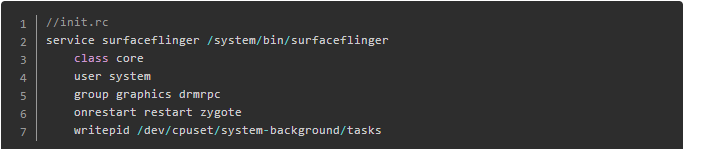
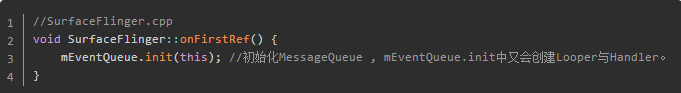
**SurfaceFlinger启动及图层合成送显过程**

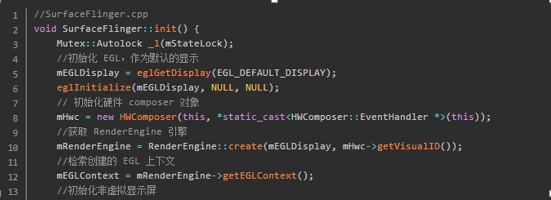
1. SurfaceFlinger启动流程

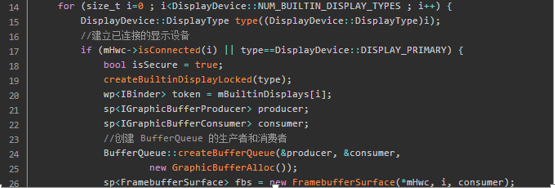
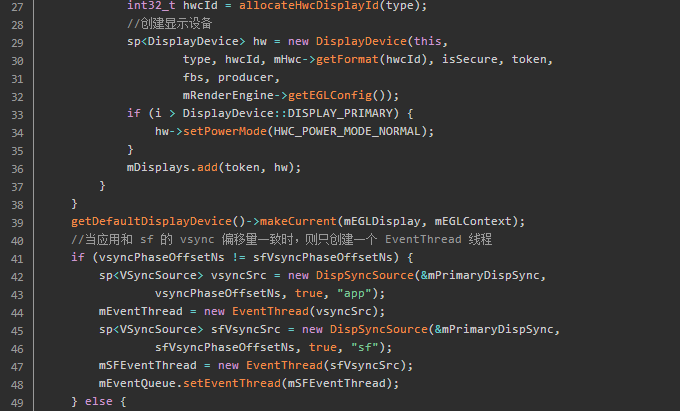
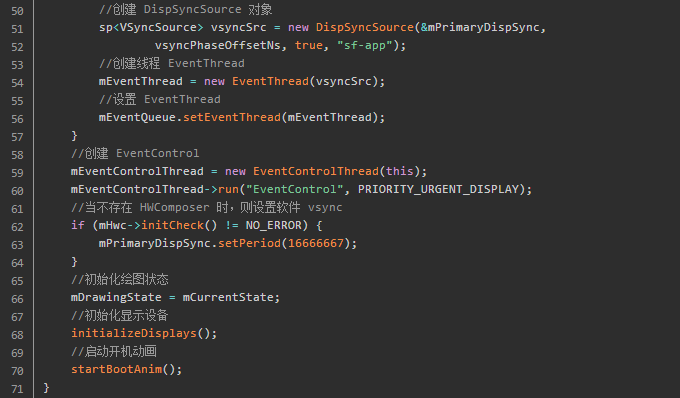
SurfaceFlinger 进程是由 init 进程创建的，运行在独立进程中。

SurfaceFlinger 的创建会执行 main() 方法：

事实上到这里，SurfaceFlinger进程就已经启动了。之后我们再来了解下SurfaceFlinger的一些初始化操作：

首先SurfaceFlinger 的实例化会执行到：onFirstRef()

然后会执行到 SurfaceFlinger::init()：



该方法主要是执行一些初始化工作，包括：EGL标准下OpenGL环境的创建、创建 HWComposer、初始化非虚拟显示屏、启动 EventThread 线程、启动开机动画等等。 在这里，有几个比较重要的对象需要介绍下：

EGL：OpenGL是一个操作GPU的API，它通过驱动向GPU发送相关指令，控制图形渲染管线状态机的运行状态。但OpenGL需要本地视窗系统进行交互，这就需要一个中间控制层，最好与平台无关。EGL正是这个中间控制层，它作为OpenGL ES和本地窗口的桥梁，主要作用是：其主要作用是为OpenGL指令创建 Context 、绘制目标Surface 、配置Framebuffer属性、Swap提交绘制结果等。

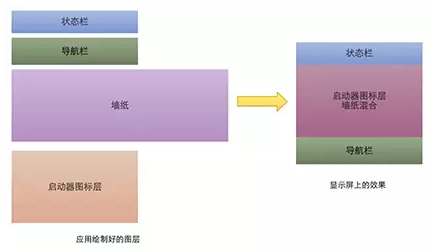
HWComposer：硬件组合抽象层，介于SurfaceFlinger和HAL之间，具体到代码级别就是一个类，封装对于Hwcomposer HAL和Gralloc HAL的使用。 主要作用是：一方面处理部分SurfaceFlinger委托过来的合成工作，另一方面就是产生vsync信号 。

EventThread： 它是Surfaceflinger中的一个线程 ，主要作用：接收VSync事件通知，并分发VSync通知给系统中的每一个感兴趣的注册者。

1. SurfaceFlinger图层合成过程

2.1 什么是图层合成 图层合成就是把多个图层按既定的显示区域，展现到显示屏上。

例如Android手机的Launcher主界面图层合成如下：



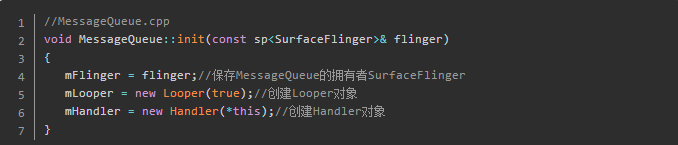
2.2 SurfaceFlinger合成消息

SurfaceFlinger合成图层依赖于Android的异步消息处理机制。

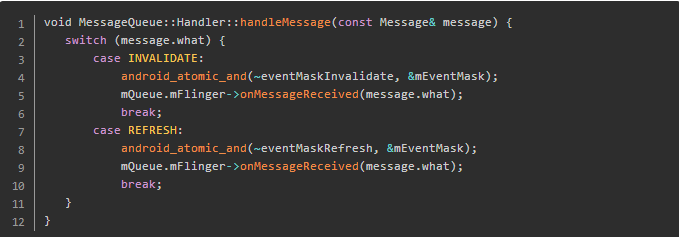
首先，它包含了一个MessageQueue对象（消息队列）用于处理各种异步消息，在onFirstRef()中对消息队列进行了初始化:



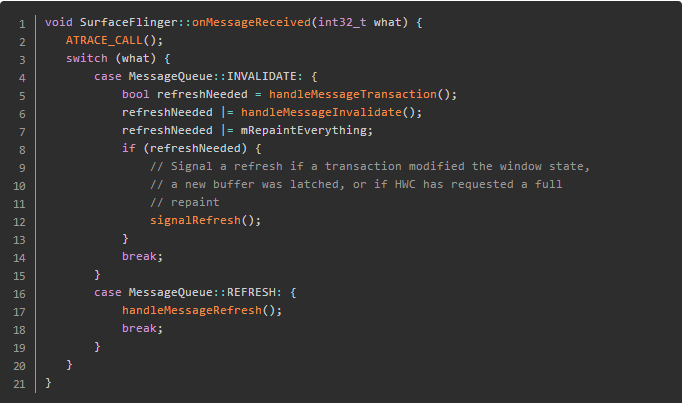
分析一下MessageQueue的实现：



调用Handler::handleMessage()处理INVALIDATE和REFRESH消息，并将其转发给SurfaceFlinger进行处理，调用onMessageReceived():



接下来看一下SurfaceFlinger对消息的处理：



2.3 handleMessageTransaction() 处理之前对屏幕和应用程序窗口的改动。因这些改动很有可能会改变图层的可见区域，进而影响脏区域的计算。

主要处理以下几个方面的transaction：

1）Layer属性的变化；

2）Layer的增加和减少；

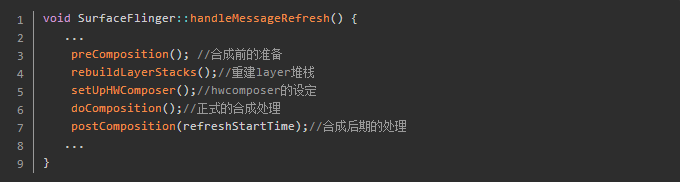
3）Display属性的变化;

4）Display的增加和减少。

2.4 handleMessageInvalidate()

主要调用handlePageFlip()函数。这里Page Flip是指从BufferQueue中取下一个图形缓冲区内容，就好像是“翻页”一样。该函数主要是从各Layer对应的BufferQueue中拿图形缓冲区数据，并根据内容更新脏区域（注：脏区域是需要重绘的屏幕区域。）。并且把GraphicBuffer映射为OpenGL的texture 。 2.5 handleMessageRefresh()

合并和渲染输出。



2.5.1 void SurfaceFlinger::preComposition()

合成前准备工作。首先得到当前所有layer的集合，然后对所有的Layer调用其onPreComposition()检查是否需要ExtralInvalidate,如果需要就调用一次signalLayerUpdate(),即通过EventThread安排一次vsync。

2.5.2 void SurfaceFlinger::rebuildLayerStacks()

计算可见layer及它们的可见区域。首先每个layer都有一个layerStack来区别它属于哪个Display，系统的Display可能不止一个，所以需要逐个处理Display，根据所有layers的当前状态通过SurfaceFlinger::computeVisibleRegions方法计算各个Layer在当前display上的可见区域和脏区域等。最后把需要绘制的layer添加到layersSortedByZ中。

2.5.3 void SurfaceFlinger::setUpHWComposer()

为合成搭建环境。这个HWComposer并不是真正的Hal模块，而是surfaceflinger为管理HWComposer模块而设计的一个类，路径是：frameworks/native/service/surfaceflinger/displayhardware/。依次处理各个Display，构造WorkList，合成过程既可以有Hwc模块完成，也可以通过OpengGlEs来完成，具体用哪种方式是有prepare()中的compositionType来决定的。

2.5.4 void SurfaceFlinger::doComposition()

执行合成操作。执行openGl合成 or HWC合成。

2.5.5 void SurfaceFlinger::postComposition(refreshStartTime)

将图像传递到物理屏幕。

1. 总结

整个surfaceflinger显示总结。

