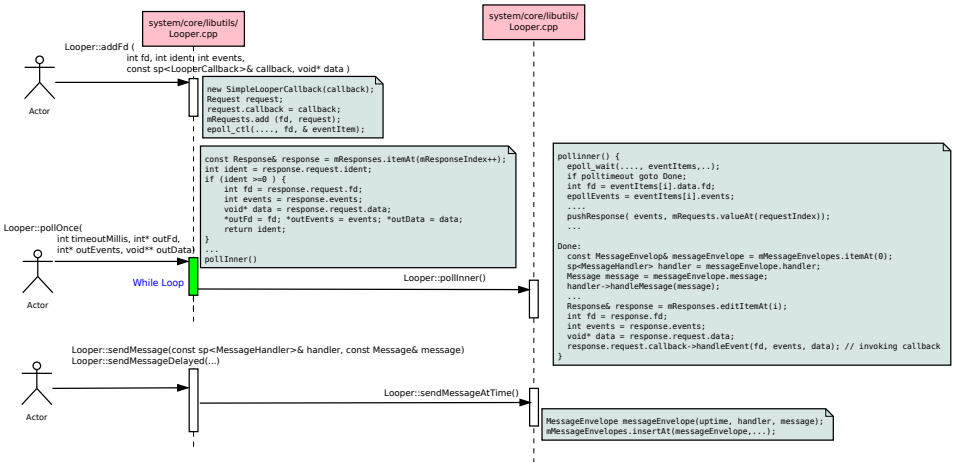
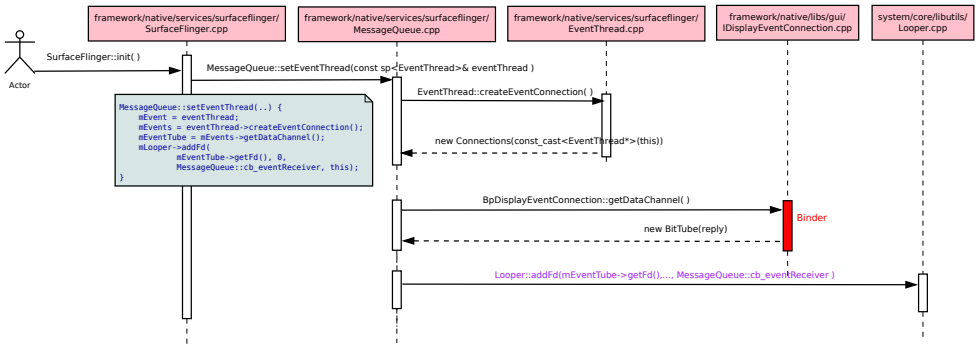
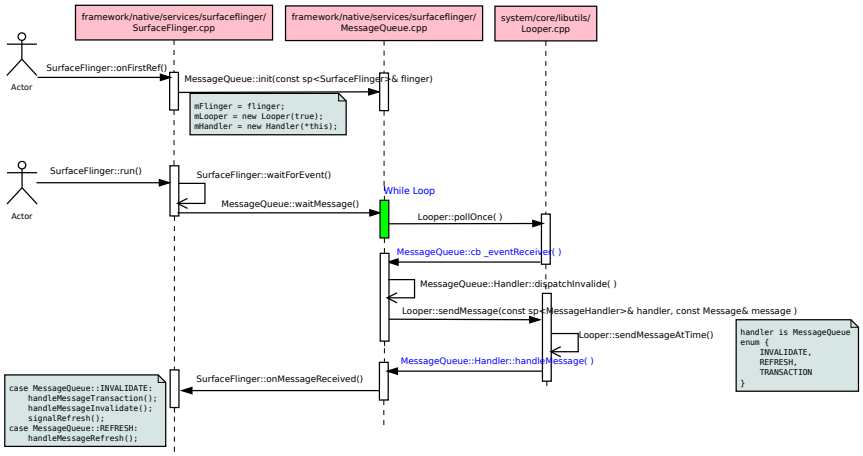
Looper's mechanism



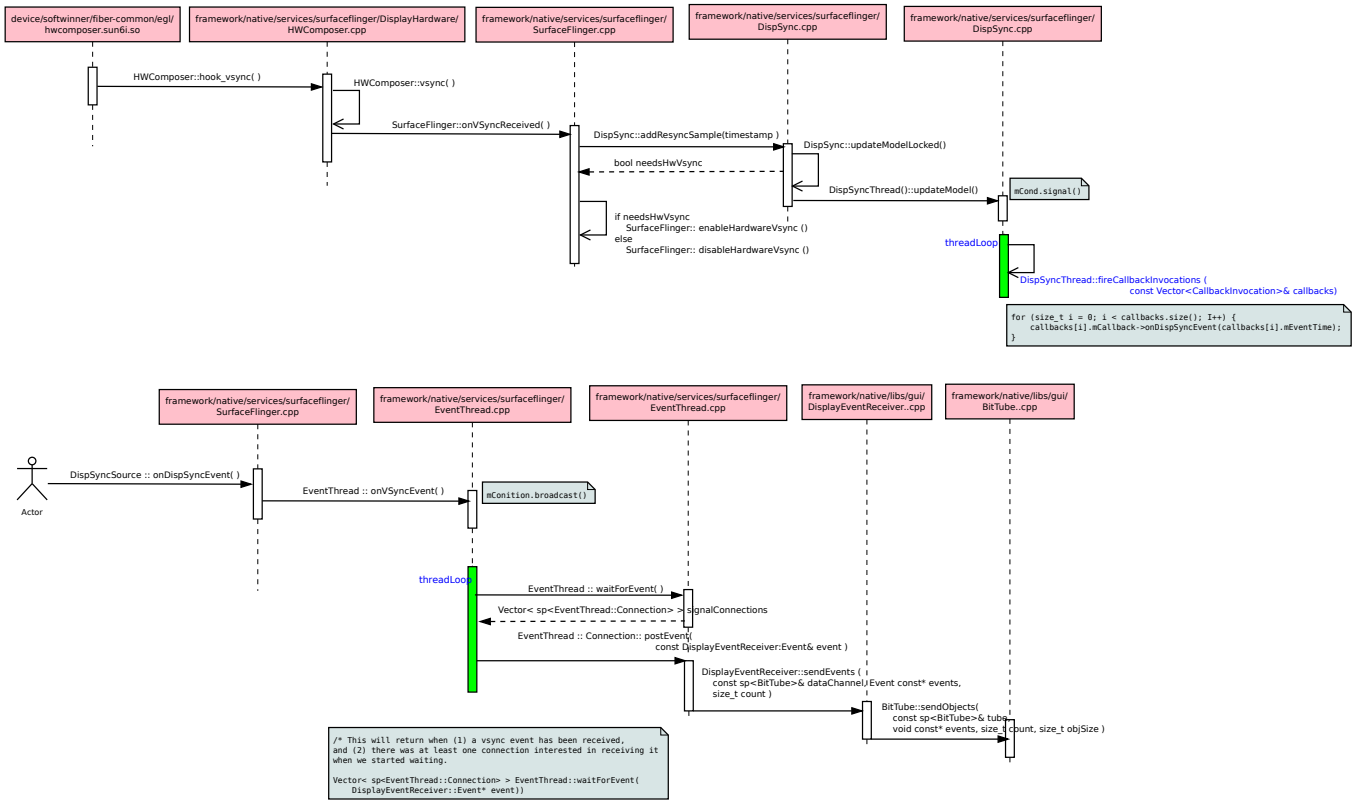
SurfaceFlinger setEventThread



SurfaceFlinger onMessageReceived



Vsync Flow


device/softwinner/fiber-common/egl/hwcomposer.sun6i.so
framework/native/services/surfaceflinger/DisplayHardware/HWComposer.cpp
framework/native/services/surfaceflinger/SurfaceFlinger.cpp
framework/native/services/surfaceflinger/DispSync.cpp
framework/native/services/surfaceflinger/DispSync.cpp
HWComposer::hook_vsync( )
HWComposer::vsync( )
SurfaceFlinger::onVSyncReceived( )
DispSync::addResyncSample(timestamp)
DispSync::updateModelLocked()
bool needsHwVsync
DispSyncThread()::updateModel()
mCond.signal()
if needsHwVsync
SurfaceFlinger:: enableHardwareVsync ()
else
SurfaceFlinger:: disableHardwareVsync ()
threadLoop
DispSyncThread::fireCallbackInvocations (
const Vector<CallbackInvocation>& callbacks)
for (size_t i = 0; i < callbacks.size(); I++) {
callbacks[i].mCallback->onDispSyncEvent(callbacks[i].mEventTime);
}
framework/native/services/surfaceflinger/SurfaceFlinger.cpp
framework/native/services/surfaceflinger/EventThread.cpp
framework/native/services/surfaceflinger/EventThread.cpp
framework/native/libs/gui/DisplayEventReceiver..cpp
framework/native/libs/gui/BitTube..cpp
Actor
DispSyncSource :: onDispSyncEvent( )
EventThread :: onVSyncEvent( )
mConition.broadcast()
threadLoop
EventThread :: waitForEvent( )
Vector< sp<EventThread::Connection> > signalConnections
EventThread :: Connection:: postEvent(
const DisplayEventReceiver:Event& event)
DisplayEventReceiver::sendEvents (
const sp<BitTube>& dataChannel, Event const* events,
size_t count)
BitTube::sendObjects(
const sp<BitTube>& tube,
void const* events, size_t count, size_t objSize)
/* This will return when (1) a vsync event has been received,
and (2) there was at least one connection interested in receiving it
when we started waiting.
Vector< sp<EventThread::Connection> > EventThread::waitForEvent(
DisplayEventReceiver::Event* event))