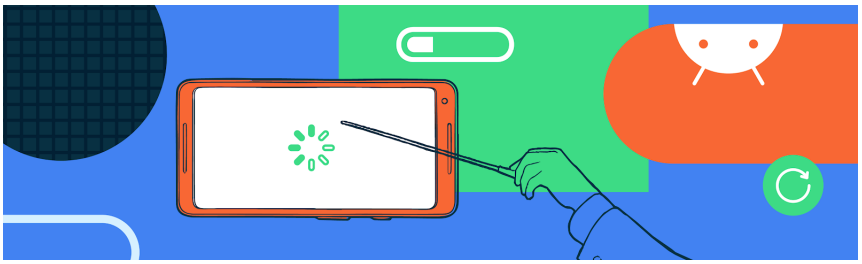
High refresh rate rendering on Android

Ady Abraham

27 April 2020



很长一段时间以来，手机的显示频率为 60Hz。应用程序和游戏开发者可以假设刷新率为 60Hz，帧截止时间为 16.6ms，一切都会正常工作。现在已经不是这样了。新的旗舰设备具有更高的刷新率显示，提供更流畅的动画、更低的延迟和更好的用户体验。还有一些设备支持多种刷新率，比如 Pixel 4，它同时支持 60Hz 和 90Hz。

60Hz 显示器每 16.6ms 刷新一次显示内容，这意味着图像将以 16.6ms 的倍数 (16.6ms、33.3ms、50ms 等) 显示。支持多种刷新率的显示器提供了更多的选项，以不同的速度渲染而不出现抖动。例如，一个不能维持 60fps 渲染的游戏必须在 60Hz 显示器上一直降到 30fps 才能保持平滑而无打顿 (由于显示器仅限于以 16.6ms 的倍数渲染图像，下一个可用的帧速率是每 33.3ms 或 30fps 一帧)。在 90Hz 的设备上，同样的游戏可以降到 45fps(每帧 22.2ms)，以提供更流畅的用户体验。支持 90Hz 和 120Hz 的设备可以以每秒 120、90、60(120/2)、45(90/2)、40(120/3)、30(90/3)、24(120/5) 等帧平滑地显示内容。

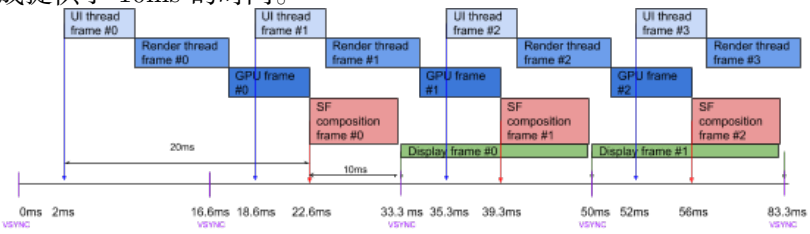
1 高速渲染

渲染速率越高，维持该帧速率就越困难，这就是因为相同工作量的可用时间更少。要在 90Hz 下渲染，应用程序只有 11.1ms 来生成帧，而在 60Hz 时还有 16.6ms。

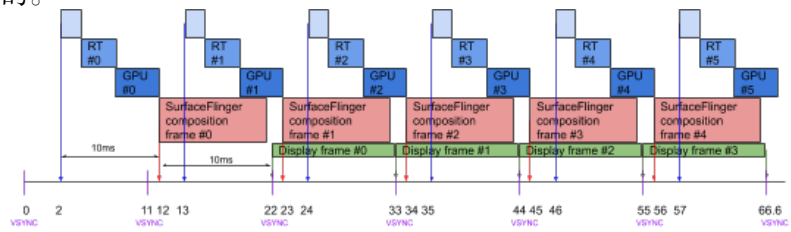
为了证明这一点，让我们看看 Android 用户界面渲染管道。我们可以将帧渲染分成大约 5 个流水线阶段：

1. 应用程序的 UI 线程处理输入事件，调用应用程序的回调，并更新视图层次结构的已记录绘图命令列表
2. 应用程序的 RenderThread 将录制的命令发送给 GPU
3. GPU 绘制帧
4. SurfaceFlinger 是负责在屏幕上显示不同应用程序窗口的系统服务，它合成屏幕并将帧提交给显示 HAL
5. 显示画面

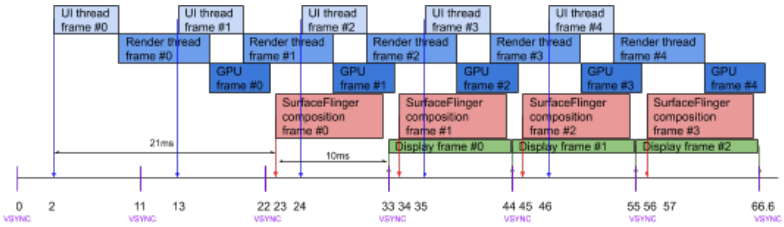
整个管道由 Android Choreographer 控制。Choreographer 基于显示垂直同步 (vsync) 事件，该事件指示显示器开始扫描图像和更新显示像素的时间。Choreographer 基于 vsync 事件，但是对于应用程序和 SurfaceFlinger 有不同的唤醒偏移量。下图显示了以 60Hz 运行的 Pixel 4 设备上的管道，其中应用程序在 vsync 事件后 2ms 被唤醒，SurfaceFlinger 在 vsync 事件后 6ms 被唤醒。这为应用程序生成帧提供了 20ms 的时间，而 SurfaceFlinger 为屏幕合成提供了 10ms 的时间。



当以 90Hz 运行时，应用程序在 vsync 事件发生 2ms 后仍然被唤醒。然而，SurfaceFlinger 在 vsync 事件发生后 1ms 被唤醒，以具有相同的 10ms 来合成屏幕。另一方面，应用程序只有 10ms 的时间来渲染一个帧，这是非常短的。



为了缓解这种情况，Android 中的 UI 子系统使用“提前渲染”(render ahead)(在启动帧的同时延迟其显示) 来深化管道，并将帧渲染延迟一个 vsync。这使得应用程序在 21ms 内产生一个帧，同时保持 90Hz 的吞吐量。



一些应用程序 (包括大多数游戏) 都有自己的自定义渲染管道。这些管道可能有更多或更少的阶段，这取决于它们要完成的任务。一般来说，随着管道越深，可以并行执行更多的阶段，从而提高了总体吞吐量。另一方面，这会增加单个帧的延迟 (延迟将是 $number\_of\_pipeline\_stages \times longest\_pipeline\_stage$)。这种权衡需要仔细考虑。

2 利用多种刷新率

如上所述，多个刷新率允许使用更广泛的可用渲染速率范围。这对于可以控制渲染速度的游戏和需要以给定速率渲染内容的视频播放器特别有用。例如，要在 60Hz 的显示器上播放 24fps 的视频，需要使用3:2 的下拉算法，这会产生抖动。但是，如果该设备有一个能够以本机方式渲染 24fps 内容的显示器 (24/48/72/120Hz)，则无需下拉和与之相关的抖动。

设备运行的刷新率由 Android 平台控制。应用程序和游戏可以通过各种方法影响刷新率 (如下所述)，但最终决定权由平台决定。当屏幕上出现多个应用程序且平台需要满足所有应用程序时，这一点至关重要。一个很好的例子就是 24fps 的视频播放器。24Hz 对于视频播放来说可能很好，但对于响应式用户界面来说却很糟糕。一个只有 24Hz 的动画通知让人感觉难受。在这种情况下，平台将设置刷新率以确保屏幕上的内容看起来很好。

因此，应用程序可能需要知道当前的设备刷新率。这可以通过以下方式实现：

- SDK:
  - 用 DisplayManager.DisplayListener 向显示侦听器注册以及通过 Display.getRefreshRate 查询刷新率
- NDK:
  - 用 AChoreographer_registerRefreshRateCallback 注册回调 (API level 30)

应用程序可以通过在 Window 或 Surface 设置帧速率来影响设备刷新率。这

是 Android 11 中引入的一个新功能，允许平台知道调用应用程序的渲染意图。应用程序可以调用以下方法之一：

- SDK
  - Surface.setFrameRate
  - SurfaceControl.Transaction.setFrameRate
- NDK
  - ANativeWindow_setRrameRate
  - ASurfaceTransaction_setFrameRate

有关如何使用这些 API，请参阅帧速率指南。

系统将根据 Window 或 Surface 上编程的帧速率选择最合适的刷新率。

在不存在 setFrameRate API 的旧版 Android (Android 11 之前)，应用程序仍然可以通过直接设置 WindowManager.LayoutParams.preferredDisplayModeId，从 Display.getSupportedModes 中选择一种可用模式来影响刷新率。从 Android 11 开始，这种方法是不可取的，因为平台不知道应用程序的渲染意图。例如，如果一个设备支持 48Hz、60Hz 和 120Hz，并且屏幕上有两个应用程序分别调用 setFrameRate(60, ⋯) 和 setFrameRate(24, ⋯)，平台可以选择 120Hz 并使这两个应用程序都满意。另一方面，如果这些应用程序使用 preferredDisplayModeId，它们可能会将模式分别设置为 60Hz 和 48Hz，从而使平台无法选择设置 120Hz。该平台将选择 60Hz 或 48Hz，都会使一个应用程序不高兴。

3 变化

刷新率并不总是 60Hz —不要假设 60Hz，也不要基于那个历史文物硬编码假设。

刷新率并不总是恒定的—如果您关心刷新率，则需要注册一个回调函数，以确定刷新率何时更改，并相应地更新内部数据。

如果你没有使用 Android UI 工具箱，并且有自己的自定义渲染器，请考虑根据当前刷新率更改渲染管道。通过在 OpenGL 上使用 eglPresentationTimeANDROID 或在 Vulkan 上使用 VkPresentTimesInfoGOOGLE 来设置显示时间戳，以深化管道。设置显示时间戳指示 SurfaceFlinger 何时渲染图像。如果将来将其设置为几个帧，则将按设置的帧数深化管道。上面例子

中的 Android UI 将当前时间设置为[1] $frameTimeNanos + 2 * vsyncPeriod$[2]

使用 setFrameRate API 告诉平台你的渲染意图。平台将通过选择适当的刷新率来匹配不同的请求。

仅在必要时使用 preferredDisplayModeId，或者当 setFrameRate API 不可用或需要使用非常特定的模式时。

最后，熟悉Android Frame Pacing 库。这个库为你的游戏处理适当的帧间隔，并使用上面描述的方法来处理多个刷新率。

[1] frameTimeNanos received from Choreographer

[2] vsyncPeriod received from Display.getRefreshRate()