alpha;
            alphaG = alpha;
            alphaB = alpha;
            // 两种颜色叠加
            newR = (int) (pixR * alphaR + layR * (1 - alphaR));
            newG = (int) (pixG * alphaG + layG * (1 - alphaG));
            newB = (int) (pixB * alphaB + layB * (1 - alphaB));
            layA = (int) (pixA * alpha + layA * (1 - alpha));
            // 值在0~255之间
            newR = Math.min(255, Math.max(0, newR));
            newG = Math.min(255, Math.max(0, newG));
            newB = Math.min(255, Math.max(0, newB));
            newA = Math.min(255, Math.max(0, layA));
            //绘制
            newColor = Color.argb(newA, newR, newG, newB);
            drawBitmap.setPixel(i, k, newColor);
        }
    }
    return drawBitmap;
}
```

它的运行效果如下所示,其中前2附图是方法一,但是它的合成效果不是很优秀,而第三张图是第二种方法,但是它的响应时

间稍微要长些.



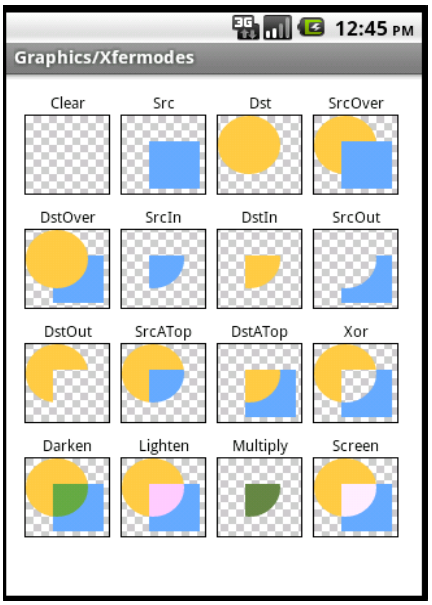
在第一种方法通过PoterDuffXfermode类作为过渡模式,该类因Thomas Porter和Tom Duff而得名,他们于1984年在ACM SIGGRAPH计算机图形学发表“Compositing digital images(合成数字图像)”的文章,它介绍了彼此重叠绘制图像的不同规则.这些规则定义了哪些图像的哪些部分将出现在结果输出中.

在Android的PorterDuff.Mode类中列举了Porter和Duff及其他更多人制定的规则.

android.graphics.PorterDuff.Mode.SRC:此规则意味着只绘制源图像,当前它正是应用此规则的Paint对象.

android.graphics.PorterDuff.Mode.DST:此规则意味着只显示目标图像,在已有画布上的初始图像.

如下图所示,定义Mode值如下:



其中,有4个规则定义了当一幅图像放置在另一幅图像上时如何合成这两幅图像,它是我们经常使用的值:

android.graphics.PorterDuff.Mode.LIGHTEN:获得每个位置上两幅图像中最亮的像素并显示.

android.graphics.PorterDuff.Mode.DARKEN:获得每个位置上两幅图像中最暗的像素并显示.

android.graphics.PorterDuff.Mode.MULTIPLY:将每个位置的两个像素相乘,除以255,使用该值创建一个新的像素进行显示.结果颜色=顶部颜色*底部颜色/255.

android.graphics.PorterDuff.Mode.SCREEN:反转每个颜色,执行相同操作.结果颜色=255-(((255-顶部颜色)*(255-底部颜色))/255)

三. 圆形和圆角矩形显示图片

最后讲述如何实现圆形和圆角矩形显示图片,在onCreate函数中添加如下代码:

```
//圆角合成图片
roundImageBn = (Button)findViewById(R.id.button5);
```

```
roundImageBn.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
                imageShow.setImageBitmap(getRoundedCornerBitmap(bmp) );
        }
});
```

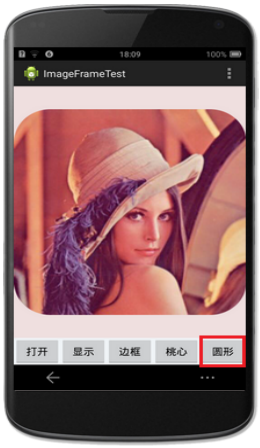
然后通过自定义函数getRoundedCornerBitmap实现圆形：

```
//生成圆角图片
private Bitmap getRoundedCornerBitmap(Bitmap bitmap)
{
        Bitmap roundBitmap = Bitmap.createBitmap(bitmap.getWidth(),
         bitmap.getHeight(), Config.ARGB_8888);
     Canvas canvas = new Canvas(roundBitmap);
     int color = 0xff424242;
     Paint paint = new Paint();
     //设置圆形半径
     int radius;
     if(bitmap.getWidth()>bitmap.getHeight()) {
         radius = bitmap.getHeight()/2;
     }
     else {
         radius = bitmap.getWidth()/2;
     }
     //绘制圆形
     paint.setAntiAlias(true);
     canvas.drawARGB(0, 0, 0, 0);
     paint.setColor(color);
     canvas.drawCircle( bitmap.getWidth()/ 2, bitmap.getHeight() / 2, radius, paint);
     paint.setXfermode(new PorterDuffXfermode(Mode.SRC_IN));
     canvas.drawBitmap(bitmap, 0, 0, paint);
     return roundBitmap;
}
```

同样,如果把该函数里面内容替换即可实现圆形矩形显示图片：

```
private Bitmap getRoundedCornerBitmap(Bitmap bitmap)
{
        //绘制圆角矩形
        Bitmap roundBitmap = Bitmap.createBitmap(bitmap.getWidth(),
         bitmap.getHeight(), Config.ARGB_8888);
     Canvas canvas = new Canvas(roundBitmap);
     int color = 0xff424242;
     Paint paint = new Paint();
     Rect rect = new Rect(0, 0, bitmap.getWidth(), bitmap.getHeight());
     RectF rectF = new RectF(rect);
     float roundPx = 80;      //转角设置80
     //绘制
     paint.setAntiAlias(true);
     canvas.drawARGB(0, 0, 0, 0);
     paint.setColor(color);
     canvas.drawRoundRect(rectF, roundPx, roundPx, paint);
     paint.setXfermode(new PorterDuffXfermode(Mode.SRC_IN));
     canvas.drawBitmap(bitmap, rect, rect, paint);
}
```

运行结果如下图所示：



总结：

该文章主要讲述如何给图像增加相框,圆角显示图像和图像合成的介绍.里面主要通过源码并有详细的过程,为什么要写这篇文章？因为在图像处理中我认为这种添加边框、改变边框、图片合成都属于同一种类型的变化和渲染.该图像处理软件还没有整合,推荐大家看下面资料中两本书.

最后希望文章对大家有所帮助,如果有不足或错误的地方请见谅！不论如何,我觉得这篇文章自己写得不错,自己先给自己一个赞吧！加油\(^o^)/~

下载地址：http://download.csdn.net/detail/eastmount/8102845

源码基本格式如下图所示：

```
protected void onCreate(Bundle savedInstanceState) {

//打开图片
protected void onActivityResult(int requestCode, int resultCode, Intent data) {

//获取assets中资源并转换为Bitmap
private Bitmap getImageFromAssets(String fileName)

//图片合成1
private void addFrameToImage(Bitmap bm) //bmp原图(前景) bm资源图片(背景)

//图片合成2
private Bitmap addFrameToImageTwo(Bitmap frameBitmap) //bmp原图 frameBitmap资源图片(边框)

//生成圆角图片
private Bitmap getRoundedCornerBitmap(Bitmap bitmap)
```

(By:Eastmount 2014-10-31 夜3点 http://blog.csdn.net/eastmount)

参考资料与推荐博文：

1.最该感谢的是两本书的作者《Android多媒体开发高级编程》和《Android图片处理总结 著:jacpy.may》,网上很多资料都是它们.

2.android图像处理系列之六 - - 给图片添加边框（下） - 图片叠加

作者-SJF0115 他是转载了该书的一些文章,也非常不错.

3.Android 图片合成：添加蒙板效果 不规则相框 透明度渐变效果的实现

作者-HappyDelano 非常不错的文章,讲述了4张图实现桃心显示的效果.

4.Android图片合成 作者-johnlxj 讲述了图片合成的实现过程.

5.Android 完美实现图片圆角和圆形(对实现进行分析)

作者-鸿洋_ 该作者很多android文章都非常不错

6.android 画图之setXfermode 作者-lipeng88213 推荐起链接的Android图片倒影

7.Android ImageView点击选中后添加边框 作者-黑米粥 该方法在切换图片中实用

8.android 轻松实现在线即时聊天【图片、语音、表情、文字】 作者-anonymousblogs