1、谈谈 60fps

2、用户感觉操作卡顿的原因是什么？

3、你都使用那些 UI 性能优化工具？

4、ANR 是什么？如何解决 ANR 现象？

5、开发中如何避免 UI 性能问题？

答案

1、谈谈 60fps

答：

Android 把达到流畅的帧率规定为 60fps。

因此，开发 App 的帧率性能目标就是保持在 60fp，

换算关系：60 帧/秒----------1000ms/60 帧=16ms/帧；

准则：尽量保证每次在 16ms 内处理完所有的 CPU 与 GPU 计算、绘制、渲染等操作，否则会造成丢帧卡顿问题。

2、用户感觉操作卡顿的原因是什么？

答：

1) 表面原因：
   a) 人为在 UI 线程中做轻微耗时操作，导致 UI 线程卡顿；
   b) 布局 Layout 过于复杂，无法在 16ms 内完成渲染；
   c) 同一时间动画执行的次数过多，导致 CPU 或 GPU 负载过重；
   d) View 过度绘制，导致某些像素在同一帧时间内被绘制多次，从而使 CPU 或 GPU 负载过重；
   e) View 频繁的触发 measure、layout，导致 measure、layout 累计耗时过多及整个 View 频繁的重新渲染；
   f) 内存频繁触发 GC 过多（同一帧中频繁创建内存），导致暂时阻塞渲染操作；
   g) 冗余资源及逻辑等导致加载和执行缓慢；
   h) 臭名昭著的 ANR。
2) 深层原因

Android 系统每隔 16ms 发出 VSYNC 信号，触发对 UI 进行渲染，如果这个过程保证在 16ms 以内就能达到一个流畅的画面。那么如果操作超过了 16ms 就会发生下面的情况：

当系统发出 VSYNC 信号，但此时无法进行渲染(原因可能是 cpu 在做别的操作)，那么就会导致丢帧的现象。这样的话，绘制就会在下一个 16ms 的时候才进行绘制，即使只丢一帧，用户也会发现卡顿的。

因此 16ms 能否完整的做完一次操作直接决定了卡顿性能问题。

3、你都使用那些 UI 性能优化工具？

1) GPU OverDraw

GPU 过度绘制，是 Android 提供的用于分析 UI 绘制性能的工具。该工具在屏幕上用四种颜色直观地显示出 GPU 在绘制 UI 界面时过度绘制的区域。

代码优化：

1、 优化布局层级；

2、 减少没必要的背景；

当 Activity 的布局文件设置了背景，则调用

```
getWindow().setBackgroundDrawable(null);
```

将系统提供的默认背景去掉。

3、 暂时不显示的 View 设置为 GONE 而不是 INVISIBLE；

4、 自定义 View 的 onDraw 方法设置 canvas.clipRect()指定绘制区域或通过 canvas.quickreject()减少绘制区域等。

5、 自定义 View 的 Paint 要在构造器中创建，不要在 onDraw 方法中创建。

2)优化工具之 Hierarchy View

Android 提供了 Hierarchy Viewer 工具，该工具称为层级观察器。用图形方式查看布局的层次和布局中的控件。

在模拟器运行的情况下，使用该工具可以将当前的 Activity 中的 UI 组件们以对象树的形式展现出来，每一个组件所包含的属性也能看到。在对象树上的任意节点可以看到该节点及以下节点的显示效果。使用 HierarchyView 能深入全面的理解 xml 布局文件，更可以通过它来学习别人优秀的布局技巧。

使用 Hierarchyviewer 还可以获取屏幕上感兴趣的颜色值。

运行程序

用 Hierarchy View 分析 RelativeLayout 布局

按上图步骤操作，在 Hierarchy View 中找到并显示登陆布局。

每个圆角矩形表示一个 View，里面的三个按钮从左至右依次是测量(Measure)、布局(Layout)和绘制(Draw)的速度。

绿色是正常、黄色慢、红色最慢。

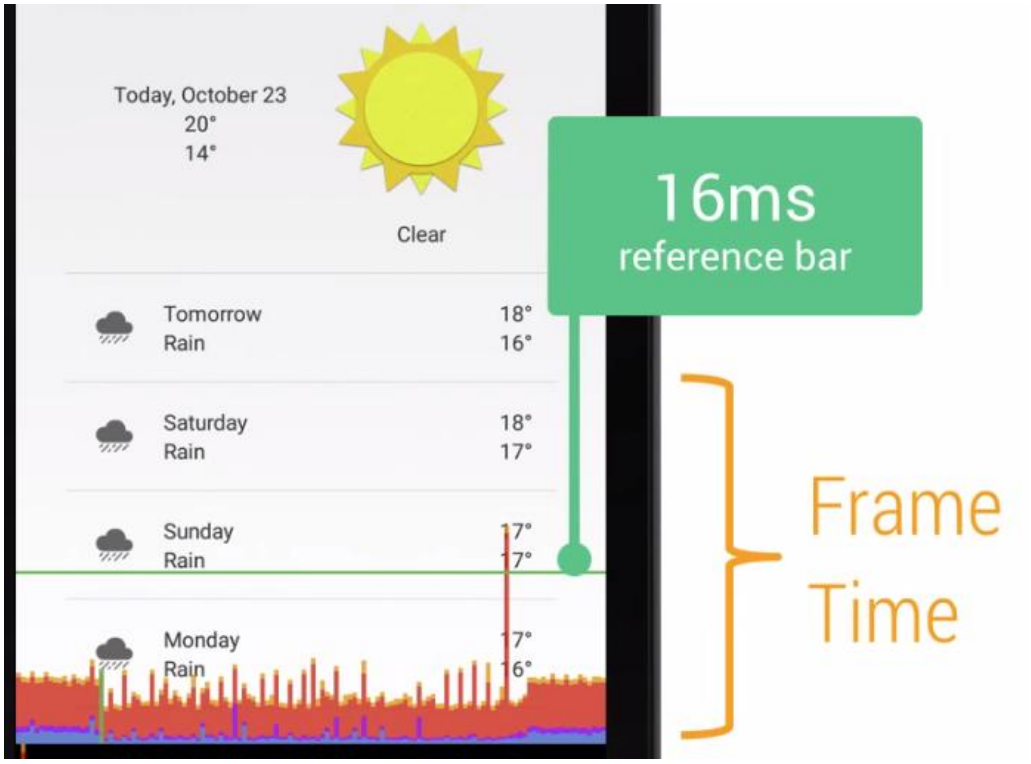
依次统计上图中登陆页面的每个标签的绘制时间，得出相对布局制作的登陆页面的时间。

3)优化工具之 profile GPU rendering

Android 界面流畅度可以通过 GPU 呈现模式图来观察。

通过开发者选项中 GPU 呈现模式图工具来进行流畅度考量的流程是（注意：如果是在开启应用后才开启此功能，记得先把应用结束后重新启动）在 dev-settings->开发者选项->GPU 呈现模式(profile GPU rendering)：

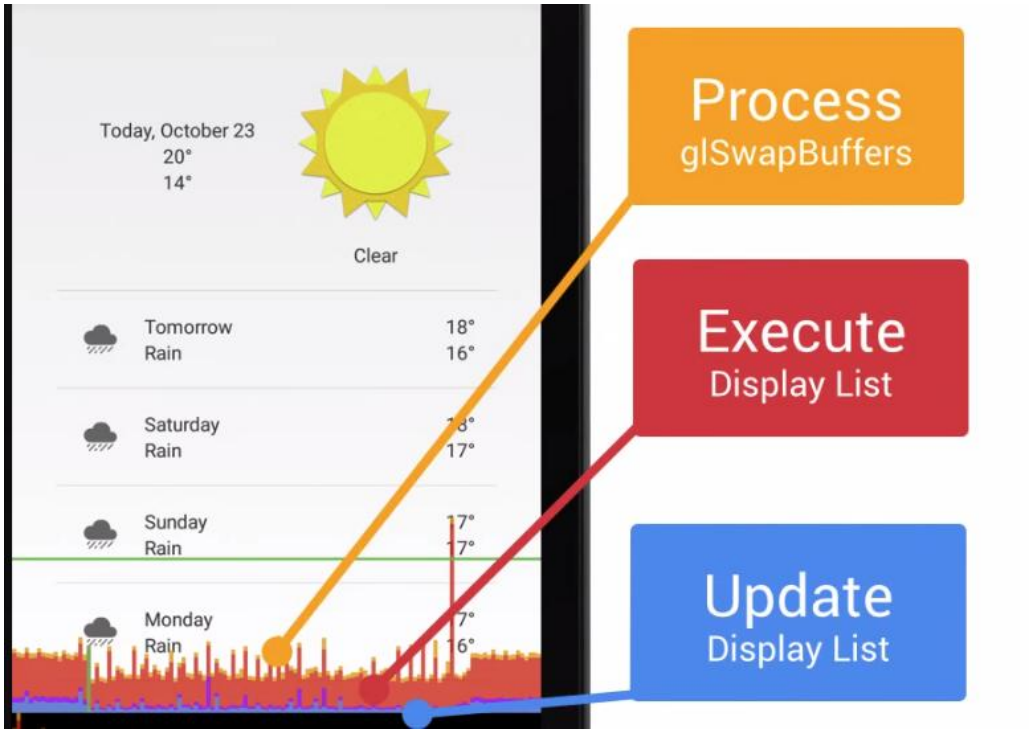
Profile GPU Rendering 用于在屏幕上实时显示 GPU 渲染每一帧图像花费的时间（单位：ms）。



渲染时间用柱状图中：

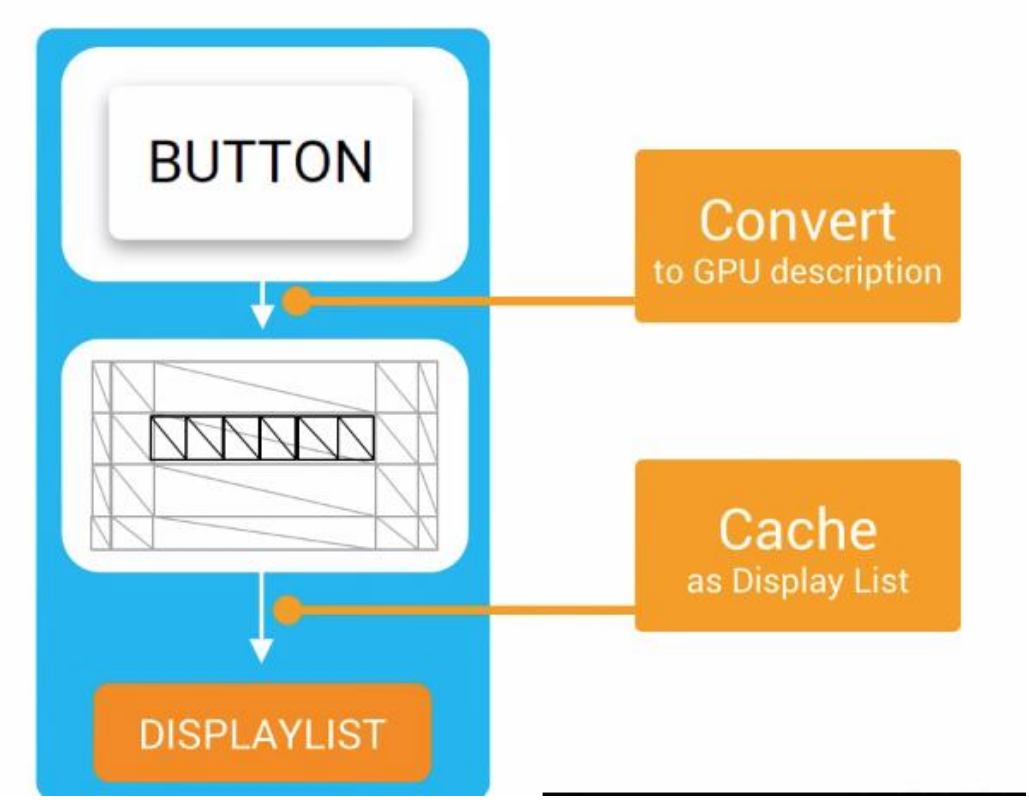
1、 绿线代表 16ms，也就是要尽量保证所有柱状图都在这条线下方。

2、 每一条柱状图都由 3 部分组成，蓝色、红色和黄色，代表渲染的 3 个不同的阶段，通过分析这三个阶段的时间就可以找到渲染时的性能瓶颈。



3、 蓝色部分表示将 View 转换为 GPU 能识别的格式和缓存 display list 的时间。

在一个 View 实际被渲染前，它需要先转换为 GPU 能识别的格式。这一步完成之后，输出结果就会被系统作为 display list 缓存起来。



蓝色部分记录了这一帧对所有需要更新的 view 完成这两步花费的时间。当它很高的时候，说明有很多 view 突然无效（invalidate）了，或者是有几个自定义 view 在 onDraw 函数中做了特别复杂的绘制逻辑。

4、 红色部分代表 OpenGL （Android 2D 渲染引擎）绘制 display list 的时间。

View 越复杂，OpenGL 绘制所需要的命令也越复杂。如果红色这一段比较高，复杂的 view 都可能是罪魁祸首。还有值得注意的是比较大的峰值，这说明有些 view 重复提交了，也就是绘制了多次，而它们可能并不需要重绘。

5、 橙色部分代表 CPU 等待 GPU 渲染的时间。

这部分是阻塞的，CPU 会等待 GPU 直到确认收到了命令，如果这里比较高，说明 GPU 做的任务太多了，通常是由于很多复杂的 view 绘制从而需要过多的 OpenGL 渲染命令去处理。

4)优化工具之 Lint

冗余资源及逻辑等也可能会导致加载和执行缓慢，用 Lint 工具可以发现需要优化的问题的。

在 Android Studio 1.4 版本中使用 Lint 最简单的办法就是将鼠标放在代码区点击右键->Analyze->Inspect Code－>界面选择你要检测的模块->点击确认开始检测。

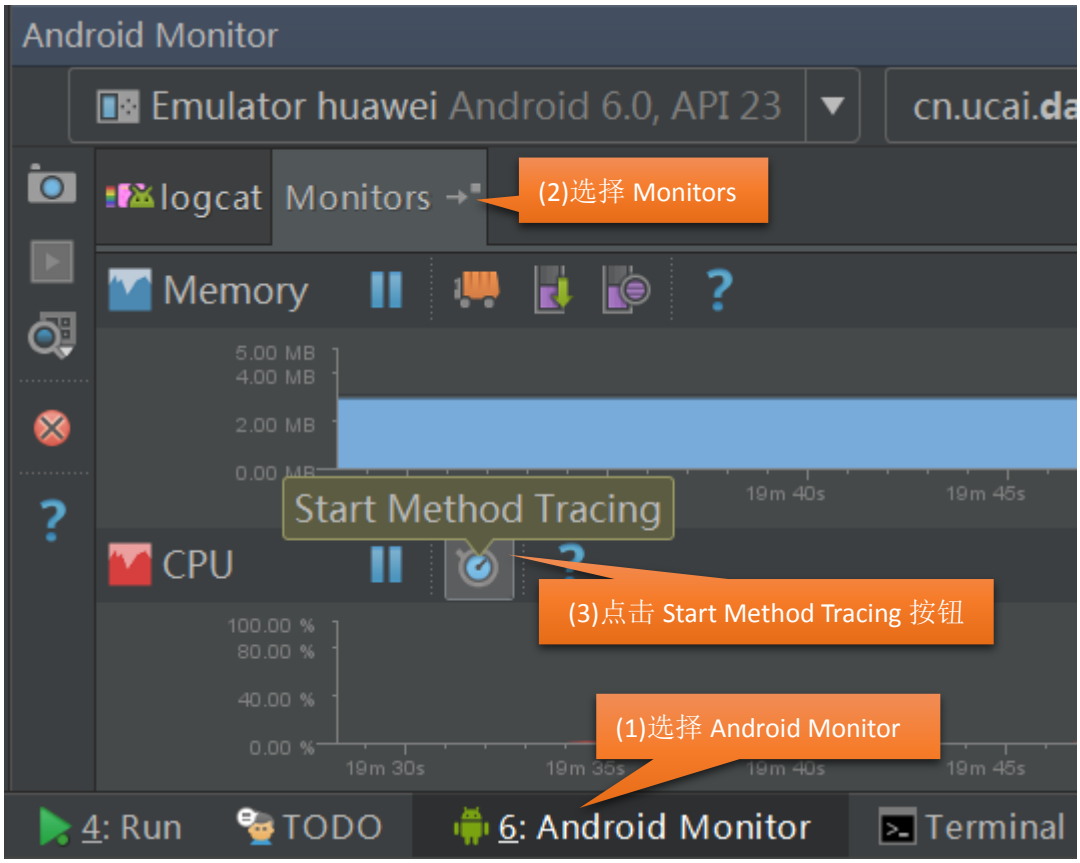
5)优化工具之 Traceview

Traceview 是 Android 平台特有的数据采集和分析工具，主要用于分析 Android 中应用程序的 hotspot。

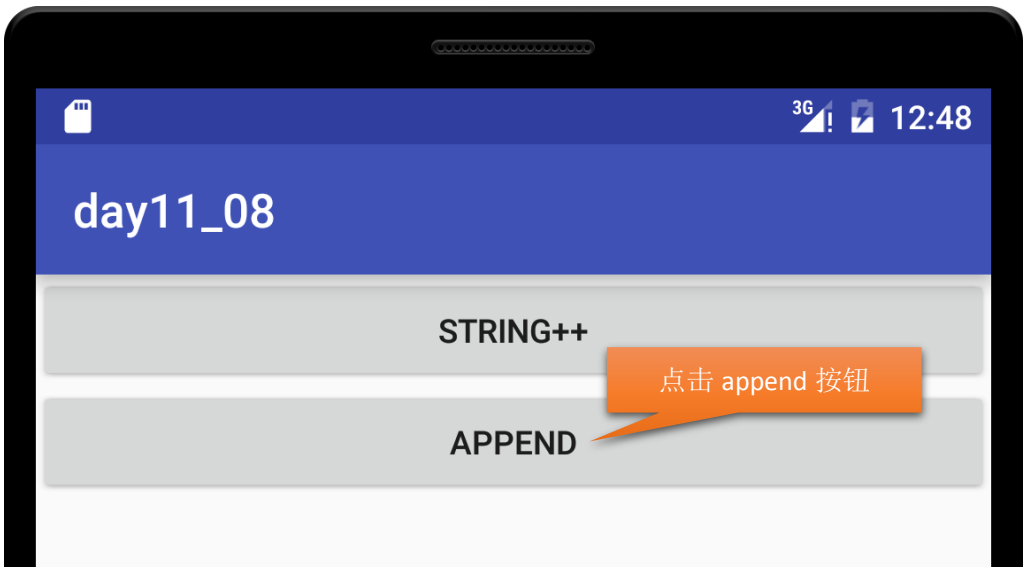
Traceview 本身只是一个数据分析工具，而数据的采集则需要使用 Android SDK 中的 Debug 类或者利用 DDMS 工具。

步骤 1、运行【案例-7】的程序

步骤 2、运行 Start Method Tracing

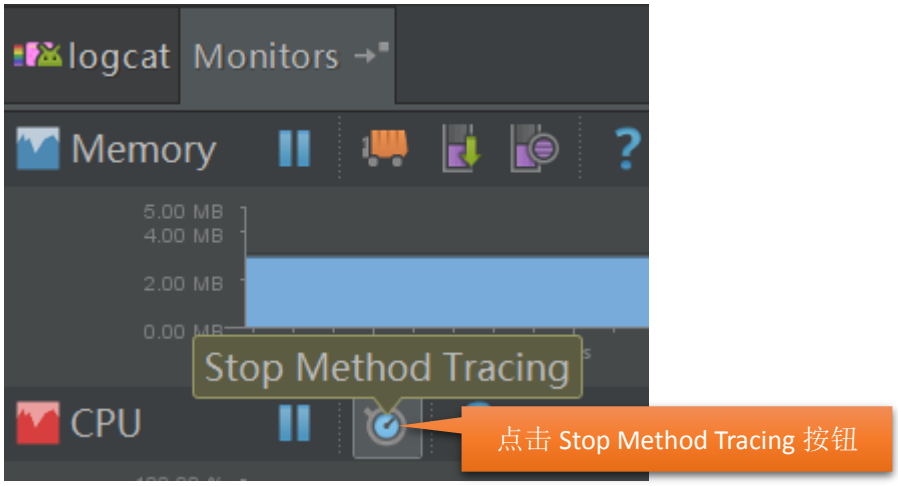


步骤 3、运行 StringBuilder.append 连接字符串代码

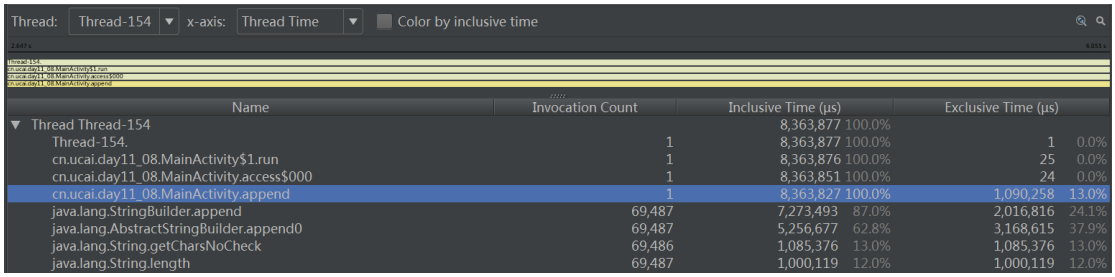


步骤 4、捕获代码运行的数据

当前屏幕上出现“完成”的消息。按下图操作，停止数据捕获



步骤 5、分析代码运行的数据



可以看到 apend 运行花了 8 秒多一点儿的时间。

4、ANR 是什么？如何解决 ANR 现象？

答：

ANR 是 Application Not Response（应用无响应）的简称。

当 UI 线程在 5 秒之后无法响应用户的触摸操作，就会发生 ANR 现象。

ANR 是直接卡死 UI 不动且必须要解掉的 bug，发开中必须尽量避免 ANR 的出现。

造成 ANR 现象的场景有：

1、 在 UI 线程执行了一个耗时操作，此时用户在该耗时操作尚未执行完毕时，触摸了屏幕，导致 UI 线程无法在 5 秒内响应该操作。

2、 广播接收者在 onReceive 方法中执行了一段耗时操作，此时用户在该耗时操作尚未执行完毕时，触摸了屏幕，导致 UI 线程无法在 5 秒内响应该操作。

3、 在 Service 的 on 打头的方法中执行了一段耗时操作，此时用户在该耗时操作尚未执行完毕时，触摸了屏幕，导致 UI 线程无法在 5 秒内响应该操作。

总之，在 UI 线程中执行了一段耗时操作，导致 UI 线程在 5 秒内响应用户的触摸操作。

避免 ANR

不在 UI 线程执行任何耗时操作，如：

1、 执行 IO 操作

2、 执行数据库读写操作

3、 执行网络操作

4、 自定义 View 中 onDraw 中执行耗时操作

5、 其它任何导致在 UI 线程阻塞的操作。如长时间的循环、递归等。

避免 ANR

不在 UI 线程执行任何耗时操作，如：

6、 执行 IO 操作

7、 执行数据库读写操作

8、 执行网络操作

9、 自定义 View 中 onDraw 中执行耗时操作

10、 其它任何导致在 UI 线程阻塞的操作。如长时间的循环、递归等。

如何捕获 ANR

1、 在 LogCat 中可以看到 ANR 的信息。

2、在 Android Device Monitor 的 FileExplore 中找到 data/anr 文件夹下有一个 traces.txt 文件，该文件中描述了 ANR 的详细信息。导出该文件并用文本编辑软件打开。

5、开发中如何避免 UI 性能问题？

答：

在开发应用时应该尽量从项目代码架构搭建及编写时就避免一些 UI 性能问题，常见的注意事项如下：

1. 布局优化
   - ✧ 尽量使用 include、merge、ViewStub 标签;
   - ✧ 尽量不存在冗余嵌套及过于复杂布局（譬如 10 层就会直接异常）;
   - ✧ 尽量使用 GONE 替换 INVISIBLE;
   - ✧ 使用 weight 后尽量将 width 和 heigh 设置为 0dp 减少运算;
   - ✧ Item 存在非常复杂的嵌套时考虑使用自定义 Item View 来取代，减少 measure 与 layout 次数等。
   - ✧ 列表及 Adapter 优化
   - ✧ 尽量复用 getView 方法中的相关 View，不重复获取实例导致卡顿;
   - ✧ 列表尽量在滑动过程中不进行 UI 元素刷新等。
2. 背景和图片等内存分配优化
   - ✧ 尽量减少不必要的背景设置;
   - ✧ 图片尽量压缩处理显示;
   - ✧ 尽量避免频繁内存抖动等问题出现。
3. 自定义 View 等绘图与布局优化
   - ✧ 尽量避免在 draw、measure、layout 中做过于耗时及耗内存操作，尤其是 draw 方法中;
   - ✧ 尽量减少 draw、measure、layout 等执行次数。
4. 避免 ANR
   - ✧ 不要在 UI 线程中做耗时操作，遵守 ANR 规避守则。