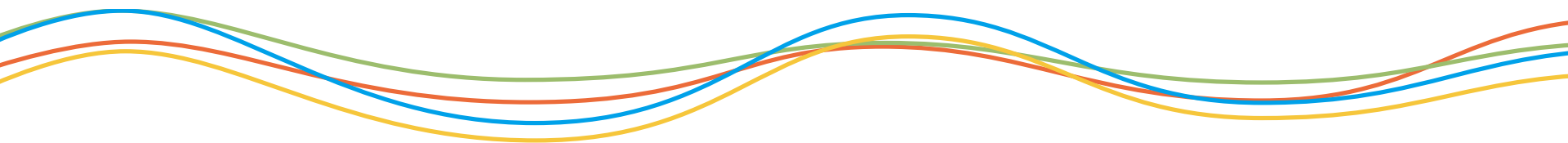


Android应用 启动速度和内存优化实践

梁炳辉 / 猎豹安全大师 CN



猎豹移动（NYSE: CMCM），于2010年10月由金山安全和可牛影像合并而成，继承了金山17年的安全技术积累和可牛影像的互联网基因，是GP全球非游戏类应用榜单排名第三的移动应用开发商。致力于为全球的移动用户提供更快速，更易用，更安全的移动互联网体验。截至2016年12月，猎豹移动产品在全球范围内安装量达38.10亿，全球移动端月活跃用户高达6.23亿。公司核心产品包括**猎豹清理大师**、**猎豹安全大师**等。



最值得信赖的病毒查杀和隐私保护工具（原金山手机毒霸）

1.0



2.0



3.0



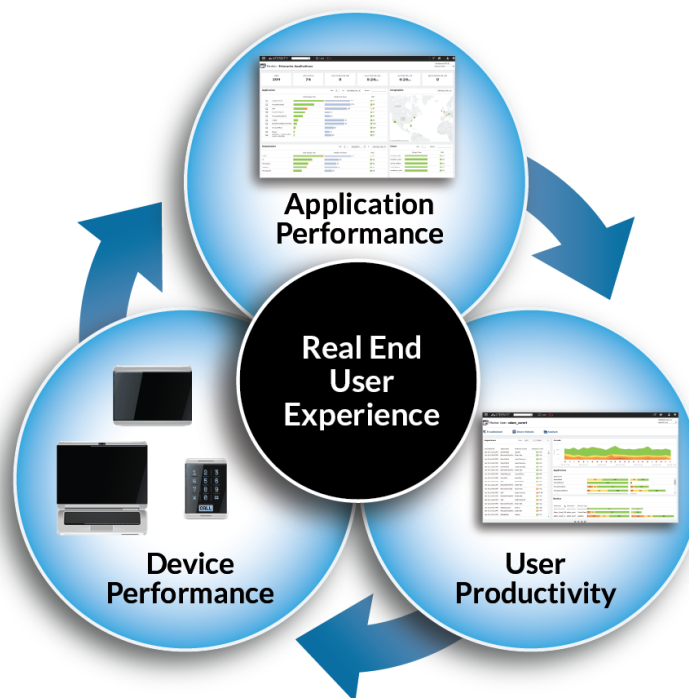
4.0



为什么要做 启动速度优化



轻快的体验



中低端手机

安心的保护

- ✓ 以用户体验为核心（用户至上）
- ✓ 中低端手机也能有高端手机一样的轻快体验
- ✓ 提高用户尝试使用其他功能的兴趣
- ✓ 产品留存

我们能做什么？



开屏页 →



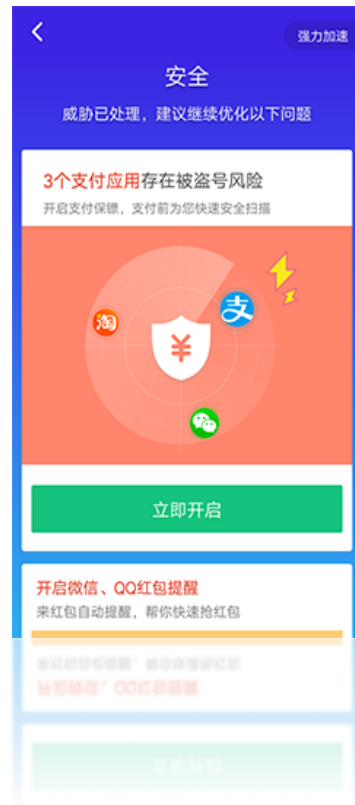
主页 →



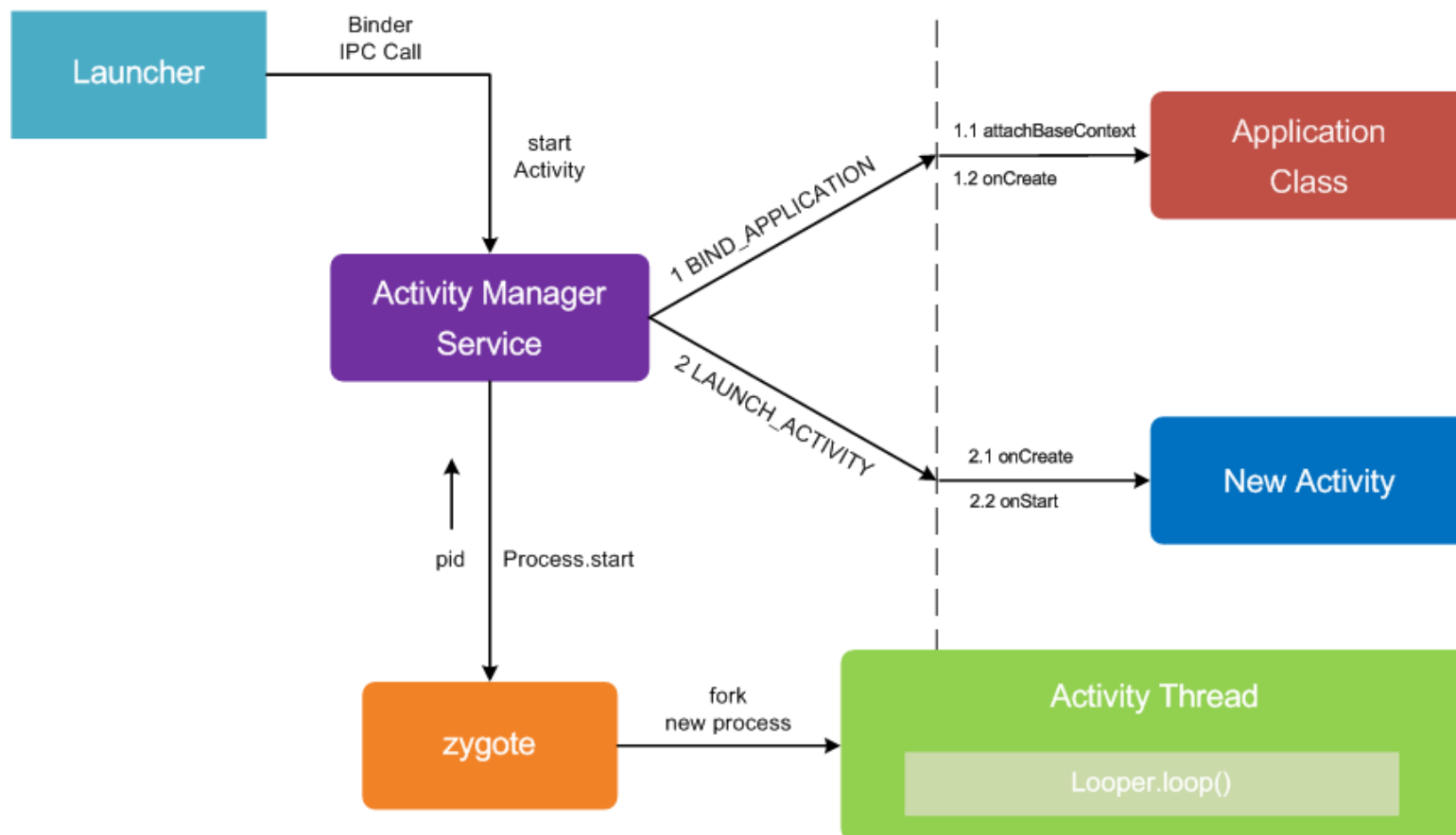
扫描页 →

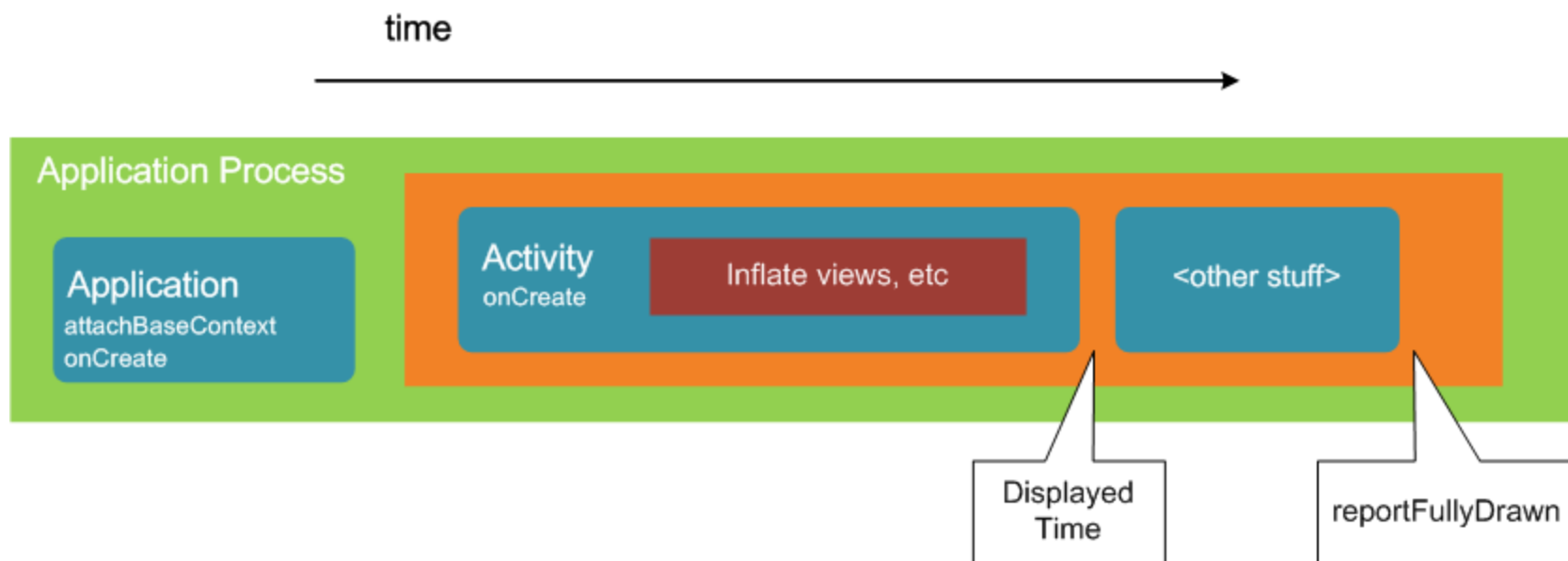


风险页 →



结果页





- ✓ Application初始化
- ✓ 首屏Activity的渲染

-  Strict Mode
-  Layout Inspector
-  Hierarchy Viewer
-  Trace View
-  AspectJ

启动过程分析

- ✓ 启动耗时的系统版本两级分化很大
- ✓ 低端机型平均耗时：5秒 → 2秒
- ✓ 中高端机型平均耗时：3秒 → 1.2秒
- ✓ 平均耗时：1.8秒

耗时区间	1秒内	1至2秒	2至3秒	3至4秒	4至5秒	大于5秒
1秒内	36.75%	39.31%	10.90%	5.62%	2.31%	5.11%
2秒内	76.06%					
3秒内	86.96%					
4秒内	92.58%					
5秒内	94.89%					

越少越好

- ✓ 布局层次
- ✓ 控件元素
- ✓ 过度绘制



极简布局

过度渲染



无图界面

- ✓ Drawable/自绘
- ✓ 文字替代单色图
- ✓ 自绘控件



简单
极致

专用车道

- ✓ 插件框架
- ✓ 执行顺序
- ✓ 主线程刷新
- ✓ 其他

错峰出行

- ✓ ViewStub
- ✓ PostDelayed
- ✓ onFirstFrameDrawn

- ✚ 界面元素/层级 越少越好
- ✚ 主界面尽量 少用图片
- ✚ 主线程业务走 专用车道
- ✚ 非紧急业务 错峰出行

App性能优化 还要做什么？

- 使用更加轻量的数据结构

- 使用更小的图片

- 复用系统自带资源

- 注意Cursor对象是否及时关闭

- 特别留意单例对象不合理的持有

...

- AS Memory Monitor

- LeakCanary

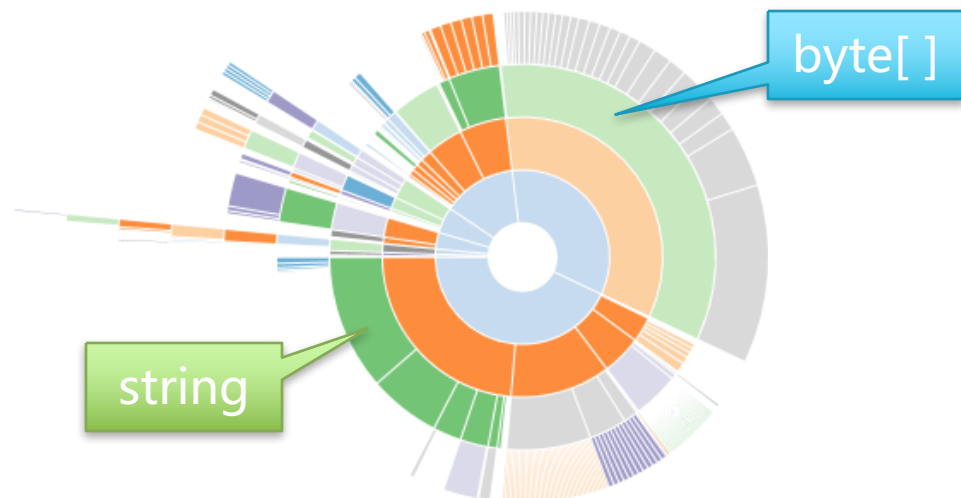
- MAT

如何做到立竿见影？

基本原则

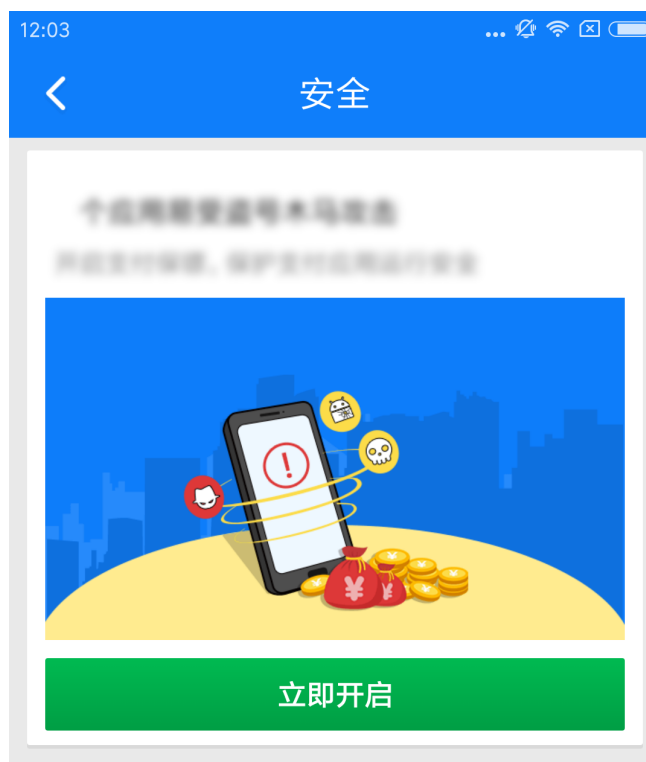
抓大放小

Class Name	Total Count	Heap Count	Size of	Shallow Size	Retained Size
byte[]	3424	1209	0	10030732	10030732
char[]	34707	25301	0	1478802	1478802
ArtMethod (java.lang.reflect)	83018	29205	48	1401840	1401840
String[] (java.lang)	2296	241	0	805908	805908
ArtMethod[] (java.lang.reflect)	14852	6022	0	760400	760400
String (java.lang)	92398	26072	24	625728	625728
ArtField[] (java.lang.reflect)	5798	2590	0	434388	434388
ArtField (java.lang.reflect)	48828	14893	24	357432	357432
Object[] (java.lang)	4886	2897	0	107388	107388
Object[] (java.lang)	4886	2897	0	107388	107388
ArtField[] (java.lang.reflect)	48858	14893	24	357432	357432
ArtField (java.lang.reflect)	48858	14893	24	357432	357432



一张图片到底有多耗内存？

问题：640 x 378的PNG图片需要多少内存



res	分辨率	DensityDPI	机型
xhdpi	720x1280	320	Galaxy Nexus
xxhdpi	1080x1920	480	Nexus 5
xxxhdpi	1440x2560	640	Nexus 6

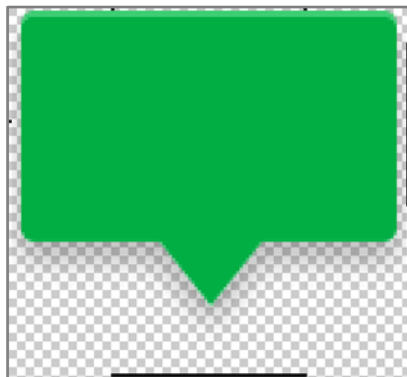
scale = (float) screenDensity / bitmapDensity
scaledWidth = int(bitmapWidth * scale + 0.5f)
scaledHeight = int(bitmapHeight * scale + 0.5f)
bytes = scaledWidth * scaledHeight * bytesPerPixel

机型1 : Galaxy Nexus **945KB**
screenDensity(320), xhdpi(320), bitmap(640*378)

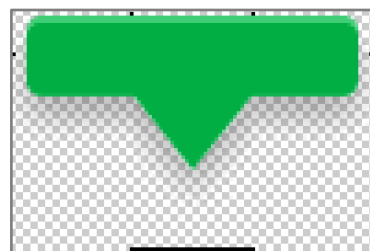
机型2 : Nexus 6 **3.7MB**
screenDensity(640), xhdpi(320), bitmap(640*378)

图片使用原则一：使用.9，保证其足够精简

修复失效功能

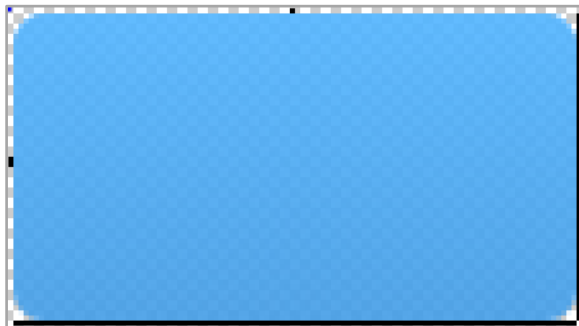


230KB



100K

安全防御

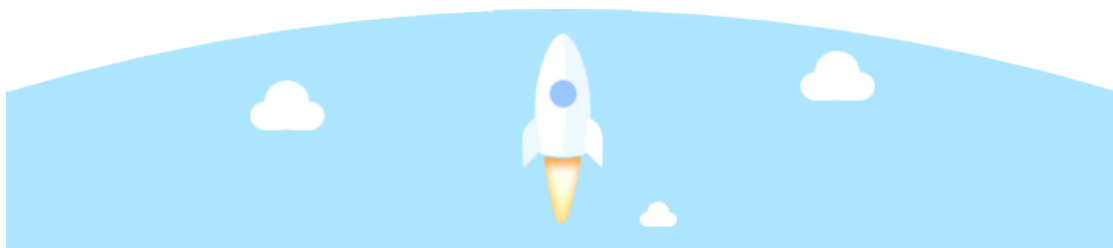


230KB

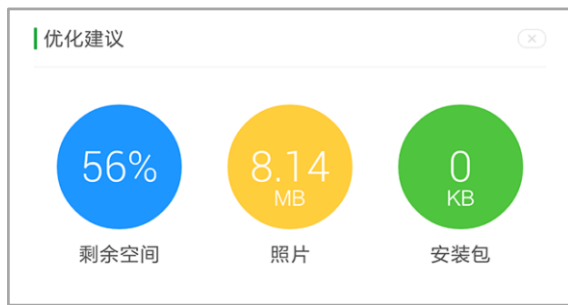


11KB

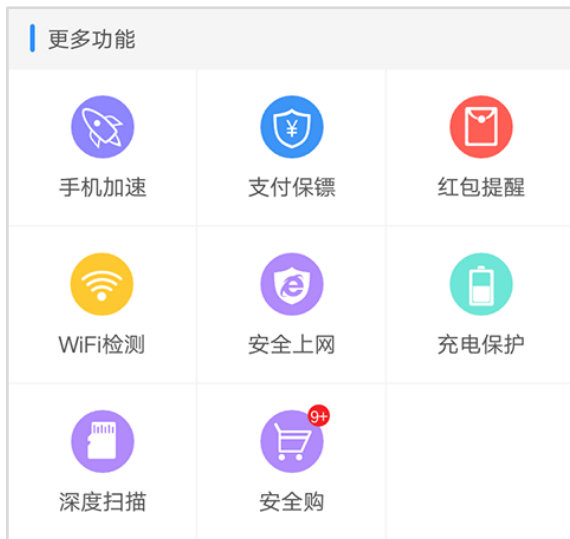
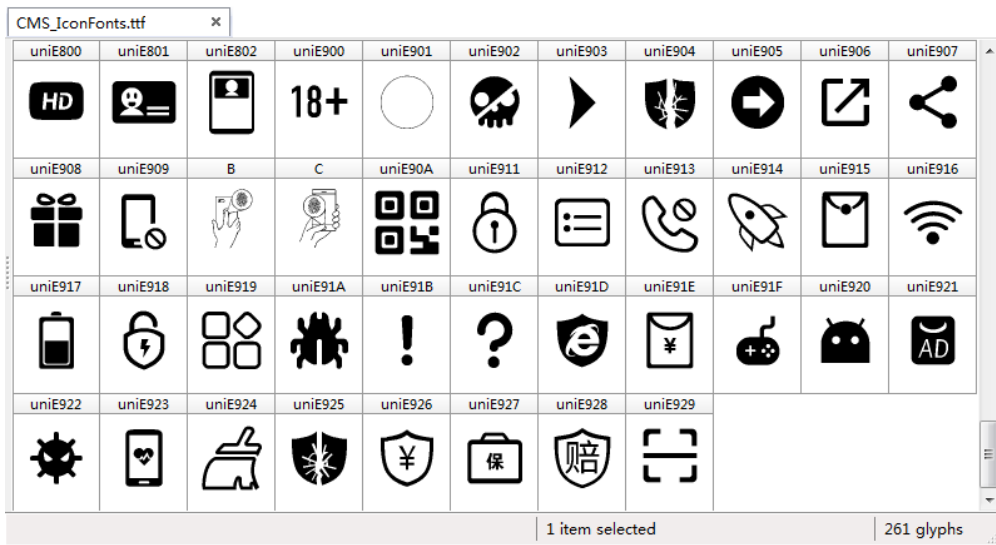
图片使用原则二：化大为小，化繁为简



图片使用原则三：用小图或者不用图



文字替换单色图片



图片使用原则四：尽量使用RGB565

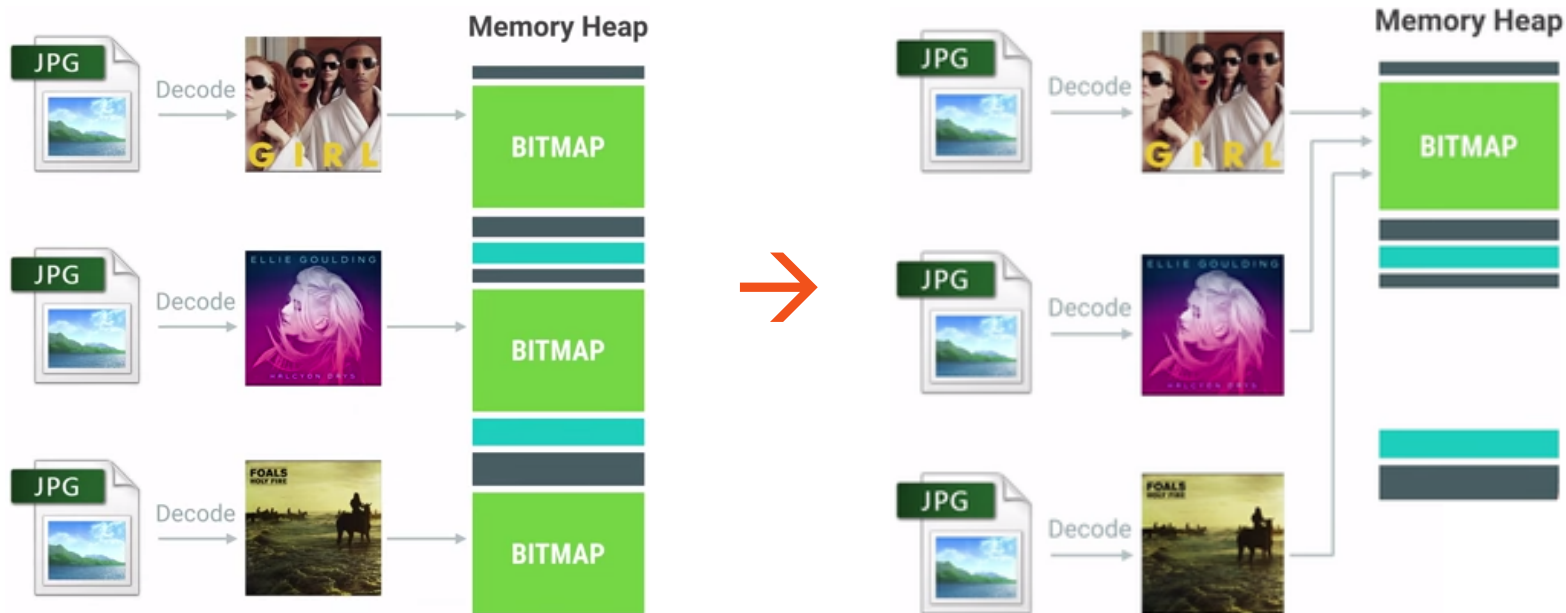


RGB565 450KB



ARGB8888 900KB

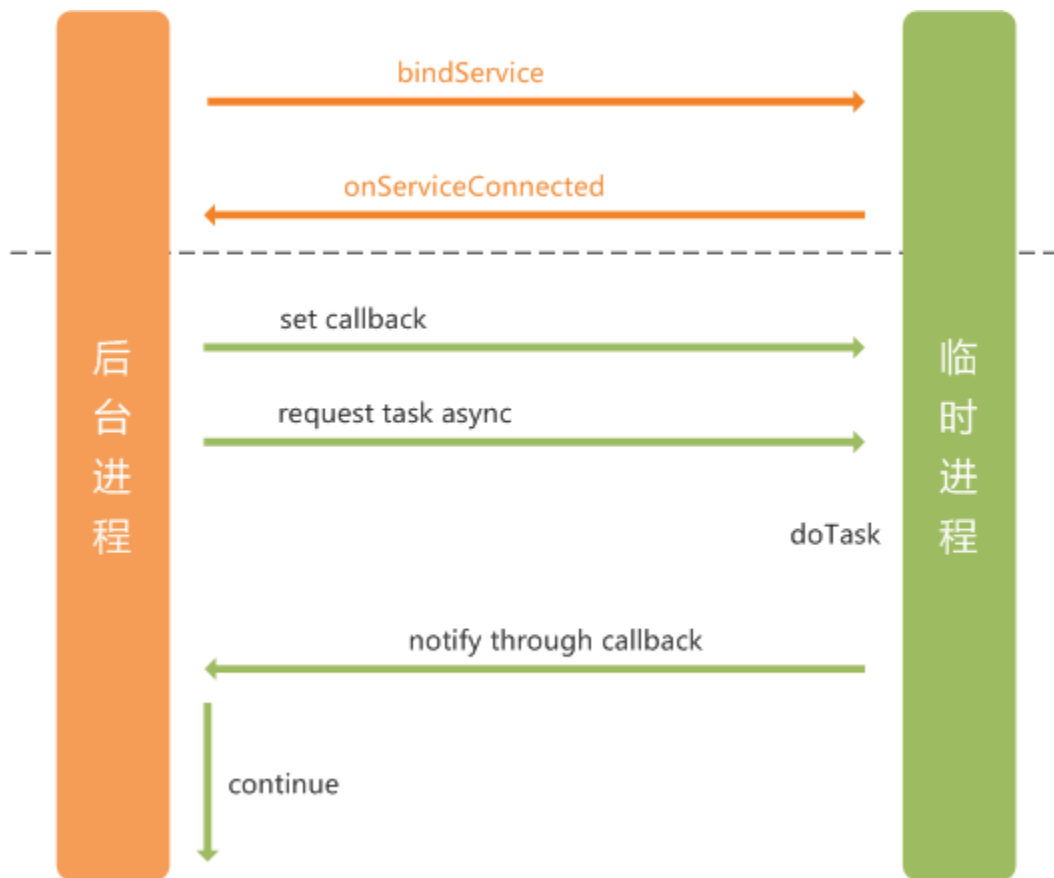
图片使用原则五：Bitmap内存复用



- ✚ 使用.9，尽量保证其足够精简
- ✚ 化大为小，化繁为简
- ✚ 用小图或者不用图
- ✚ 文字替换单色图片
- ✚ 尽量使用RGB565
- ✚ Bitmap内存尽量复用
- ✚ 内存中保存压缩后的图片
- ✚ 用完及时释放图片内存

- ✓ WebView内存泄漏
- ✓ 第三方服务内存泄漏
- ✓ 特殊场景功能





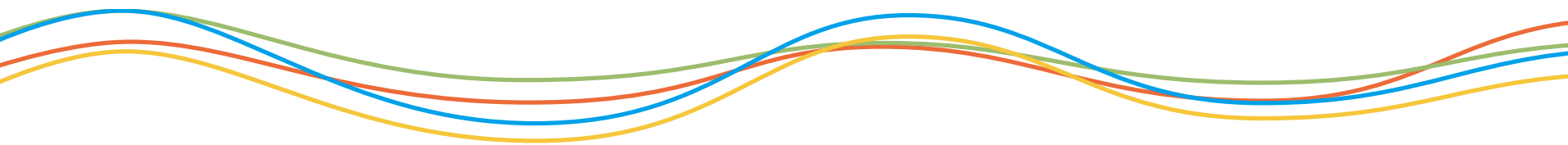
- ✚ 找出图片内存大户并优化
- ✚ 找出废弃逻辑代码并删除
- ✚ 找出执行过程耗费大量内存、具备一次性性质的逻辑迁到临时进程
- ✚ 无法解决泄漏（如WebView）移至其他进程
- ✚ 坚决防止Activity内存泄漏

GC是非常耗时的操作，使用越少内存程序越流畅。

核心思想

不做事情肯定不耗费内存

Q & A



THANK YOU

