traveView :

<https://blog.csdn.net/u011240877/article/details/54347396>

https://blog.csdn.net/itfootball/article/details/48792435

MAT : hrpof文件转换命令：hprof-conv xxx.xxx.xxx.hprof xxx.xxx.xxx.hprof

使用说明：<https://android-developers.googleblog.com/2011/03/memory-analysis-for-android.html>

MAT 查询语句 : <https://blog.csdn.net/yxz329130952/article/details/50288145>

## 图片Jpg 与png 的区别:

https://blog.csdn.net/theone10211024/article/details/50801099

1、jpg是一种有损压缩的图片存储格式，而png则是无损的图片存储格式。

2、jpg的图片没有alpha通道，所以读到的内存如果用的是RGB565的格式内存相对于AGB8888小一半。

JPG 不适用于所含颜色很少、具有大块颜色相近的区域或亮度差异十分明显的较简单的图片。对于需要高保真的较复杂的图像，PNG 虽然能无损压缩，但图片文件较大。

二者的差别主要体现在：

alpha 你是否真的需要？如果需要 alpha 通道，那么没有别的选择，用 png。

你的图色值丰富还是单调？就像刚才提到的，如果色值丰富，那么用jpg，如果作为按钮的背景，请用 png。

对安装包大小的要求是否非常严格？如果你的 app 资源很少，安装包大小问题不是很凸显，看情况选择 jpg 或者 png（不过，我想现在对资源文件没有苛求的应用会很少吧。。）

目标用户的 cpu 是否强劲？jpg 的图像压缩算法比 png 耗时。这方面还是要酌情选择，前几年做了一段时间 Cocos2dx，由于资源非常多，项目组要求统一使用 png，可能就是出于这方面的考虑。

图片压缩

BitmapFactory 在解码图片时，可以带一个Options，有一些比较有用的功能，比如：

inTargetDensity 表示要被画出来时的目标像素密度

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

inJustDecodeBounds 字面意思就可以理解就是只解析图片的边界，有时如果只是为了获取图片的大小就可以用这个，而不必直接加载整张图片。

inPreferredConfig 默认会使用ARGB\_8888,在这个模式下一个像素点将会占用4个byte,而对一些没有透明度要求或者图片质量要求不高的图片，可以使用RGB\_565，一个像素只会占用2个byte，一下可以省下50%内存。

inPurgeable和inInputShareable 这两个需要一起使用，BitmapFactory.java的源码里面有注释，大致意思是表示在系统内存不足时是否可以回收这个bitmap，有点类似软引用，但是实际在5.0以后这两个属性已经被忽略，因为系统认为回收后再解码实际会反而可能导致性能问题

inBitmap 官方推荐使用的参数，表示重复利用图片内存，减少内存分配，在4.4以前只有相同大小的图片内存区域可以复用，4.4以后只要原有的图片比将要解码的图片大既可以复用了。

图片占用内存大小：

public final int getByteCount() {

// int result permits bitmaps up to 46,340 x 46,340

return getRowBytes() \* getHeight();

}

一张 522x686 的 PNG 图片，我把它放到 drawable-xxhdpi 目录下，在三星s6上加载，占用内存2547360B，就可以用这个方法获取到。

我们发现 ARGB\_8888（也就是我们最常用的 Bitmap 的格式）的一个像素占用 4byte，那么 rowBytes 实际上就是 4\*width bytes。

那么结论出来了，一张 ARGB\_8888 的 Bitmap 占用内存的计算公式

bitmapInRam = bitmapWidth\*bitmapHeight \*4 bytes

一张522\*686的PNG 图片，我把它放到 drawable-xxhdpi 目录下，在三星s6上加载，占用内存2547360B，其中 density 对应 xxhdpi 为480，targetDensity 对应三星s6的密度为640：

522/480【原始Density】 \* 640【targetDensity】 \* 686/480 \*640 \* 4【AGB8888每像素所占大小】 = 2546432B

实际是有误差的，根据源码可以

if (willScale && decodeMode != SkImageDecoder::kDecodeBounds\_Mode) {

scaledWidth = int(scaledWidth \* scale + 0.5f);

scaledHeight = int(scaledHeight \* scale + 0.5f);

}

scaledWidth = int( 522 \* 640 / 480f + 0.5) = int(696.5) = 696

scaledHeight = int( 686 \* 640 / 480f + 0.5) = int(915.16666…) = 915

915\*696 = 2547360；

当然，也并不是所有格式的图片每个像素占用4字节，这和图片在加载时设置的Bitmap.Config有关，默认的是Bitmap.Config.ARGB\_8888，其他类型如下：

Bitmap.Config.ALPHA\_8 此时图片只有alpha值，没有RGB值，

一1个像素占用一个字节

Bitmap.Config.ARGB\_4444 一个像素占用2个字节，alpha(A)值，Red（R）值，Green(G)值，Blue（B）值各占4个bites共16bites,即2个字节

Bitmap.Config.ARGB\_8888 一个像素占用4个字节，alpha(A)值，Red（R）值，Green(G)值，Blue（B）值各占8个bites，共32bites,即4个字节。这是一种高质量的图片格式，在电脑上普通采用。它也是Android手机上一个Bitmap的默认格式。

Bitmap.Config.RGB\_565 一个像素占用2个字节，没有alpha(A)值，即不支持透明和半透明，Red（R）值占5个bites ，Green(G)值占6个bites ，Blue（B）值占5个bites,共16bites,即2个字节。对于没有透明和半透明颜色的图片来说，该格式的图片能够达到比较的呈现效果，相对于ARGB\_8888来说也能减少一半的内存开销。因此它是一个不错的选择。

TextView加载自定义字体内存占用：

db shell dumpsys meminfo <package\_name|pid> 查看内存分配情况

class TypefaceHelperKt {  
  
 companion object {  
 private const val TAG = "TypefaceHelperKt"  
  
 private val TYPEFACE\_CACHE = SimpleArrayMap<String, Typeface>()  
  
 @JvmStatic  
 fun get(context: Context, typePath: String): Typeface? {  
 *synchronized*(Companion.TYPEFACE\_CACHE) **{** if (!Companion.TYPEFACE\_CACHE.containsKey(typePath)) {  
 try {  
 val t = Typeface.createFromAsset(context.*assets*, typePath)  
 TYPEFACE\_CACHE.put(typePath, t)  
 } catch (e: Exception) {  
 LogUtil.w(TAG, "Could not get typeface '" + typePath  
 + "' because " + e.message)  
 return null  
 }  
 }  
 return TYPEFACE\_CACHE.get(typePath)  
 **}** }  
 }  
}