Android系统中Bitmap是否有调用recycle搜索方法的必要性

Bitmap需调用 recycle() 是历史问题，在 Android [3.0](https://www.baidu.com/s?wd=3.0&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLuWT3rHfkujuWPADzmyDv0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHbzPHn1PH0kn1bvrjmznHDsPs)之前，Bitmap 的图片数据是在底层C中处理的，因此在 Android[3.0](https://www.baidu.com/s?wd=3.0&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLuWT3rHfkujuWPADzmyDv0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHbzPHn1PH0kn1bvrjmznHDsPs) 之前 recycle() 是应该调用的。虽然 finalize() 会调用 recycle() ,但对Java有经验的同学应该知道只依靠 finalize() 去释放资源是会出很多问题的.  
在Android [3.0](https://www.baidu.com/s?wd=3.0&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLuWT3rHfkujuWPADzmyDv0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHbzPHn1PH0kn1bvrjmznHDsPs)之后，图片数据放在了Bitmap对象的一个成员变量 mBuffer[] 中。因此可以不调用recycle() .在 Bitmap 置 null 后图片数据会被GC回收。  
现在都 [Android5.0](https://www.baidu.com/s?wd=Android5.0&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLuWT3rHfkujuWPADzmyDv0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHbzPHn1PH0kn1bvrjmznHDsPs) 的年代了，建议不考虑支持3.0之前的版本。  
实际上Bitmap.recycle()的说明上也有说明：  
This is an advanced call, and normally need not be called, since the normal GC process will free up this memory when there are no more references to this bitmap.  
(这是一个高级函数，一般来说没必要调用。在没有引用指向 Bitmap 时，GC 会自动释放内存)

以下情况如果recycle会发生错误：

当你根据id从drawable（drawable资源文件夹）中获取一个drawable时，系统会将这个drawable加入缓存之中。这样，你第二次继续获取这个drawable时，如果缓存之中的drawable没有被回收，则会被返回。  
如果你通过getDrawable(id)方法获取到一个bitmap1，继续通过getDrawable(id)方法获取到一个bitmap2。那么bitmap1=bitmap2。所以，当你对bitmap1进行recycle之后，又将bitmap2设置给Imageview显示，极大可能会出现java.lang.RuntimeException: Canvas: trying to use a recycled bitmap android.graphics.Bitmap@xxx的错误。

Android中BitmapFactory.Options详解

原创 2015年08月01日 14:33:16

标签：

[android](http://so.csdn.net/so/search/s.do?q=android&t=blog) /

[Bitmap](http://so.csdn.net/so/search/s.do?q=Bitmap&t=blog) /

[BitmapFactory](http://so.csdn.net/so/search/s.do?q=BitmapFactory&t=blog) /

[Options](http://so.csdn.net/so/search/s.do?q=Options&t=blog)

10190

在Android中，BitmapFactory相信大家都很熟悉了，这个类里面的所有方法都是用来解码创建一个Bitmap，其中有一个重要的类是Options，此类用于解码Bitmap时的各种参数控制，那这里面的各种参数都是什么含义呢？今天我们就来重点解读几个参数。

inJustDecodeBounds：

如果将这个值置为true，那么在解码的时候将不会返回bitmap，只会返回这个bitmap的尺寸。这个属性的目的是，如果你只想知道一个bitmap的尺寸，但又不想将其加载到内存时。这是一个非常有用的属性。

inSampleSize：

这个值是一个int，当它小于1的时候，将会被当做1处理，如果大于1，那么就会按照比例（1 / inSampleSize）缩小bitmap的宽和高、降低分辨率，大于1时这个值将会被处置为2的倍数。例如，width=100，height=100，inSampleSize=2，那么就会将bitmap处理为，width=50，height=50，宽高降为1 / 2，像素数降为1 / 4。

inPreferredConfig：

这个值是设置色彩模式，默认值是ARGB\_8888，在这个模式下，一个像素点占用4bytes空间，一般对透明度不做要求的话，一般采用RGB\_565模式，这个模式下一个像素点占用2bytes。

inPremultiplied：

这个值和透明度通道有关，默认值是true，如果设置为true，则返回的bitmap的颜色通道上会预先附加上透明度通道。

inDither：

这个值和抖动解码有关，默认值为false，表示不采用抖动解码。如果想知道什么是抖动解码，请参看我另一篇文章：<http://blog.csdn.net/haozipi/article/details/47185535>

inDensity：

表示这个bitmap的像素密度（对应的是DisplayMetrics中的densityDpi，不是density）。

inTargetDensity：

表示要被画出来时的目标像素密度（对应的是DisplayMetrics中的densityDpi，不是density）。

inScreenDensity：

表示实际设备的像素密度（对应的是DisplayMetrics中的densityDpi，不是density）。

inScaled：

设置这个Bitmap是否可以被缩放，默认值是true，表示可以被缩放。

ps：inDensity，inTargetDensity，inScreenDensity三个值的具体关系请参看我的另一篇文章：<http://blog.csdn.net/haozipi/article/details/47185917>

inPurgeable和inInputShareable：

这两个值一般是一起使用，设置为true时，前者表示空间不够是否可以被释放，后者表示是否可以共享引用。这两个值在Android5.0后被弃用。

inPreferQualityOverSpeed：

这个值表示是否在解码时图片有更高的品质，仅用于JPEG格式。如果设置为true，则图片会有更高的品质，但是会解码速度会很慢。

outWidth和outHeight：

表示这个Bitmap的宽和高，一般和inJustDecodeBounds一起使用来获得Bitmap的宽高，但是不加载到内存。