宝贵建议请发送至：wangzhenyang@itcast.cn

黑马程序员 itheima.com -编程，始于黑马

Android 课程同步笔记

Alpha 0.01 版

By 阳哥

Android-图片处理

1.加载大图片（★★★）

Android 虚拟机默认为每个应用分配的堆内存空间是 16M，当加载大图片时，加载图片需要的内存空间不是按图片的大小来算的，而是按像素点的多少来算的。图片加载到内存中需要把每一个像素都加载到内存中. 所以对内存的要求非常高，一不小心就会造成 OOM(OutOfMemoryError) 内存溢出致命错误。

假设：

当前有一张图片，大小仅为 1M，但是其规格为 3648*2736，现在需要加载此图片总像素数=3648*2736=9980928

三种像素单位如下：

ARGB_4444 : 2bytes

ARGB_8888 : 4bytes

RGB_565 : 4bytes

假设现在像素采用 ARGB_4444 标准，则其占用的总空间为：

图片占用空间=总像素数 *像素的单位

=9980928 * 2bytes

=19961856bytes

=19M>16M　　OOM 内存溢出

解决方案：

Java 代码可以对图片进行比例缩放

假设：

图片的宽和高: 3648 * 2736

屏幕的宽和高: 320 * 480

计算缩放比：

宽度缩放比例: 3648 / 320 = 11

高度缩放比例: 2736 / 480 = 5

比较宽和高的缩放比例，哪一个大用哪一个进行缩放

缩放后的图片：

3648 / 11 = 331

2736 / 11 = 248

缩放后图片的宽和高: 331* 248

331* 248=882088 * 2bytes=160K

1.1 实现图片的缩放加载

Tips：这里只给出核心代码，用于演示加载大图片的原理。

普通方法加载图片代码清单：

```
public void load(View v) {
      String path = etPath.getText().toString().trim();
      // 根据路径得到图片对象
      Bitmap bitmap = BitmapFactory.decodeFile(path);
      // 把图片展示到 ImageView 控件上.
      ivIcon.setImageBitmap(bitmap);
   }
```

采用缩放方式加载图片：

```
public void scaleLoad(View v) {
	String path = etPath.getText().toString().trim();
	// 得到图片的宽和高
	Options opts = new Options();
	opts.inJustDecodeBounds = true; // 加载器不加载图片，而是把图片的 out(宽和高)的字段信息取出来
	BitmapFactory.decodeFile(path, opts);
	int imageWidth = opts.outWidth;
	int imageHeight = opts.outHeight;
	System.out.println("图片的宽和高: " + imageWidth + " * " + imageHeight);

	// 得到屏幕的宽和高
	Display display = getWindowManager().getDefaultDisplay();
	int screenWidth = display.getWidth();
	int screenHeight = display.getHeight();

	System.out.println("屏幕的宽和高: " + screenWidth + " * " + screenHeight);

	// 计算缩放比例
	int widthScale = imageWidth / screenWidth;
	int heightScale = imageHeight / screenHeight;

	int scale = widthScale > heightScale ? widthScale : heightScale;
	System.out.println("缩放比例为: " + scale);

	// 使用缩放比例进行缩放加载图片
	opts.inJustDecodeBounds = false; // 加载器就会返回图片了
	opts.inSampleSize = scale;
	Bitmap bm = BitmapFactory.decodeFile(path, opts);

	// 显示在屏幕上
	ivIcon.setImageBitmap(bm);
    }
```

2.图片加水印（★★）

Android 提供了两个类 Canvas 和 Paint：

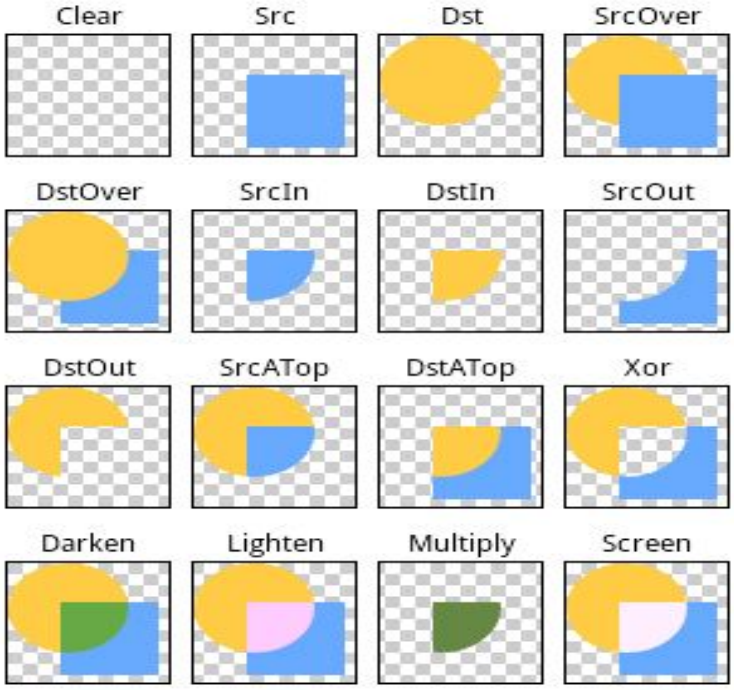
- Canvas 画画板,用于绘制各种图形(点，线，圆，矩形等等)
- Paint 画笔,和 Canvas 搭配使用,用于指定绘制的颜色，线条的粗细，过渡，渐变等效果。

使用方法：

```
Canvas.drawBitmap(
        bitmap,      // 被绘制的图片
        x,   // 图片的左上角在 x 轴上的位置
        y,   // 图片的左上角在 y 轴上的位置
        paint);  // 画笔，绘制图片时可以忽略. 设置为 null
```

2.1 图片加水印代码实现

- 需求：图片加水印，即准备一张原图和一张水印图片，然后对两个图片进行合成。
- 前提知识：两张图片的合成有如下几种方式，在该案例中我们为了让两张图片都显示，因此我们选择 Darken 方法。图片合成的几种方式见下图。



现在有两张图片，1、原图：

2、水印：

Tips：这里我会新创建一个 Android 工程，并把这两个资源加入到 drawable 目录中。关于创建工程的步骤等简单操作在以后的文档中就一笔带过，我们直接将重点放在核心代码

上。

图片合成代码清单：

```
//图片加水印
    public void combine(View view){
        //获取用于显示图片的 ImageView 控件
        iv_combine = (ImageView) findViewById(R.id.iv_combine);
        //获取原始图片
        Bitmap bm_android =
BitmapFactory.decodeResource(getResources(), R.drawable.android);
        //获取水印图片
        Bitmap bm_logo = BitmapFactory.decodeResource(getResources(),
R.drawable.logo);
        //创建一个新的空白 Bitmap 对象，用于合成后的额图片
        Bitmap bm_new = Bitmap.createBitmap(bm_android.getWidth(),
bm_android.getHeight(), Config.ARGB_8888);
        //在上一步创建的 Bitmap 的基础上新建一个画布对象
        Canvas canvas = new Canvas(bm_new);
        //将原始图片绘制到画布上，第二个参数是左边，第三个参数是上边距，第
四个参数是 Pain 对象，这里设置为 null
        canvas.drawBitmap(bm_android, 0, 0, null);
        //新创建一个 Pain 对象
        Paint paint = new Paint();
        //设置两张图片的相交模式：Darken，注意：Porter、Duff 是两个发明人
的合成单词，本身并没有任何意义
        paint.setXfermode(new PorterDuffXfermode(Mode.DARKEN));
        //设置水印的左边距，这里设置为右下角，右边距为 10
        float left = bm_android.getWidth()-bm_logo.getWidth()+10;
        //设置水印的上边距，这里设置为距离底部 10
        float top = bm_android.getHeight()-bm_logo.getHeight()+10;
        //在画布上将水印绘制上去
        canvas.drawBitmap(bm_logo, left, top, paint);
        //在控件中显示合成后的图片
        iv_combine.setImageBitmap(bm_new);

    }
```

运行上面的代码，效果图如下： 我们发现在 Android 图片的右下角成功添加了黑马的

LOGO 作为水印。

3.图片特效（★★）

图片的特效包括，图形的缩放、镜面、倒影、旋转、位移等。图片的特效是将原图的图形矩阵乘以一个特效矩阵，形成一个新的图形矩阵来实现的。

Matrix 维护了一个 3*3 的矩阵去更改像素点的坐标。

图形的默认矩阵用数组表示为:

{ 1, 0, 0,　　表示向量 x = 1x + 0y + 0z

0, 1, 0,　　表示向量 y = 0x + 1y + 0z

0, 0, 1 }　表示向量 z = 0x + 0y + 1z

通过更改图形矩阵的值，可以做出缩放/镜面/倒影等图片特效。

下面分别给出各种特效实现的代码，在代码中会有详细的注释。

3.1 缩放

- 矩阵示例：

{ 2, 0, 0,

0, 1, 0,

0, 0, 1 }

- 意义：x 轴所有的像素点放大 2 倍，展现的效是: 图片宽度 x2

代码清单：

```
// 缩放
    public void scale() {
        //获取原图
        Bitmap bm = BitmapFactory.decodeResource(getResources(),
R.drawable.logo);
        //创建一个新的 Bitmap 对象，宽高跟原始图片保持一致
        Bitmap newBm = Bitmap.createBitmap(bm.getWidth(),
bm.getHeight(), Config.ARGB_8888);
        //以新 Bitmap 构造一个画布
        Canvas canvas = new Canvas(newBm);
        //创建一个 Matrix 对象
        Matrix matrix = new Matrix();
        //矩阵值
        float[] values = new float[] { 2, 0, 0, //x=2*x+0*y+0*z
                                       0, 1, 0, //y=0*x+1*y+0*z
                                       0, 0, 1 };//z=0*x+0*y+1*z
        matrix.setValues(values);
        //将原始图片乘以矩阵后画到画布上
        canvas.drawBitmap(bm, matrix, null);
```

```
        //将新 bitmap 输出到 ImageView 控件
        iv.setImageBitmap(newBm);
    }
```

3.2 镜面

- 矩阵示例：

  { -1, 0, 0,　　x 坐标变为复数，代表以 y 轴为镜面成像

  0, 1, 0,

  0, 0, 1 }

- 意义：x 轴所有的像素点沿负数方向反过来，展现的效果是：镜面。

```
// 镜面
    public void mirror() {
        Bitmap bm = BitmapFactory.decodeResource(getResources(),
R.drawable.logo);
        Bitmap newBm = Bitmap.createBitmap(bm.getWidth(),
bm.getHeight(), Config.ARGB_8888);
        Canvas canvas = new Canvas(newBm);
        Matrix matrix = new Matrix();
//因为镜面成像以后，图片 x 轴全为负数，跑出了屏幕范围，因此为了看到效果把图像往 x 轴正方向移动一个图片的宽度
        matrix.postTranslate(bm.getWidth(), 0);
        float[] values = new float[] { -1, 0, 0, //x=-1*x+0*y+0*z
                                       0, 1, 0, //y=0*x+1*y+0*z
                                       0, 0, 1 };//z=0*x+0*y+1*z
        matrix.setValues(values);
        canvas.drawBitmap(bm, matrix, null);
        iv.setImageBitmap(newBm);
    }
```

通过上面的代码其实我们发现镜面的代码跟缩放其实基本相同的，唯一不同的就是矩阵的参数不同而已。

3.3 倒影

矩阵示例：

{ 1, 0, 0, x 轴不变

0, -1, 0, y 轴变为负数

0, 0, 1 }

y 轴所有的像素点沿负数方向反过来，展现的效果是：倒影

Tips：倒影的代码跟 3.2 章节中的代码一样，唯一不同的就是矩阵参数不同。同时为了看到倒影后的效果需要在矩阵中添加如下代码：

```
matrix.postTranslate(0, bm.getHeight());
```

3.4 旋转

Matrix 中提供了设置图片旋转角度的方法：

setRotate(**float** degrees, **float** px, **float** py)

degrees：要旋转的角度

px ： 旋转原点的 X 轴坐标

py ： 旋转原点的 Y 轴坐标

```
// 旋转
    public void rotate() {
        Bitmap bm = BitmapFactory.decodeResource(getResources(),
R.drawable.logo);
        Bitmap newBm = Bitmap.createBitmap(bm.getWidth(),
bm.getHeight(), Config.ARGB_8888);
        Canvas canvas = new Canvas(newBm);
        Matrix matrix = new Matrix();
```

```
        //获取屏幕的宽度
        int width = getWindowManager().getDefaultDisplay().getWidth();
        //获取屏幕的高度
        int height =
getWindowManager().getDefaultDisplay().getHeight();
        //算出图片的 x 轴中心坐标
        int pointX = (width-bm.getWidth())/2;
        //算出图片 y 轴中心坐标
        int pointY = (height-bm.getHeight())/2;
        //顺时针旋转 30 度
        matrix.setRotate(30, pointX,pointY);
        canvas.drawBitmap(bm, matrix, null);
        iv.setImageBitmap(newBm);
    }
```

3.5 位移

Matrix 中提供了设置图片位移的方法：

setTranslate(**float** dx, **float** dy)

dx　：　位移的 X 轴距离

dy　：　位移的 Y 轴距离

```
// 位移
    public void translate() {
        Bitmap bm = BitmapFactory.decodeResource(getResources(),
R.drawable.logo);
        Bitmap newBm = Bitmap.createBitmap(bm.getWidth(),
bm.getHeight(), Config.ARGB_8888);
        Canvas canvas = new Canvas(newBm);
        Matrix matrix = new Matrix();
        //像 x 轴方向移动 10，y 轴方向移动 40
        matrix.setTranslate(10, 40);
        canvas.drawBitmap(bm, matrix, null);
        iv.setImageBitmap(newBm);
    }
```

4.图片颜色处理（★★）

4.1 颜色过滤器 ColorMatrixColorFilter

Android 提供了颜色过滤器来进行颜色处理。

- ColorMatrixColorFilter：通过使用一个 4*5 的颜色矩阵来创建一个颜色过滤器，改变图片的颜色信息。

- 图形颜色默认矩阵是一个 4x5 的矩阵，数组表现为:

```
{1, 0, 0, 0, 0,     // red     1*R + 0*G + 0*B + 0*A + 0
 0, 1, 0, 0, 0,     // green   0*R + 1*G + 0*B + 0*A + 0
 0, 0, 1, 0, 0,     // blue    0*R + 0*G + 1*B + 0*A + 0
 0, 0, 0, 1, 0}     // alpha 0*R + 0*G + 0*B + 1*A + 0
```

颜色矩阵的每一行的最后一个值更改时，其对应的颜色值就会发生改变,所以更改颜色只需修改其对应颜色矩阵行的最后一项的值即可，最大值范围为 255。

4.2 实现图片美化功能

- 需求：加载一张图片，通过 4 个 SeekBar 分别调整 R（Red）、G（Green）、B（Blue）、A（Alpha）值，第四个同时改变 RGB 值，实现图片颜色的变亮。

- 设置页面布局

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

```
    android:orientation="vertical"
    tools:context=".MainActivity" >

    <ImageView
        android:src="@drawable/v"
        android:id="@+id/iv"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1" />

    <EditText
        android:visibility="invisible"
        android:id="@+id/et"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="b.jpg"
        />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        >
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="R:"
            />
        <SeekBar
        android:id="@+id/sb_red"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:max="255" />
    </LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    >
```

```
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="G:"
        />
    <SeekBar
        android:id="@+id/sb_green"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:max="255" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    >
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="B:"
        />
    <SeekBar
        android:id="@+id/sb_blue"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:max="255" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    >
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="RGB:"
        />
    <SeekBar
        android:id="@+id/sb_rgb"
```

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:max="255" />
</LinearLayout>

</LinearLayout>
```

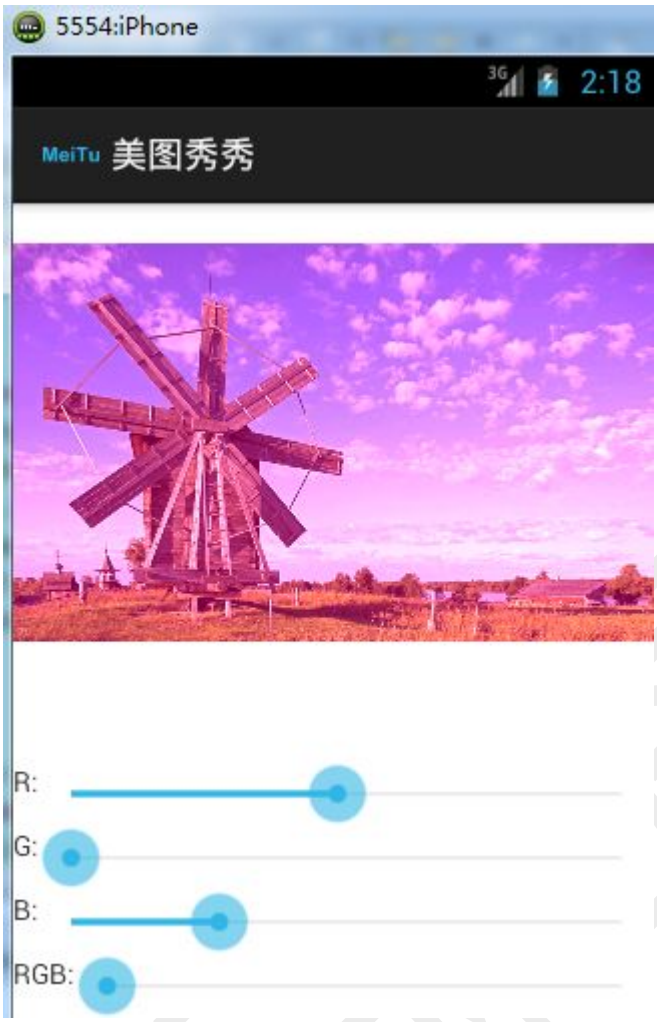
- 在 drawable 目录下放置一张需要处理的图片

- 实现业务逻辑代码

```
public class MainActivity extends Activity implements
OnSeekBarChangeListener {
    //声明页面控件对象
    private ImageView iv;
    private SeekBar sb_red;
    private SeekBar sb_green;
    private SeekBar sb_blue;
    private SeekBar sb_rgb;
    private EditText et;
    //初始化一个矩阵数组
    private float[] arrays = new float[]{
        1.0f,0,0,0,0,
        0,1.0f,0,0,0,
        0,0,1.0f,0,0,
        0,0,0,1,0
    };
    //声明一个颜色过滤器
    private ColorFilter colorFilter = new
ColorMatrixColorFilter(arrays);
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //实例化界面控件
        sb_red = (SeekBar) findViewById(R.id.sb_red);
        sb_green = (SeekBar) findViewById(R.id.sb_green);
        sb_blue = (SeekBar) findViewById(R.id.sb_blue);
        sb_rgb = (SeekBar) findViewById(R.id.sb_rgb);
```

```
        et = (EditText) findViewById(R.id.et);
        iv = (ImageView) findViewById(R.id.iv);
        //给 SeekBar 对象设置监听事件，这是本类实现了
OnSeekBarChangeListener 接口
        sb_blue.setOnSeekBarChangeListener(this);
        sb_green.setOnSeekBarChangeListener(this);
        sb_red.setOnSeekBarChangeListener(this);
        sb_rgb.setOnSeekBarChangeListener(this);
    }
    //覆写 onProgressChanged 方法
    @Override
    public void onProgressChanged(SeekBar seekBar, int progress,
boolean fromUser) {
        int id = seekBar.getId();
        switch (id) {
        case R.id.sb_red:
            arrays[4] = progress;
            break;
        case R.id.sb_blue:
            arrays[14] = progress;
            break;
        case R.id.sb_green:
            arrays[9] = progress;
            break;
        case R.id.sb_rgb:
                arrays[4]=arrays[9]=arrays[14]=progress;
                break;
        default:
            break;
        }
        //修改对应的 RGB 值，穿创建新的 ColorFilter 对象
        colorFilter = new ColorMatrixColorFilter(arrays);
        //给 ImageView 控件设置颜色过滤器
        iv.setColorFilter(colorFilter);
    }
}
```

运行效果如下图：



5.案例-随手涂鸦（★★）

5.1 实现原理

Android 中只有 View 才可以捕获到用户触摸的事件。

ImageView 控件可以设置一个触摸事件的监听器来监听触摸事件，重写 OnTouchListener 的 onTouch 方法,结合 Canvas 类，即可实现随手涂鸦的画板功能

注意：onTouch 方法的返回值默认是 false 的，必须设置为 true,否则触摸事件将不会被处理。

触摸事件的类型分为：

- MotionEvent.ACTION_DOWN　按下
- MotionEvent.ACTION_MOVE　移动
- MotionEvent.ACTION_UP　抬起

5.2 代码实现

- 需求：手指在界面滑动的时候绘制线条。点击保存按钮可以将绘制的图形保证到存储卡上，点击取消按钮可以将当前界面清空。
- 布局文件比较简单，这里不再给出。
- 代码清单：

```
public class MainActivity extends Activity implements OnTouchListener
{
    private ImageView iv;
    private Bitmap bitmap;
    private Canvas canvas;
    //起始坐标
    private int startX;
    private int startY;
    private Paint paint;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //实例化 ImageView 对象
        iv = (ImageView) findViewById(R.id.iv);
        //给 ImageView 设置触摸事件监听
        iv.setOnTouchListener(this);
    }
    //覆写 onTouch 方法
    @Override
```

```
    public boolean onTouch(View view, MotionEvent motionEvent) {
        //判断动作类型
        switch (motionEvent.getAction()) {
        //手指按下事件
        case MotionEvent.ACTION_DOWN:
            //如果当前 bitmap 为空
            if(bitmap==null){
                //创建一个新的 bitmap 对象，宽、高使用界面布局中 ImageView
对象的宽、高
                bitmap = Bitmap.createBitmap(iv.getWidth(),
iv.getHeight(), Config.ARGB_8888);
                //根据 bitmap 对象创建一个画布
                canvas = new Canvas(bitmap);
                //设置画布背景色为白色
                canvas.drawColor(Color.WHITE);
                //创建一个画笔对象
                paint = new Paint();
                //设置画笔的颜色为红色，线条粗细为 5 磅
                paint.setColor(Color.RED);
                paint.setStrokeWidth(5);
            }
            //记录手指按下时的屏幕坐标
            startX = (int)motionEvent.getX();
            startY = (int)motionEvent.getY();
            break;
        case MotionEvent.ACTION_MOVE://手指滑动事件
            //记录移动到的位置坐标
            int moveX = (int) motionEvent.getX();
            int moveY = (int) motionEvent.getY();
            //绘制线条，连接起始位置和当前位置
            canvas.drawLine(startX, startY, moveX, moveY, paint);
            //在 ImageView 中显示 bitmap
            iv.setImageBitmap(bitmap);
            //将起始位置改变为当前移动到的位置
            startX = moveX;
            startY = moveY;
            break;
        default:
            break;
        }
        return true;
```

```
    }
    //清除界面
    public void clear(View view){
        bitmap = null;
        iv.setImageBitmap(null);
    }
    //将当前绘制的图形保存到文件
    public void save(View view){
        if (bitmap==null) {
            Toast.makeText(this, "没有图片可以保存",
Toast.LENGTH_SHORT).show();
            return ;
        }
        //创建一个文件对象
        File file = new File(getFilesDir(),"mypic"+new
Date().getDate()+"_"+new Date().getHours()+"_"+new
Date().getMinutes()+"_"+new Date().getSeconds()+".jpg");
        FileOutputStream stream = null;
        try {
            stream = new FileOutputStream(file);
            //以 JPEG 的图形格式将当前图片以流的形式输出
            boolean compress = bitmap.compress(CompressFormat.JPEG,
100, stream);
            if (compress) {
                Toast.makeText(this, "保存成功:
"+file.getAbsolutePath(), Toast.LENGTH_SHORT).show();
            }else {
                Toast.makeText(this, "保存失败",
Toast.LENGTH_SHORT).show();
            }
        } catch (FileNotFoundException e) {
            Toast.makeText(this, "保存失败"+e.getLocalizedMessage(),
Toast.LENGTH_SHORT).show();
        }finally{
            if (stream!=null) {
                try {
                    stream.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
```

```
        }
    }
}
```

运行改项目，效果如下图：

6.案例-撕衣服游戏（★★）

6.1 实现原理

使用帧布局，准备 2 张图片，一张图片有衣服，一张图片没有衣服。没有衣服的图片放置在下面，有衣服的图片放置在上面，为在上面的 ImageView 设置触摸的事件，当手指

触摸到图片上时，将手指触摸的点周边的上层图片的像素点设置为透明的，就会出现一个撕衣服的效果。

Tips：

1. 触摸事件 onTouch 的返回值必须设置为 true，否则触摸的事件将不被处理
2. 使用 BitmapFactory 的 decodeResouces 方法得到的图片是没有透明度的，即图片格式为 RGB_565,所以若想能够修改透明度，需要使用 Canvas 对象对图片进行重绘，重新绘制的图片格式采用 ARGB。
3. 加载图片时需要对其进行一下压缩，防止图片与控件大小不匹配，导致触摸时点对不上，达不到触摸那里就设置哪里的像素点透明的效果。

6.2 代码实现

```
public class MainActivity extends Activity implements OnTouchListener
{

    private Bitmap bitmap;
    private ImageView ivTop;
    private ImageView ivBottom;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //实例化两个 ImageView 控件对象
        ivBottom = (ImageView) findViewById(R.id.iv_bottom);
        ivTop = (ImageView) findViewById(R.id.iv_top);


        Options opts = new Options();
        opts.inSampleSize = 2;
        //加载 2 张图片
```

```
        Bitmap topBitmap =
BitmapFactory.decodeResource(getResources(), R.drawable.a8, opts);
        Bitmap bottomBitmap =
BitmapFactory.decodeResource(getResources(), R.drawable.b8, opts);
        //创建一个 bitmap 对象
        bitmap = Bitmap.createBitmap(topBitmap.getWidth(),
topBitmap.getHeight(), Config.ARGB_8888);
       //创建一个画布对象
        Canvas canvas = new Canvas(bitmap);
        //将顶层图片绘制到 bitmap 对象中
        canvas.drawBitmap(topBitmap, 0, 0, null);
        //给 ImageView 控件设置图片
        ivBottom.setImageBitmap(bottomBitmap);
        //给顶层 ImageView 设置新绘制的位图
        ivTop.setImageBitmap(bitmap);
        //给顶层图片控件设置监听事件
        ivTop.setOnTouchListener(this);
    }
    /**
     * 覆写 onTouch 方法
     */
   public boolean onTouch(View v, MotionEvent event) {
      if(event.getAction() == MotionEvent.ACTION_MOVE) {
         //获取当前屏幕坐标
         int x = (int) event.getX();
         int y = (int) event.getY();
         //将当前坐标周围的一个矩形区域设置为透明
          for(int i = x - 10; i < x + 10; i ++) {
            for (int j = y - 10; j < y + 10; j++) {
               if(i >= 0 && i < bitmap.getWidth()
                     && j >= 0 && j < bitmap.getHeight()) {
                  bitmap.setPixel(i, j, Color.TRANSPARENT);
               }
            }
          }
          //重新给 ivTop 控件设置 bitmap 对象
          ivTop.setImageBitmap(bitmap);
      }
      //必须返回 true，否则事件不成功
      return true;
   }
```

运行效果如下图所示：

至此，本文档完！

2014-12-21 14.12.21

北京市海淀区东馨园小区