



作者 D\_clock爱吃葱花 (/users/ec95b5891948) 2016.09.04 21:17\*

写了43444字，被2320人关注，获得了2491个喜欢

(/users/ec95b5891948)

+ 添加关注 (/sign\_in)

# ScratchView：一步步打造万能的 Android 刮奖效果控件

字数2350 阅读1778 评论13 喜欢68

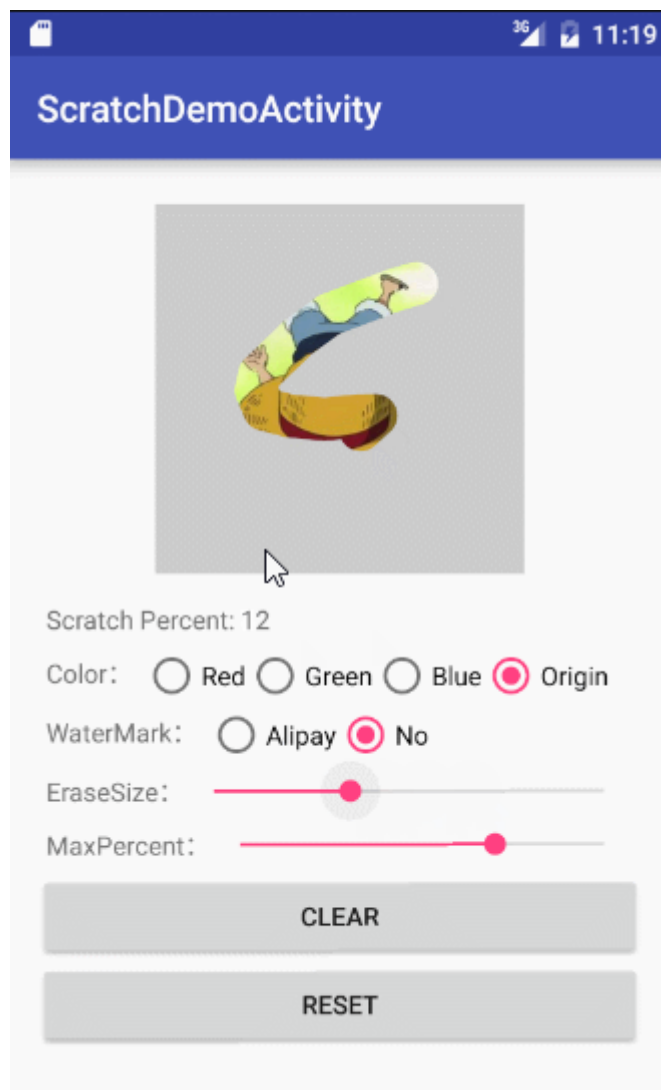
Hello，大家好，我是Clock。这周为大家带来一篇关于自定义控件的文章，这也是我本人第一次写关于自定义控件的文章，希望可以写得言简意赅，通俗易懂点。

## 前言

我身边有一部分开发的小伙伴，存在着这样一种习惯。某一天，突然看到某一款 App 上有个很漂亮的自定义控件（动画）效果，就会绞尽脑子想办法去自己实现一发。当然，我自己也是属于这类型的骚年，看到某种效果就会手痒难耐琢磨着实现套路。个人觉得这是一种需求驱动进步的方法，当你绞尽脑子去实现自己想要的效果时，你就会发现你对 Android 自定义控件（动画）的知识体系认识越深，久而久之，自己也能轻松的造出各种控件（动画）效果。要是哪天，产品童鞋拿着个原型（或者对着某款 App）跟你讲：“XXXX，你看这个效果我们能不能实现？”，然后你瞥了一眼，胸有成竹丢回一句：“开玩笑，还有我实现不了的效果？”。想想心里是不是有点小激动？好了，差不多要说回正题了，这是我第一篇关于自定义控件的文章，以后也会陆续穿插更新此类型的文章，希望大家能够喜欢。（偷偷剧透下，我下篇文章是关于性能优化的干货。当然我自己觉得很干货，希望到时候发出来不要打脸，哈哈哈！）

## 实现效果

说了这么多，还是先给大家看看最终的实现效果先



上面只是基本实现效果的一部分，你会看到下方还有很多其他控件，它们是用来干嘛的，接下来即将为你揭晓一切。

## 基本实现

日常生活中，我们对刮奖效果想必不会陌生，其原理就是通过在原有图案和文字上添加刮层来实现的。如果我们想看到刮层后面藏的图案和文字是什么，势必要通过刮开刮层才行。知道了这样的套路，就可以开始整理一下编码实现思路，然后愉快开干。

我一开始的实现思路是想通过重写 `ImageView` 和 `TextView`，然后在分别用代码在图像和文字上添加图层，这样的话就能实现出效果了。然而回头一想，不对，这种实现存在的局限性比较大。如果照这种思路实现，那么刮层下面只能存在图片或者文字，如果产品经理要求同时存在图片和文字呢？要求存在两张图片呢？要求同时存在图片和文字，且文字放在图片的上（下、左、右）呢？...**我们都知道，世界上最善变的除了妹纸的心，就是产品经理和他们的需求了。**于是，便想出另外一种实现思路，**直接继承 `View` 来实现一个刮层**，让这个刮层和图片以及文字不产生任何依赖，**再结合 `FrameLayout` 将刮层放置最上一层**，刮层之下你想放多少图片文字，图片文字要怎么布局摆放都行。到此，思路明确，可以愉快的开始编码了。

**第一步：绘制出刮层效果。**

```

package com.clock.scratch;

import ...;

/**
 * Created by Clock on 2016/8/26.
 */
public class ScratchView extends View {

    ...

    public ScratchView(Context context) {
        super(context);
        TypedArray typedArray = context.obtainStyledAttributes(R.styleable.ScratchView);
        init(typedArray);
    }

    ...

    private void init(TypedArray typedArray) {
        ...
        mMaskColor = typedArray.getColor(R.styleable.ScratchView_maskColor, DEFAULT_MASKER_COLOR);

        mMaskPaint = new Paint();
        mMaskPaint.setAntiAlias(true); //抗锯齿
        mMaskPaint.setDither(true); //防抖
        setMaskColor(mMaskColor);
        ...
    }

    /**
     * 设置蒙板颜色
     *
     * @param color 十六进制颜色值, 如: 0xffff0000 (不透明的红色)
     */
    public void setMaskColor(int color) {
        mMaskPaint.setColor(color);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        canvas.drawBitmap(mMaskBitmap, 0, 0, mBitmapPaint); //绘制图层遮罩
    }

    @Override
    protected void onSizeChanged(int w, int h, int oldw, int oldh) {
        super.onSizeChanged(w, h, oldw, oldh);
        createMasker(w, h);
    }

    /**

```

```

    * 创建蒙层
    *
    * @param width
    * @param height
    */
    private void createMasker(int width, int height) {
        mMaskBitmap = Bitmap.createBitmap(width, height, Bitmap.Config.ARGB_8888);
        mMaskCanvas = new Canvas(mMaskBitmap);
        Rect rect = new Rect(0, 0, width, height);
        mMaskCanvas.drawRect(rect, mMaskPaint); //绘制生成和控件大小一致的遮罩 Bitmap
    }
}

```

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="ScratchView">
        <!-- 蒙层的颜色 -->
        <attr name="maskColor" format="color|reference" />
    </declare-styleable>
</resources>

```

上面的代码思路如下：

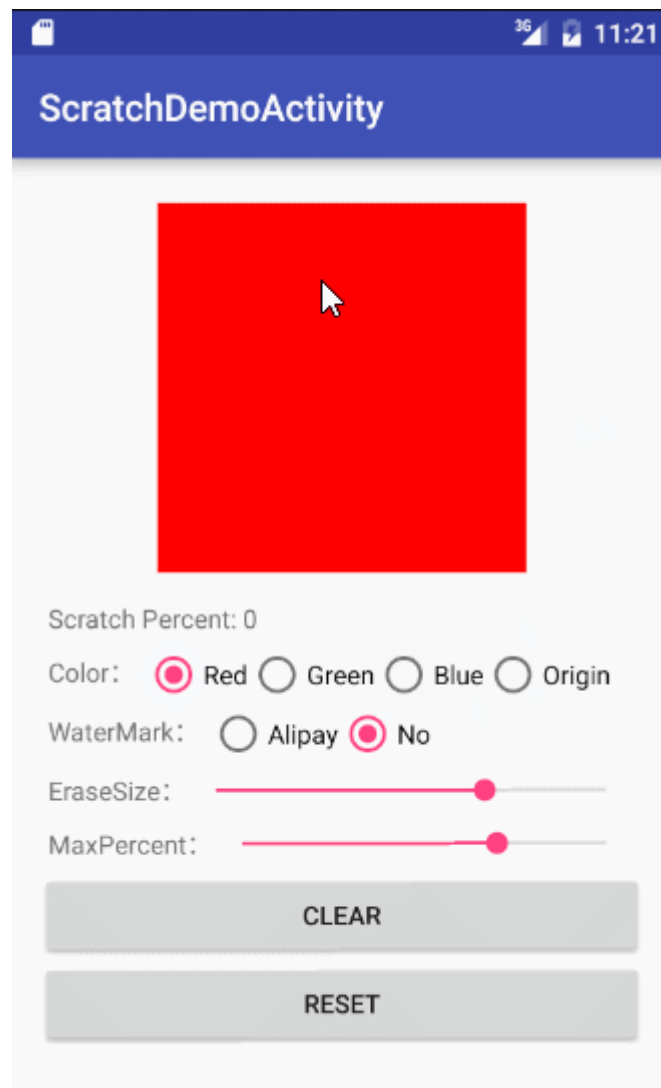
- 创建出继承于 View 的自定义控件 ScratchView，同时在 init() 函数中初始化各类参数设置。如刮层的颜色等等；
- 为了方便设置，需要把参数抽离成控件的自定义属性，同时 ScratchView 类中提供 set 方法，供代码调用。如刮层的颜色属性就是 maskColor，其在类中对应的方法就是 setMaskColor；
- 在 onSizeChanged 中，利用 View 已经 Measure 完毕，可以获得 View 的宽高，并使用 Canvas 来初始化生成 mMaskBitmap 用于制作刮层；
- 在 onDraw 中，利用 canvas.drawBitmap 将 onSizeChanged 中初始化生成 mMaskBitmap 绘制显示到界面，生成刮层；

在 Demo 中添加如下布局，看下效果：

```

<FrameLayout
    android:layout_width="200dp"
    android:layout_height="200dp"
    android:layout_gravity="center_horizontal"
    android:layout_marginTop="8dp">
    <!--刮层下遮住的内容-->
    <ImageView
        android:layout_width="150dp"
        android:layout_height="150dp"
        android:layout_gravity="center"
        android:src="@mipmap/lufy" />
    <!--刮层-->
    <com.clock.scratch.ScratchView
        android:id="@+id/scratch_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</FrameLayout>

```



到此，我们已经获得了一个刮层的实现效果，同时可以直接在 xml 布局和 java 代码中设置刮层的颜色了。但是这时候，只是空有刮层，并没有实现刮开的效果，接下来继续添加实现代码。

## 第二步：实现刮开效果。

```

package com.clock.scratch;

import ...;

public class ScratchView extends View {

    public ScratchView(Context context) {
        super(context);
        TypedArray typedArray = context.obtainStyledAttributes(R.styleable.ScratchView);
        init(typedArray);
    }

    private void init(TypedArray typedArray) {
        mEraseSize = typedArray.getFloat(R.styleable.ScratchView_eraseSize, DEFAULT_ERASER_SIZE);
        ...

        mErasePaint = new Paint();
        mErasePaint.setAntiAlias(true);
        mErasePaint.setDither(true);
        mErasePaint.setXfermode(new PorterDuffXfermode(PorterDuff.Mode.CLEAR)); // 设置擦除效果
        mErasePaint.setStyle(Paint.Style.STROKE);
        mErasePaint.setStrokeCap(Paint.Cap.ROUND); // 设置笔尖形状，让绘制的边缘圆滑
        setEraserSize(mEraseSize);

        mErasePath = new Path();

        ViewConfiguration viewConfiguration = ViewConfiguration.get(getContext());
        mTouchSlop = viewConfiguration.getScaledTouchSlop();
    }

    /**
     * 设置橡皮擦尺寸大小（默认大小是 60）
     *
     * @param eraserSize 橡皮擦尺寸大小
     */
    public void setEraserSize(float eraserSize) {
        mErasePaint.setStrokeWidth(eraserSize);
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        int action = event.getAction();
        switch (action) {
            case MotionEvent.ACTION_DOWN:
                startErase(event.getX(), event.getY());
                invalidate();
                return true;
            case MotionEvent.ACTION_MOVE:
                erase(event.getX(), event.getY());
                invalidate();
                return true;
            case MotionEvent.ACTION_UP:

```

```

        stopErase();
        invalidate();
        return true;
    default:
        break;
    }
    return super.onTouchEvent(event);
}

/**
 * 开始擦除
 *
 * @param x
 * @param y
 */
private void startErase(float x, float y) {
    mErasePath.reset();
    mErasePath.moveTo(x, y);
    this.mStartX = x;
    this.mStartY = y;
}

/**
 * 擦除
 *
 * @param x
 * @param y
 */
private void erase(float x, float y) {
    int dx = (int) Math.abs(x - mStartX);
    int dy = (int) Math.abs(y - mStartY);
    if (dx >= mTouchSlop || dy >= mTouchSlop) {
        this.mStartX = x;
        this.mStartY = y;

        mErasePath.lineTo(x, y);
        mMaskCanvas.drawPath(mErasePath, mErasePaint);

        mErasePath.reset();
        mErasePath.moveTo(mStartX, mStartY);
    }
}

/**
 * 停止擦除
 */
private void stopErase() {
    this.mStartX = 0;
    this.mStartY = 0;
    mErasePath.reset();
}
}

```

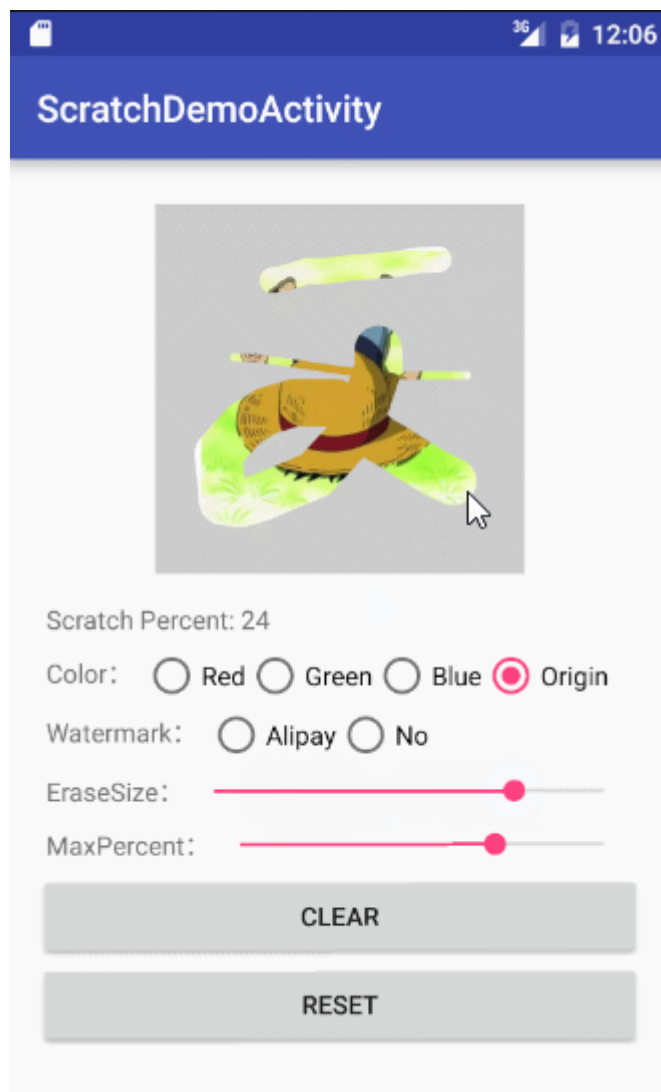
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="ScratchView">
        <!--擦除尺寸大小-->
        <attr name="eraseSize" format="float" />
    </declare-styleable>
</resources>
```

上面的代码思路如下：

- 在 init() 中初始化 mErasePaint 和 mErasePath，并设置 mErasePaint 的 Xfermode 为 PorterDuff.Mode.CLEAR 用于后面制造出刮奖效果；
- 重写 onTouchEvent 函数，处理触摸事件 ACTION\_DOWN、ACTION\_MOVE、ACTION\_UP 等三种事件类型，并利用 mErasePath 记录手指滑动轨迹，再用 mMaskCanvas 将滑动轨迹绘制到第一步生成的 mMaskBitmap 上，最后通过调用 invalidate() 引起 View 的重绘生成刮开效果；
- 为了防止滑动过于灵敏，我们需要对滑动做一个判断就是通过系统提供的 viewConfiguration.getScaledTouchSlop() 获取系统认为的最小滑动距离，当等于或者超过这个距离时，才认为是在滑动，这就是为什么我在 erase() 要加 dx >= mTouchSlop || dy >= mTouchSlop 的判断；
- 为了控制刮痕的粗细，和前面设置刮层的颜色一样，同样为 ScratchView 自定义一个属性 eraseSize 实现在 xml 中控制。同时，在 Java 代码中提供调用方法；

到此，一个基本的刮奖效果已经完成了，我们来看看实现效果如何。





以上两步仅仅完成基础效果而已了，接下来我们来做一些优化。

## 效果优化

### 第一步优化：添加水印

很多刮奖的效果都会有在刮层上添加自家 logo 做水印效果（这里不知道称为水印合适吗？反正就是大概那个意思）。如下面的支付宝一样



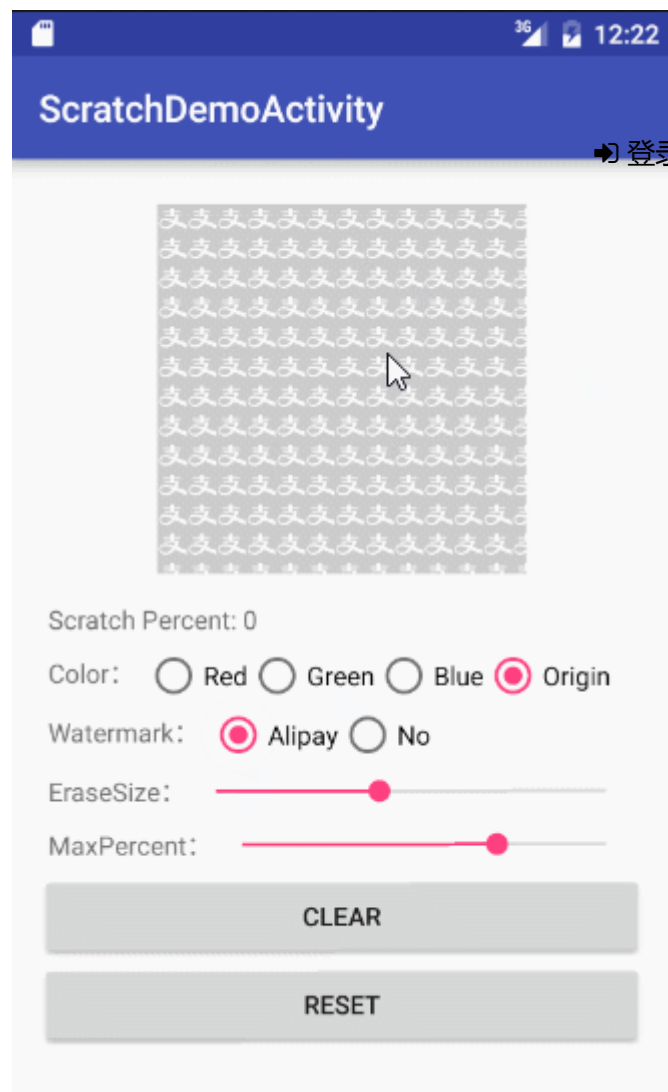
我们在基础实现的第一步中的创建刮层函数里面添加实现代码，同时也添加一个自定义属性和 set 方法可供调用：

```
/**
 * 设置水印图标
 *
 * @param resId 图标资源id, -1表示去除水印
 */
public void setWatermark(int resId) {
    if (resId == -1) {
        mWatermark = null;
    } else {
        Bitmap bitmap = BitmapFactory.decodeResource(getResources(), resId);
        mWatermark = new BitmapDrawable(bitmap);
        mWatermark.setTileModeXY(Shader.TileMode.REPEAT, Shader.TileMode.REPEAT);
    }
}

/**
 * 创建蒙层
 *
 * @param width
 * @param height
 */
private void createMasker(int width, int height) {
    ...

    if (mWatermark != null) {
        Rect bounds = new Rect(rect);
        mWatermark.setBounds(bounds);
        mWatermark.draw(mMaskCanvas);
    }
}
```

实现效果如下：



当然，像效果上还有很多可以进行添加，例如还可以加上面支付宝的那种边缘锯齿效果等等，这里就各位童鞋自行脑洞实现啦。

**第二步优化：添加相应事件监听器，以及完善一些常用函数。**

说到事件监听，我想这里莫过于刮奖完成的事件了吧。对于使用这个控件的开发者，肯定需要在刮完之后做相应的操作，例如，提示用户中奖啦，还是继续努力之类的。怎么样判断刮奖完成呢？这里的实现思路是通过异步计算刮层 mMaskBitmap 中的像素信息值，通过算得透明像素个数占总像素个数的比例，当这个比例超过一定阈值的时候，我们认为刮奖完成了。为什么要说超过一定阈值就算完成，这和现实生活中刮奖一样，你不需要把刮层完全刮得干干净净才能得到结果。当然这个比例是多少，我们同样需要抽离成可动态设置的。再添加监听器接口和设置监听器的 API 即可。实现代码，大致如下：

```
private void onErase() {
    int width = getWidth();
    int height = getHeight();
    new AsyncTask<Integer, Integer, Boolean>() {

        @Override
        protected Boolean doInBackground(Integer... params) {
            int width = params[0];
            int height = params[1];
            int pixels[] = new int[width * height];
            mMaskBitmap.getPixels(pixels, 0, width, 0, 0, width, height); //获取覆盖图层中所有的像素信息, stri

            float erasePixelCount = 0; //擦除的像素个数
            float totalPixelCount = width * height; //总像素个数

            for (int pos = 0; pos < totalPixelCount; pos++) {
                if (pixels[pos] == 0) { //透明的像素值为0
                    erasePixelCount++;
                }
            }

            int percent = 0;
            if (erasePixelCount >= 0 && totalPixelCount > 0) {
                percent = Math.round(erasePixelCount * 100 / totalPixelCount);
                publishProgress(percent);
            }

            return percent >= mMaxPercent;
        }

        @Override
        protected void onProgressUpdate(Integer... values) {
            super.onProgressUpdate(values);
            mPercent = values[0];
            onPercentUpdate();
        }

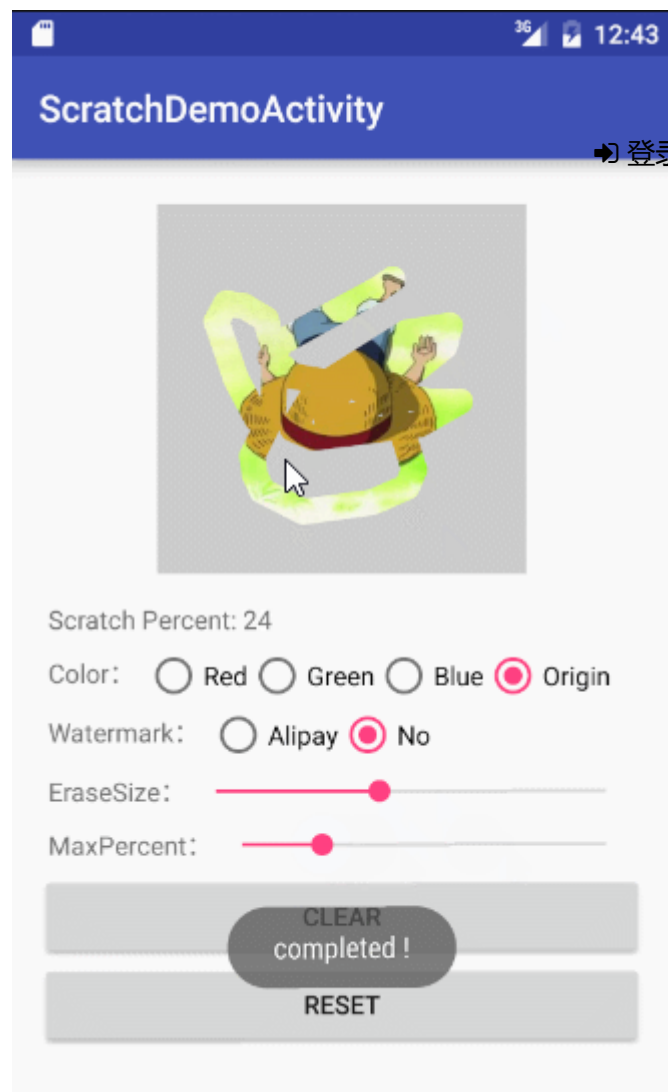
        @Override
        protected void onPostExecute(Boolean result) {
            super.onPostExecute(result);
            if (result && !mIsCompleted) { //标记擦除, 并完成回调
                mIsCompleted = true;
                if (mEraseStatusListener != null) {
                    mEraseStatusListener.onCompleted(ScratchView.this);
                }
            }
        }

    }.execute(width, height);
}

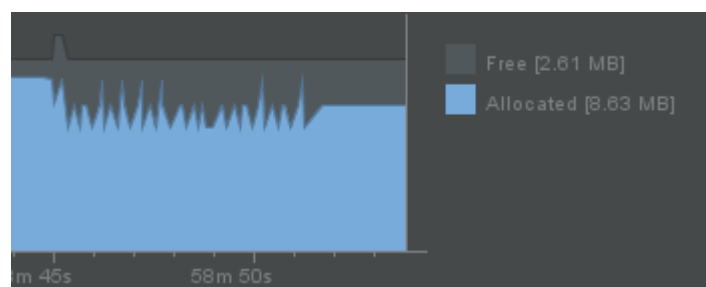
/**
 * 设置擦除监听器
```

[illegible]

## 我们来看看最终效果



到这里，一个完整的刮奖效果自定义控件实现已经完成。不过，这里还有一个问题需要抛给大家共同思考下，就是在判断刮奖是否完成的实现上，我在代码中的实现方式会创建出大量的 int 数组，这样造成后果就是会产生内存抖动。



目前，因为我本人也没想到什么好方案，所以，大家如果有好的思路，不妨在下方留言赐教一下。

## 总结

第一次写自定义控件这类型的文章，不知道大家看明白实现思路了吗？关于自定义控件，单看文章只能懂其中的思路，结合源代码边动手实践调试再加上文章会更深有体会。需要源代码的童鞋可以到 <https://github.com/D-clock/ScratchView> (<https://github.com/D-clock/ScratchView>) 中下载，接下来还有很多关于自定义控件（动画）的文章想写，敬请期待。

欢迎大家关注我的简书 ([http://www.jianshu.com/users/ec95b5891948/latest\\_articles](http://www.jianshu.com/users/ec95b5891948/latest_articles))和Github (<https://github.com/D-clock>)

感谢赞赏，么么哒！

¥ 打赏支持

  
(/users/4a4e14f0236a2887c)

♡ 喜欢 | 68


🐦 分享到微博

👤 分享到微信

更多分享 ▼


13条评论 ( 按时间正序 · 按时间倒序 · 按喜欢排序 )

✎ 添加新评论 (/sign\_in)

 英勇青铜5 (/users/8edc8ef5fef3)  
2 楼 · 2016-09-04 21:54 (/p/2514a08d8217/comments/4016598#comment-4016598)  
学习大神


♡ 喜欢(0)

回复

 Mr孙 (/users/9d03e58e7628)  
3 楼 · 2016-09-05 08:13 (/p/2514a08d8217/comments/4020623#comment-4020623)  
已收藏


♡ 喜欢(0)

回复

 弋茹\_ (/users/67d75c11e8a3)  
4 楼 · 2016-09-05 10:35 (/p/2514a08d8217/comments/4022500#comment-4022500)  
天了噜竟然看到了原作。之前在微信公众号看到了转载~

♡ 喜欢(0)

回复

 doyee (/users/31ad56ed957f)  
5 楼 · 2016-09-05 17:06 (/p/2514a08d8217/comments/4023261#comment-4023261)  
好绚的效果 果断收藏 学习


↑

📄

⋮

♡ 喜欢(0)


回复

 简控 (/users/5b8db0f1c979)  
6楼 - 2016.09.05 17:52 (/p/2514a08d8217/comments/4025853#comment-4025853)  
66666

登录 (/sign\_in) 注册 (/sign\_up)


♡ 喜欢(0)

回复

 wo叫天然呆 (/users/b55a43d1711d)  
7楼 - 2016.09.05 17:52 (/p/2514a08d8217/comments/4034683#comment-4034683)  
一直关注着，受益良多


♡ 喜欢(0)

回复

 月骑三四 (/users/ac7b73ba7d43)  
8楼 - 2016.09.06 07:49 (/p/2514a08d8217/comments/4037573#comment-4037573)  
找到了启发

♡ 喜欢(0)

回复

 CrazyLeaf (/users/ece3fc82ad9f)  
9楼 - 2016.09.07 15:16 (/p/2514a08d8217/comments/4061672#comment-4061672)  
我用onErase()方法的时候,里面的AsyncTask有一些Bug,重置的时候,当前的mPrecent为最大了. 然后一直为0.是不是在reset() 的时候,关闭AsyncTask,要不然好多进程.都混乱了.


♡ 喜欢(0)

回复

D\_clock爱吃葱花 (/users/ec95b5891948) : @Laputa\_Zeej (/users/ece3fc82ad9f) 可以上Github详细描述一下BUG，我找时间看看，优化下！  
2016.09.07 19:57 (/p/2514a08d8217/comments/4066253#comment-4066253)


回复

添加新回复

 LLS不想挂机了 (/users/72baa1b75a82)  
10楼 - 2016.09.09 10:38 (/p/2514a08d8217/comments/4093375#comment-4093375)  
思路非常棒啊~~~  
顺便问一下，用GIF录制工具是什么，好清晰。

♡ 喜欢(0)

回复


 Zack\_zhou (/users/3974ab7605b8)  
11楼 - 2016.09.11 15:05 (/p/2514a08d8217/comments/4131150#comment-4131150)

^







 callmeharry (/users/cc3304937d43)  
(/users/cc3304937d43) 12楼 2016.09.15 23:19 (/p/2514a08d8217/comments/4197039#comment-4197039)


擦除百分比那里的pixel数组可以作为view的成员变量，erase的时候用坐标和橡皮擦半径的像素值计算被擦除区域每一行像素的边界来更新pixel，同时更新一个被擦除的像素数量，应该能避免内存抖动。

D\_clock爱吃葱花 (/users/ec95b5891948) : @callmeharry (/users/cc3304937d43) 这个问题已经优化啦！ 😊  
2016.09.15 23:15 (/p/2514a08d8217/comments/4210630#comment-4210630)

✎ 添加新回复

登录后发表评论 (/sign\_in)

被以下专题收入，发现更多相似内容：




**程序员** (/collection/NEt52a)

如果你是程序员，或者有一颗喜欢写程序的心，喜欢分享技术干货、项目经验、程序员日常囧事等等，欢迎投稿《程序员》专题。 专题主编：小...

23707篇文章 (/collection/NEt52a) · 173079人关注

✎ 添加关注 (/sign\_in)




**Android知识** (/collection/3fde3b545a35)

分享Android开发的知识，教程，解析，前沿信息，都可以，欢迎大家投稿~ 内容可搞笑，可逗比，另外欢迎申请管理员

5010篇文章 (/collection/3fde3b545a35) · 20989人关注

✎ 添加关注 (/sign\_in)



**今日看点** (/collection/3sT4qY)

本专题仅让编辑在特殊情况下使用，入选文章将立刻上首页。绕过推荐队列。 使用场景： - 突发事件的相关内容，需要立刻推首页 -...

38758篇文章 (/collection/3sT4qY) · 15869人关注

✎ 添加关注 (/sign\_in)