**DisplayDevice**

1. **DisplayDevice的初始化：**

mEGLDisplay = eglGetDisplay(EGL\_DEFAULT\_DISPLAY);

eglInitialize(mEGLDisplay, NULL, NULL);

mRenderEngine = RenderEngine::create(mEGLDisplay, mHwc->getVisualID());

mEGLContext = mRenderEngine->getEGLContext();

for (size\_t i=0 ; i<DisplayDevice::NUM\_BUILTIN\_DISPLAY\_TYPES ; i++) {

DisplayDevice::DisplayType type((DisplayDevice::DisplayType)i);

if (mHwc->isConnected(i) || type==DisplayDevice::DISPLAY\_PRIMARY) {

bool isSecure = true;

createBuiltinDisplayLocked(type);

wp<IBinder> token = mBuiltinDisplays[i];

sp<IGraphicBufferProducer> producer;

sp<IGraphicBufferConsumer> consumer;

BufferQueue::createBufferQueue(&producer, &consumer,

new GraphicBufferAlloc());

sp<FramebufferSurface> fbs = new FramebufferSurface(\*mHwc, i,

consumer);

int32\_t hwcId = allocateHwcDisplayId(type);

sp<DisplayDevice> hw = new DisplayDevice(this,

type, hwcId, mHwc->getFormat(hwcId), isSecure, token,

fbs, producer,

mRenderEngine->getEGLConfig());

if (i > DisplayDevice::DISPLAY\_PRIMARY) {

hw->setPowerMode(HWC\_POWER\_MODE\_NORMAL);

}

mDisplays.add(token, hw);

}

}

getDefaultDisplayDevice()->makeCurrent(mEGLDisplay, mEGLContext);

**1.1 openglEs**

**1.1.1 创建EGLSurface**

**//使用BufferQueueProducer来创建继承于ANativeWindow的surface**

**//所以说surface就是BufferQueue中的生产者**

mNativeWindow = new Surface(producer, false);

ANativeWindow\* const window = mNativeWindow.get();

EGLSurface surface;

EGLDisplay display = eglGetDisplay(EGL\_DEFAULT\_DISPLAY);

if (config == EGL\_NO\_CONFIG) {

config = RenderEngine::chooseEglConfig(display, format);

}

surface = eglCreateWindowSurface(display, config, window, NULL);

eglQuerySurface(display, surface, EGL\_WIDTH, &mDisplayWidth);

eglQuerySurface(display, surface, EGL\_HEIGHT, &mDisplayHeight);

**1.1.2 opengl怎样调用到surface中的dequeueBuffer、queueBuffer等函数。**

**//在surface的构造函数中将ANativeWindow中函数指针绑定到surface中的**

**//静态hook函数中，opengl调用的都是这些函数指针最终调用到surface中的函数。**

Surface::Surface(

const sp<IGraphicBufferProducer>& bufferProducer,

bool controlledByApp)

: mGraphicBufferProducer(bufferProducer),

mGenerationNumber(0)

{

ANativeWindow::setSwapInterval = hook\_setSwapInterval;

ANativeWindow::dequeueBuffer = hook\_dequeueBuffer;

ANativeWindow::cancelBuffer = hook\_cancelBuffer;

ANativeWindow::queueBuffer = hook\_queueBuffer;

ANativeWindow::query = hook\_query;

ANativeWindow::perform = hook\_perform;

ANativeWindow::dequeueBuffer\_DEPRECATED = hook\_dequeueBuffer\_DEPRECATED;

ANativeWindow::cancelBuffer\_DEPRECATED = hook\_cancelBuffer\_DEPRECATED;

ANativeWindow::lockBuffer\_DEPRECATED = hook\_lockBuffer\_DEPRECATED;

ANativeWindow::queueBuffer\_DEPRECATED = hook\_queueBuffer\_DEPRECATED;

。。。。。。

}

**1.1.3 opengl在合成的时候进行queue buffer的示例**

**//eglSwapBuffers就是一次queueBuffer的过程。**

void DisplayDevice::swapBuffers(HWComposer& hwc) const {

if (hwc.initCheck() != NO\_ERROR ||

(hwc.hasGlesComposition(mHwcDisplayId) &&

(hwc.supportsFramebufferTarget() || mType >= DISPLAY\_VIRTUAL))) {

EGLBoolean success = eglSwapBuffers(mDisplay, mSurface);

}

status\_t result = mDisplaySurface->advanceFrame();

}

#0 android::FramebufferSurface::nextBuffer (this=0xb81db058, outBuffer=..., outFence=...)

#1 0xb6f68218 in android::FramebufferSurface::onFrameAvailable (this=0xb81db058)

#2 0xb6cea244 in android::BufferQueue::ProxyConsumerListener::onFrameAvailable (this=<optimized out>, item=...)

#3 0xb6cf026c in android::BufferQueueProducer::queueBuffer (this=<optimized out>, slot=<optimized out>,

#4 0xb6d02682 in android::Surface::queueBuffer (this=0xb81db738, buffer=<optimized out>, fenceFd=<optimized out>)

#5 0xb5fe82ba in ?? () from vendor/lib/egl/eglsubAndroid.so

#6 0xb6679814 in qeglDrvAPI\_eglSwapBuffers () system/vendor/lib/egl/libEGL\_adreno.so

#7 0xb6d53520 in eglSwapBuffersWithDamageKHR (dpy=<optimized out>, draw=0xb81dd750, rects=0x0, n\_rects=0)

#8 0xb6f52374 in android::DisplayDevice::swapBuffers (this=0xb81db520, hwc=...)

#9 0xb6f60222 in android::SurfaceFlinger::doDisplayComposition (this=0xb815fdf0, hw=..., inDirtyRegion=...)

#10 0xb6f5f3aa in android::SurfaceFlinger::doComposition (this=0xb815fdf0)

#11 0xb6f5e6a2 in android::SurfaceFlinger::handleMessageRefresh (this=0xb815fdf0)

#12 0xb6f5e442 in android::SurfaceFlinger::onMessageReceived (this=0xb815fdf0, what=<optimized out>)

#13 0xb6ef0c92 in android::Looper::pollInner (this=this@entry=0xb8161660, timeoutMillis=<optimized out>, timeoutMillis@entry=-1)

#14 0xb6ef0dd2 in android::Looper::pollOnce (this=0xb8161660, timeoutMillis=-1, outFd=0x0, outEvents=0x0, outData=0x0)

#15 0xb6f5b5b0 in pollOnce (this=0xbe91fe50, timeoutMillis=-1) at system/core/include/utils/Looper.h:265

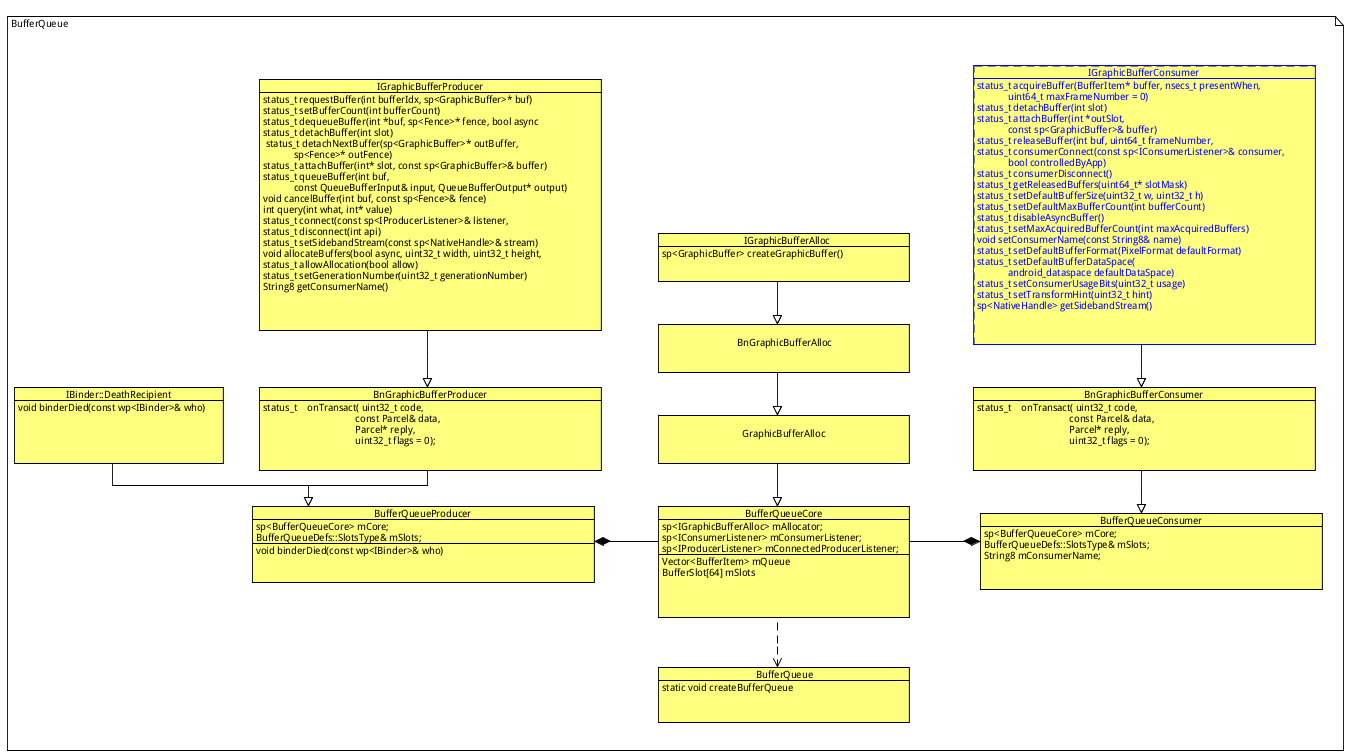
#16 android::MessageQueue::waitMessage (this=0xb815ff9c) at frameworks/native/services/surfaceflinger/MessageQueue.cpp:115

#17 0xb6f5df80 in waitForEvent (this=<optimized out>) at frameworks/native/services/surfaceflinger/SurfaceFlinger.cpp:769

#18 android::SurfaceFlinger::run (this=<optimized out>) at frameworks/native/services/surfaceflinger/SurfaceFlinger.cpp:801

#19 0xb6fd7efc in main () at frameworks/native/services/surfaceflinger/main\_surfaceflinger.cpp:52

**1.2 BufferQueue**

****

**1.2.1 create producer and consumer:**

sp<IGraphicBufferProducer> producer;

sp<IGraphicBufferConsumer> consumer;

BufferQueue::createBufferQueue(&producer, &consumer,

new GraphicBufferAlloc());

**1.2.2 create consumer listener and producer listener for BufferQueueCore**

ConsumerBase::ConsumerBase(const sp<IGraphicBufferConsumer>& bufferQueue, bool controlledByApp) :mAbandoned(false),mConsumer(bufferQueue) {

mName = String8::format("unnamed-%d-%d", getpid(), createProcessUniqueId());

wp<ConsumerListener> listener = static\_cast<ConsumerListener\*>(this);

sp<IConsumerListener> proxy = new BufferQueue::ProxyConsumerListener(listener);

status\_t err = mConsumer->consumerConnect(proxy, controlledByApp);

if (err != NO\_ERROR) {

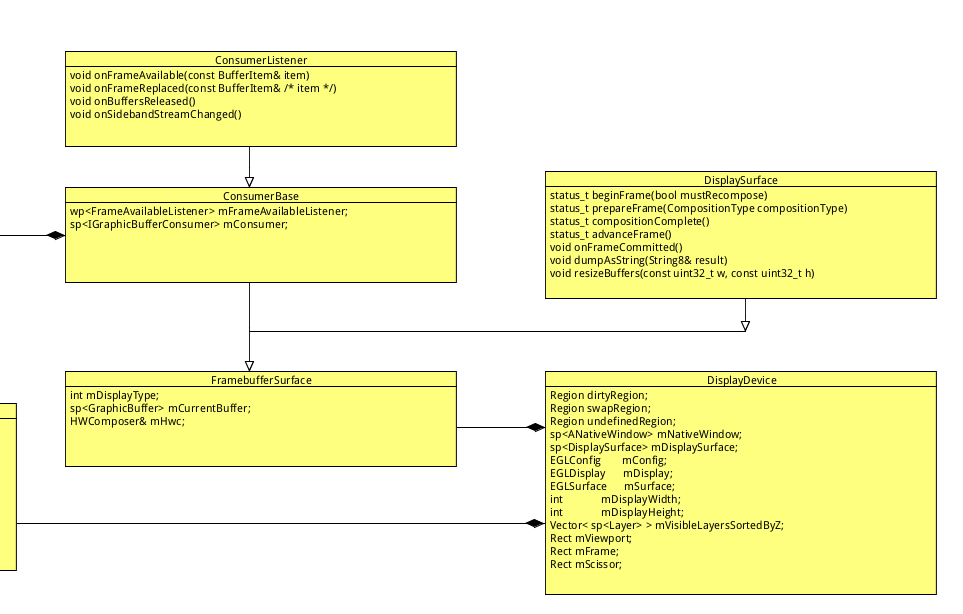
} else {

mConsumer->setConsumerName(mName);

}

}

**1.3 FramebufferSurface**

****

FramebufferSurface 将注册到BufferQueueCore中作为mConsumerListener，当BufferQueueProducer进行queueBuffer的时候调用FramebufferSurface 的onFrameAvailable函数。

DisplayDevice是包含了生产者，生产者为FramebufferSurface中的与BufferQueueConsumer公用一个bufferQueueCore的BufferQueueProducer，DisplayDevice使用这个BQP来创建继承于ANativeWindow的surface。