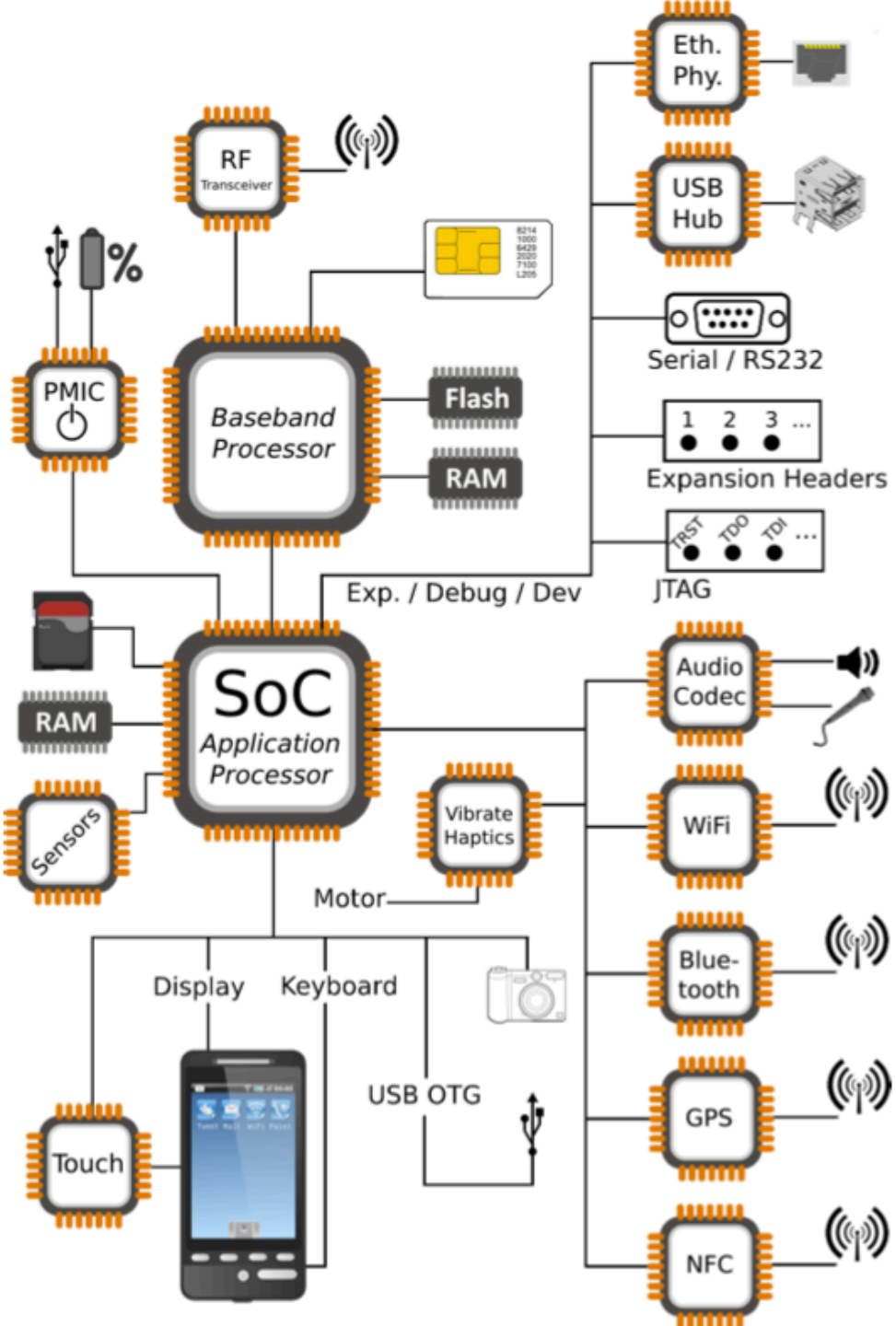
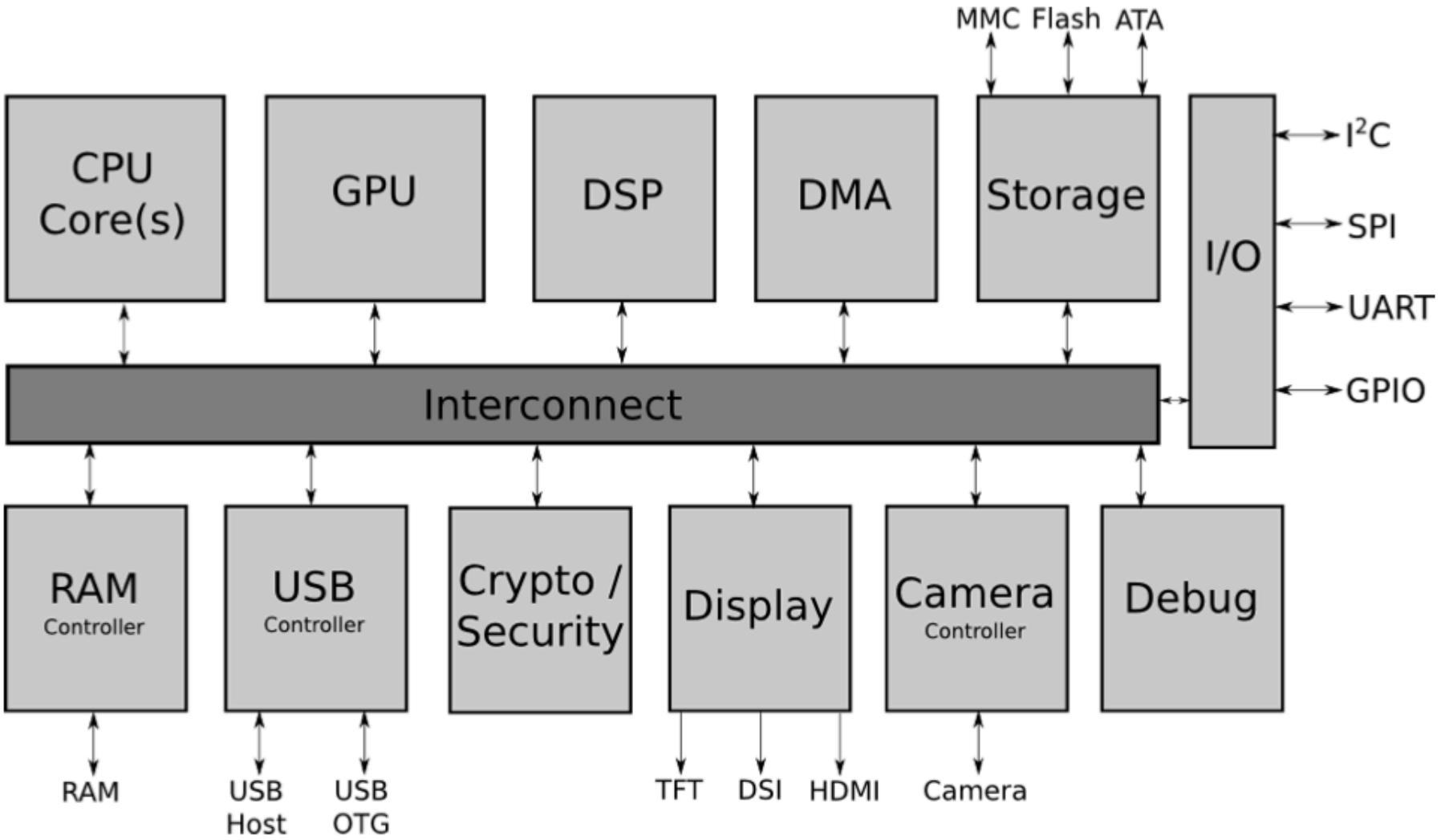
Android GUI Architecture Summary

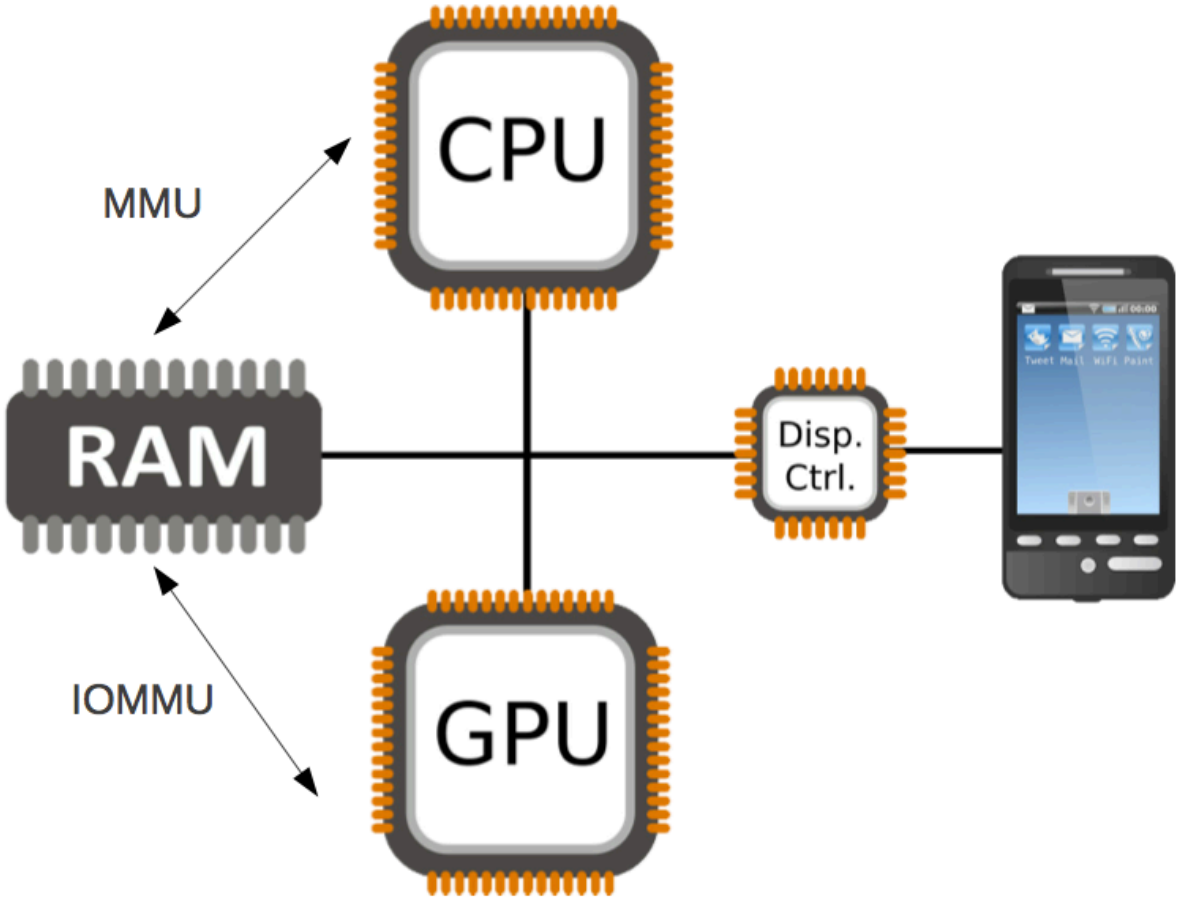
David Lau • China

Email: zhiyong.liu@aliyun.com Latest Update：2013-09-08

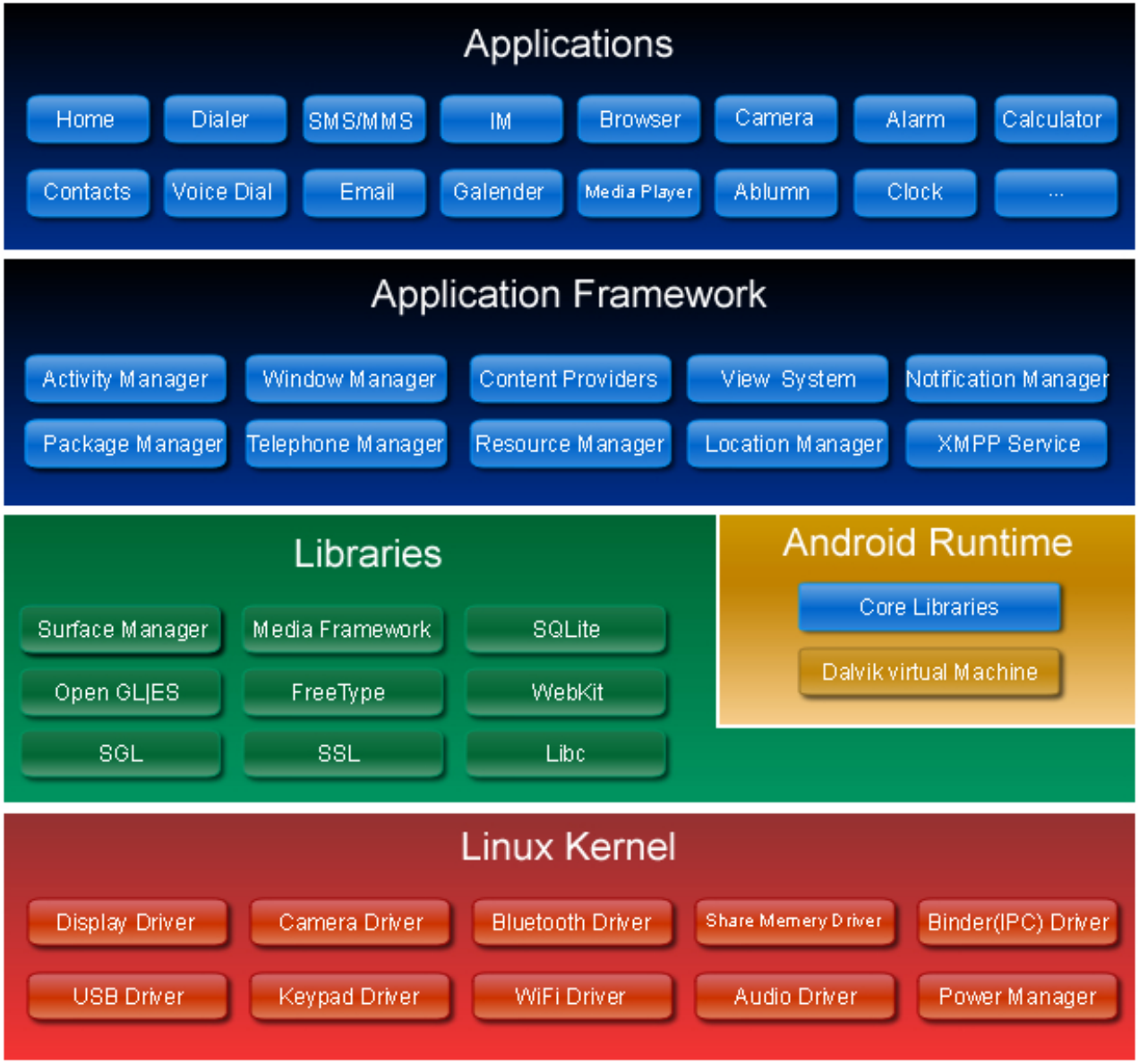
Smart Phone’s Hardware

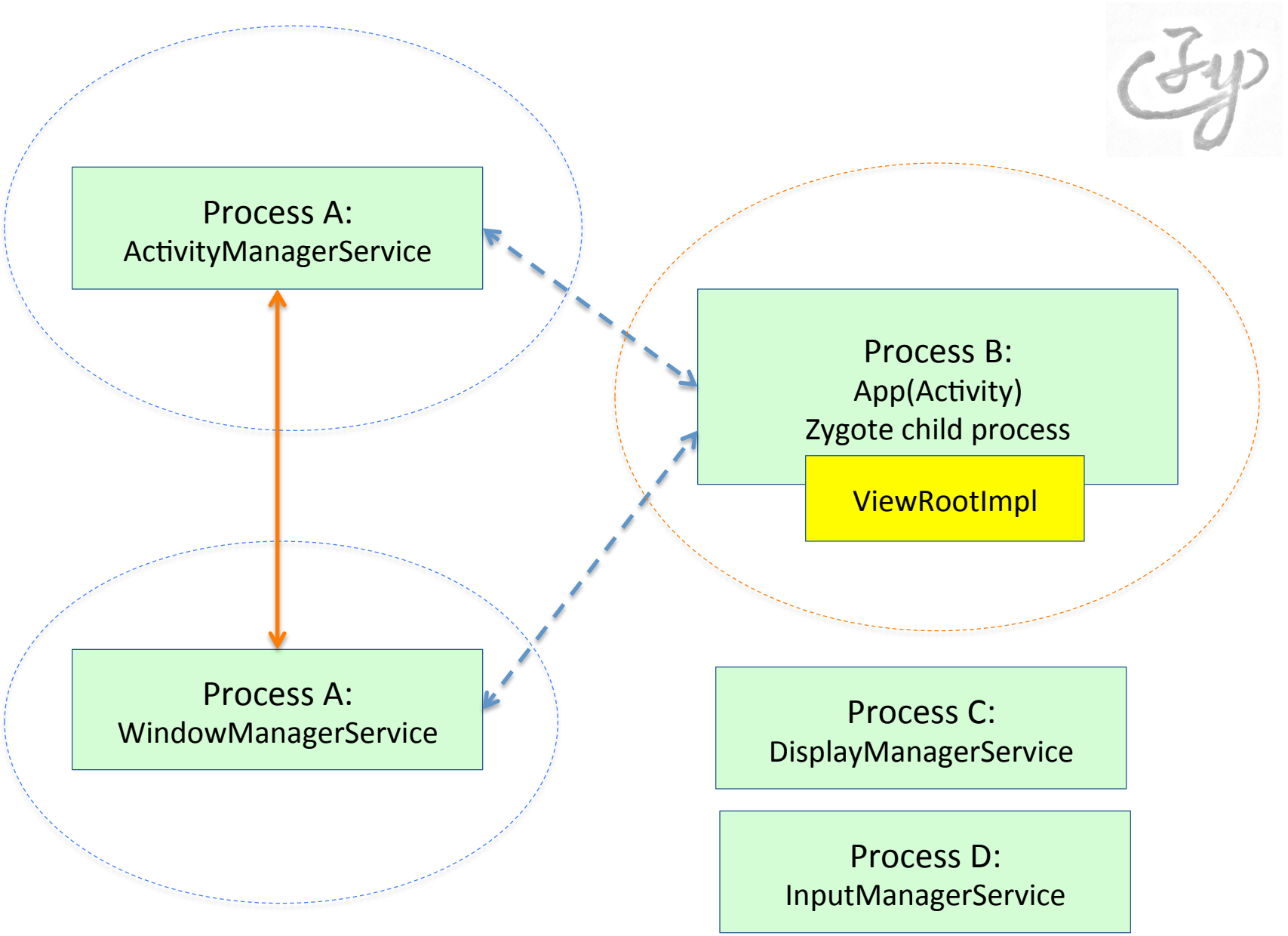

Eth.
Phy.
RF
Transceiver
USB
Hub
PMIC
Baseband
Processor
Flash
RAM
Serial / RS232
1 2 3 ...
Expansion Headers
TRST TDO TDI ...
JTAG
Exp. / Debug / Dev
SoC
Application
Processor
RAM
Sensors
Audio
Codec
Vibrate
Haptics
WiFi
Motor
Blue-
tooth
Display
Keyboard
Touch
USB OTG
GPS
NFC


MMC Flash ATA
CPU Core(s)
GPU
DSP
DMA
Storage
I/O
I²C
SPI
UART
GPIO
Interconnect
RAM Controller
USB Controller
Crypto / Security
Display
Camera Controller
Debug
RAM
USB Host
USB OTG
TFT
DSI
HDMI
Camera


CPU
MMU
RAM
Disp.
Ctrl.
IOMMU
GPU

Smart Phone's System
—— Android Framework


Applications
Home
Dialer
SMS/MMS
IM
Browser
Camera
Alarm
Calculator
Contacts
Voice Dial
Email
Galender
Media Player
Ablumn
Clock
...
Application Framework
Activity Manager
Window Manager
Content Providers
View System
Notification Manager
Package Manager
Telephone Manager
Resource Manager
Location Manager
XMPP Service
Libraries
Surface Manager
Media Framework
SQLite
Open GL|ES
FreeType
WebKit
SGL
SSL
Libc
Android Runtime
Core Libraries
Dalvik virtual Machine
Linux Kernel
Display Driver
Camera Driver
Bluetooth Driver
Share Memery Driver
Binder(IPC) Driver
USB Driver
Keypad Driver
WiFi Driver
Audio Driver
Power Manager


Process A:
ActivityManagerService
Process B:
App(Activity)
Zygote child process
ViewRootImpl
Process A:
WindowManagerService
Process C:
DisplayManagerService
Process D:
InputManagerService

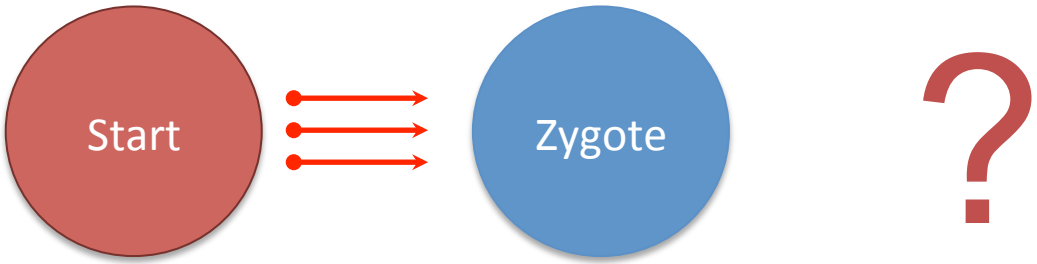
GUI's Actions

- Lauch a new Activity
- Create a new window for the activity and register it at the wms
- Create a new surface for the Activity's window and hand it over to the Activity.
- Draw the activity's view hierarchy into the offscreen surface(layer)
- Composite all visible offscreen surfaces onto the display
- AMS start the Activity and attach the activity, create a PhoneWindow
- Add the window to WMS, and then do performTraversals
- The WMS lock the surface and create a native surface
- Surface unlocks the Canvas and post
- At the SurfaceFlinger, compose all surfaces
- ViewRootImpl finish Drawing

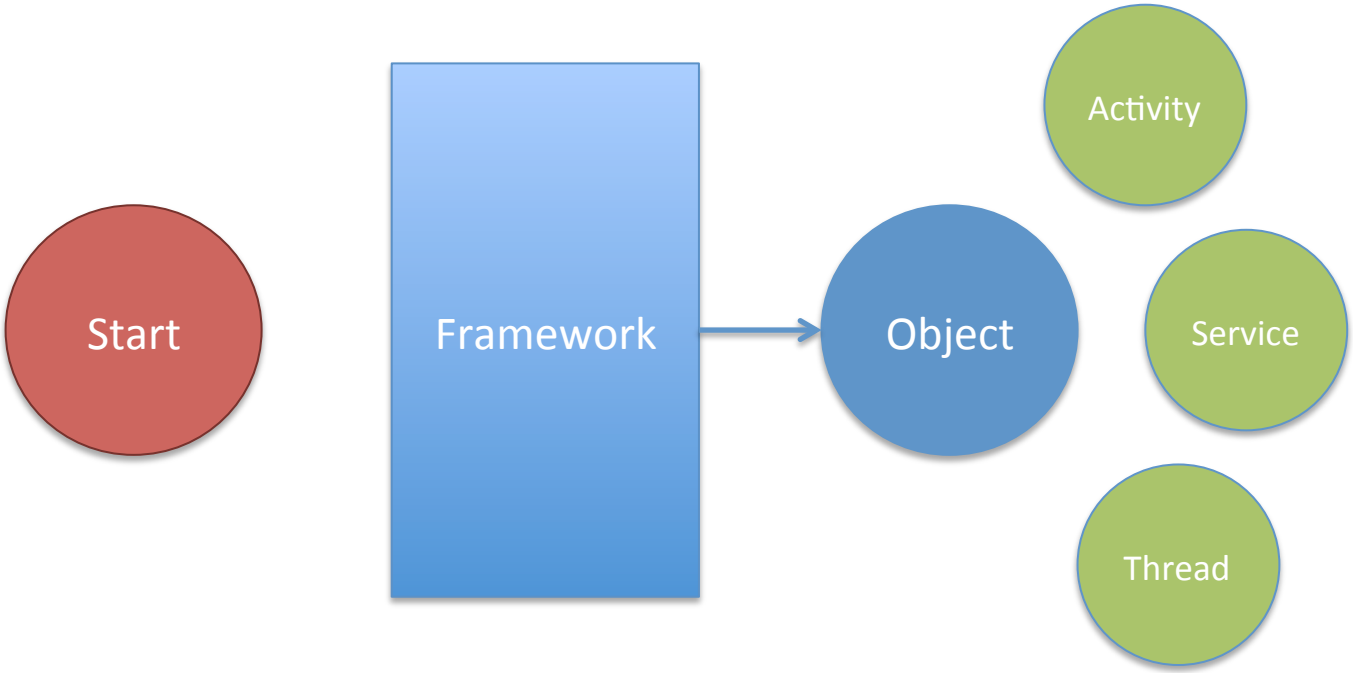
```
Slog.i(TAG, "Input Manager");
inputManager = new InputManagerService(context, wmHandler);

Slog.i(TAG, "Window Manager");
wm = WindowManagerService.main(context, power, display, inputManager,
        uiHandler, wmHandler,
        factoryTest != SystemServer.FACTORY_TEST_LOW_LEVEL,
        !firstBoot, onlyCore);
ServiceManager.addService(Context.WINDOW_SERVICE, wm);
ServiceManager.addService(Context.INPUT_SERVICE, inputManager);

ActivityManagerService.self().setWindowManager(wm);

inputManager.setWindowManagerCallbacks(wm.getInputMonitor());
inputManager.start();

display.setWindowManager(wm);
display.setInputManager(inputManager);
```

frameworks/base/services/java/com/android/server/SystemServer.java

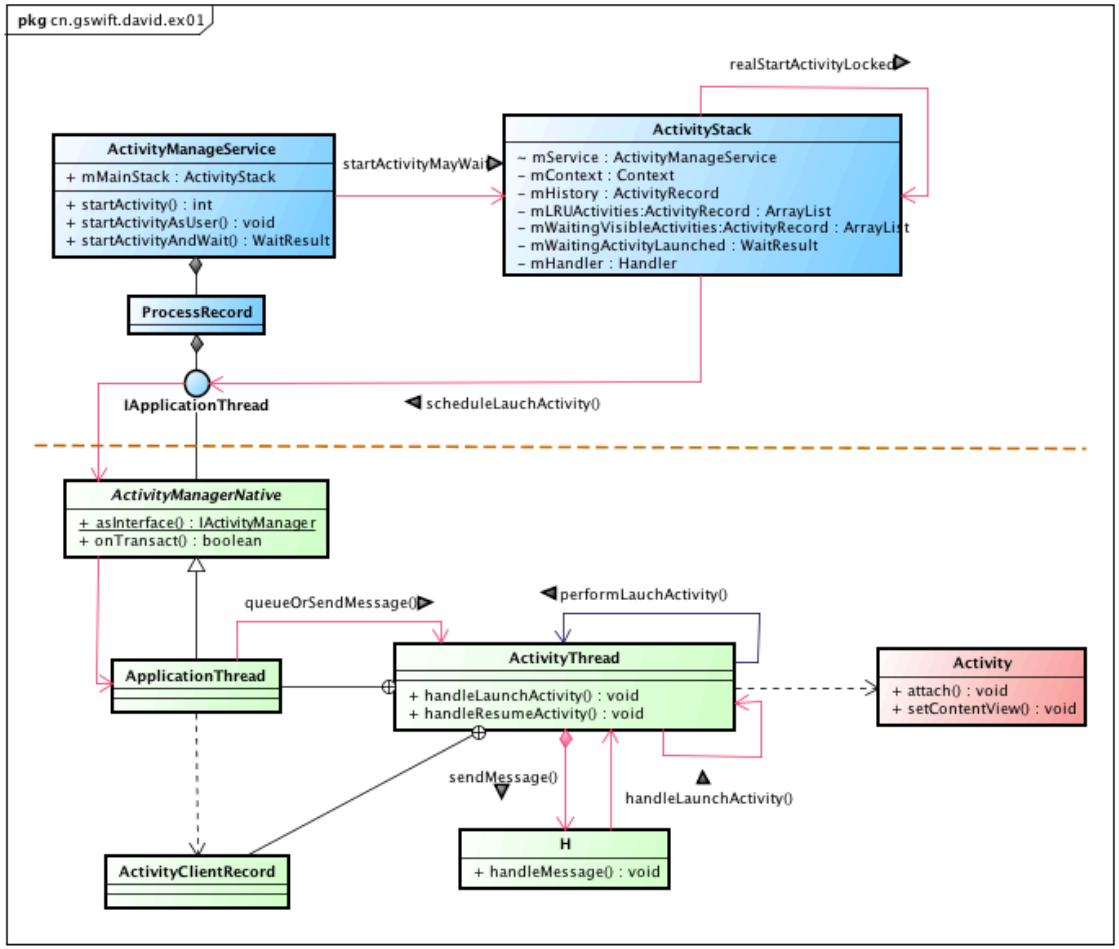

Start


Start
Zygote
?

Architecture Key Point



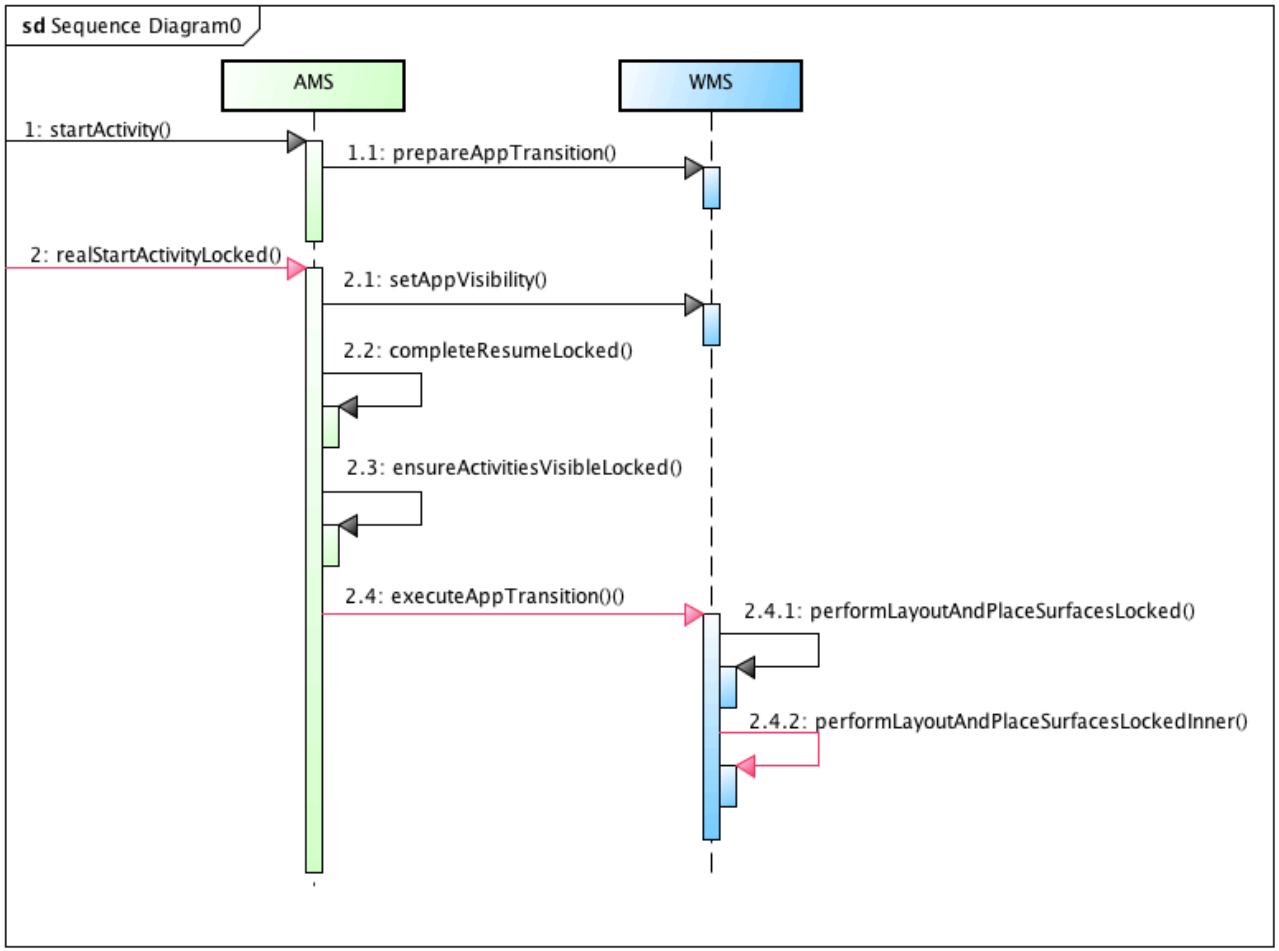
Launch Activity



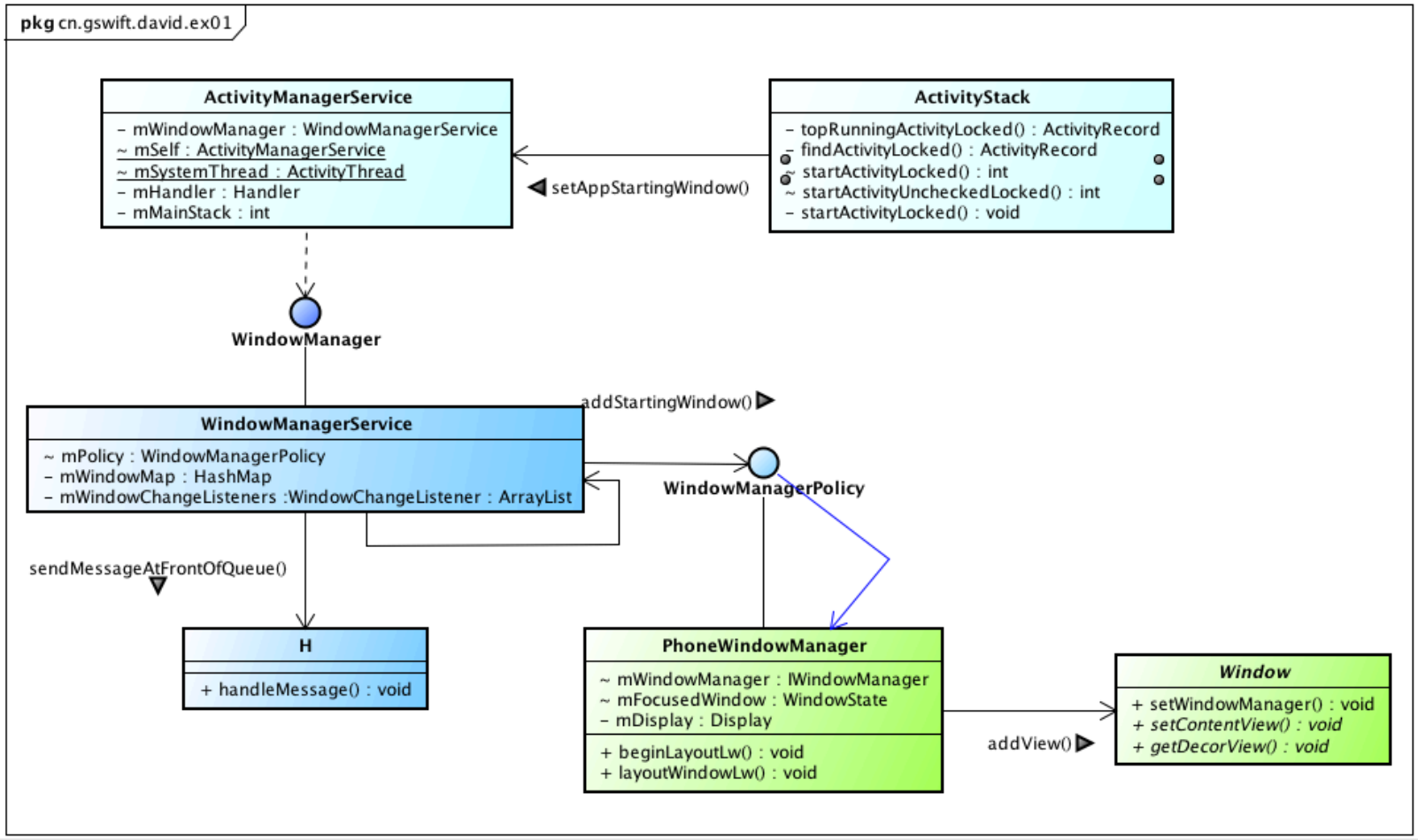
```
private Activity performLaunchActivity(ActivityClientRecord r, Intent customIntent) {
    // System.out.println("##### [" + System.currentTimeMillis() + "] ActivityThread.performLaunchActivity(" + r + ")");

    ActivityInfo aInfo = r.activityInfo;
    if (r.packageInfo == null) {
        r.packageInfo = getPackageInfo(aInfo.applicationInfo, r.compatInfo,
                Context.CONTEXT_INCLUDE_CODE);
    }

    ComponentName component = r.intent.getComponent();
    if (component == null) {
        component = r.intent.resolveActivity(
            mInitialApplication.getPackageManager());
        r.intent.setComponent(component);
    }

    if (r.activityInfo.targetActivity != null) {
        component = new ComponentName(r.activityInfo.packageName,
                r.activityInfo.targetActivity);
    }

    Activity activity = null;
    try {
        java.lang.ClassLoader cl = r.packageInfo.getClassLoader();
        activity = mInstrumentation.newActivity(
                cl, component.getClassName(), r.intent);
        StrictMode.incrementExpectedActivityCount(activity.getClass());
        r.intent.setExtrasClassLoader(cl);
        if (r.state != null) {
            r.state.setClassLoader(cl);
        }
    } catch (Exception e) {
        if (!mInstrumentation.onException(activity, e)) {
            throw new RuntimeException(
                "Unable to instantiate activity " + component
                + ": " + e.toString(), e);
        }
    }
```

Activity Instance

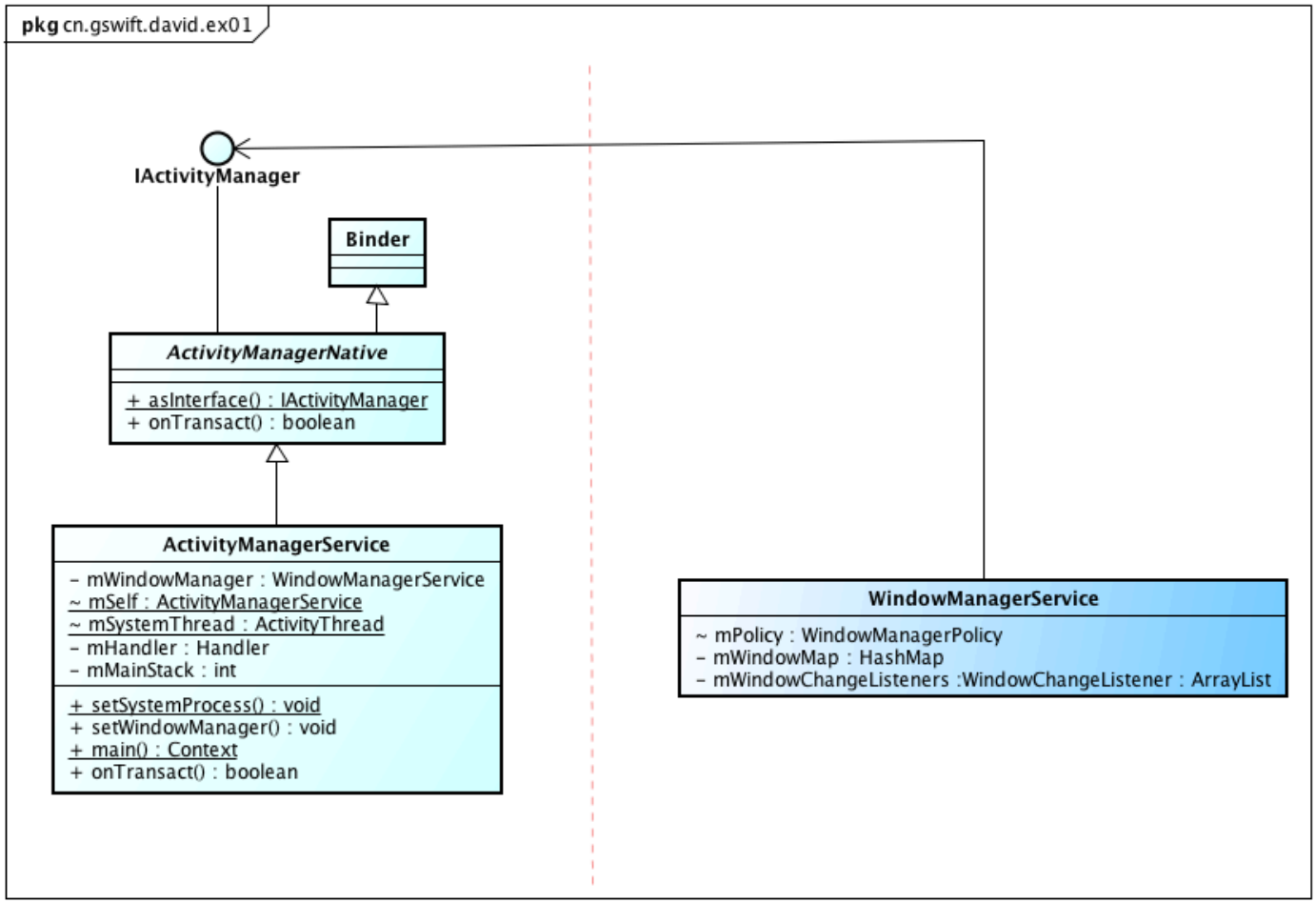
AMS-WMS


sd Sequence Diagram0
AMS
WMS
1: startActivity()
1.1: prepareAppTransition()
2: realStartActivityLocked()
2.1: setAppVisibility()
2.2: completeResumeLocked()
2.3: ensureActivitiesVisibleLocked()
2.4: executeAppTransition()()
2.4.1: performLayoutAndPlaceSurfacesLocked()
2.4.2: performLayoutAndPlaceSurfacesLockedInner()

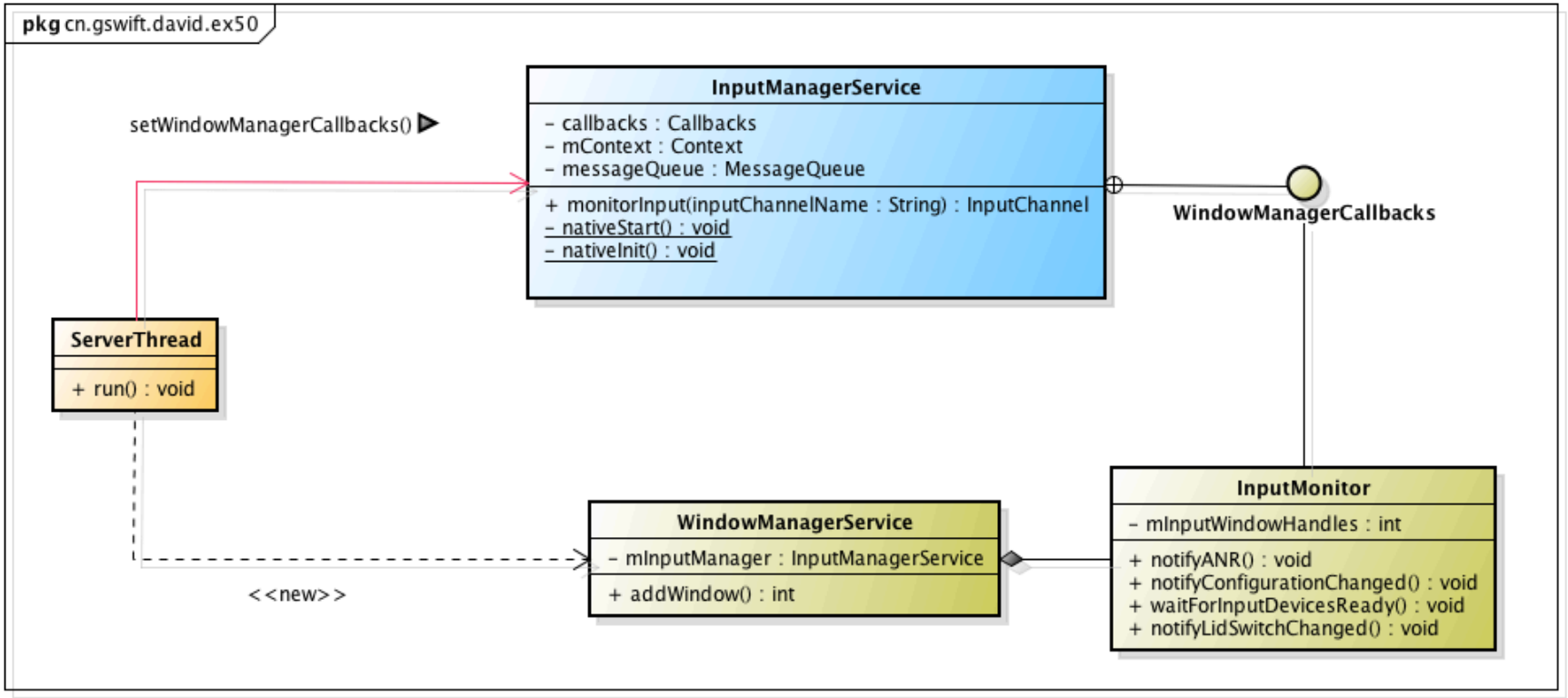
AMS-WMS


pkg cn.gswift.david.ex01
ActivityManagerService
- mWindowManager : WindowManagerService
~ mSelf : ActivityManagerService
~ mSystemThread : ActivityThread
- mHandler : Handler
- mMainStack : int
setAppStartingWindow()
ActivityStack
- topRunningActivityLocked() : ActivityRecord
- findActivityLocked() : ActivityRecord
- startActivityLocked() : int
~ startActivityUncheckedLocked() : int
- startActivityLocked() : void
WindowManager
addStartingWindow()
WindowManagerService
~ mPolicy : WindowManagerPolicy
- mWindowMap : HashMap
- mWindowChangeListeners :WindowChangeListener : ArrayList
WindowManagerPolicy
sendMessageAtFrontOfQueue()
H
+ handleMessage() : void
PhoneWindowManager
~ mWindowManager : IWindowManager
~ mFocusedWindow : WindowState
- mDisplay : Display
+ beginLayoutLw() : void
+ layoutWindowLw() : void
addView()
Window
+ setWindowManager() : void
+ setContentView() : void
+ getDecorView() : void

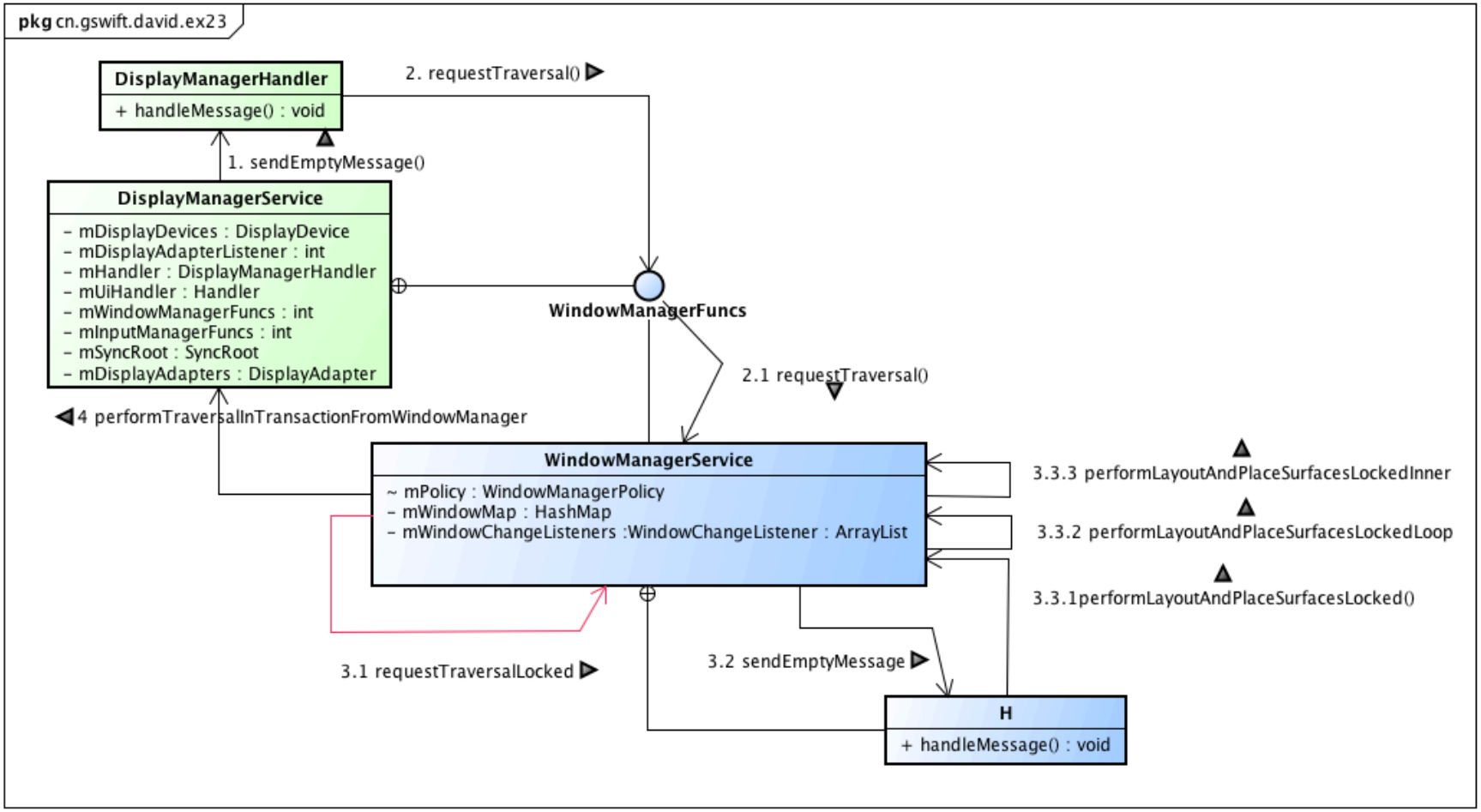
WMS-AMS


pkg cn.gswift.david.ex01
IActivityManager
Binder
ActivityManagerNative
+ asInterface() : IActivityManager
+ onTransact() : boolean
ActivityManagerService
- mWindowManager : WindowManagerService
~ mSelf : ActivityManagerService
~ mSystemThread : ActivityThread
- mHandler : Handler
- mMainStack : int
+ setSystemProcess() : void
+ setWindowManager() : void
+ main() : Context
+ onTransact() : boolean
WindowManagerService
~ mPolicy : WindowManagerPolicy
- mWindowMap : HashMap
- mWindowChangeListeners :WindowChangeListener : ArrayList

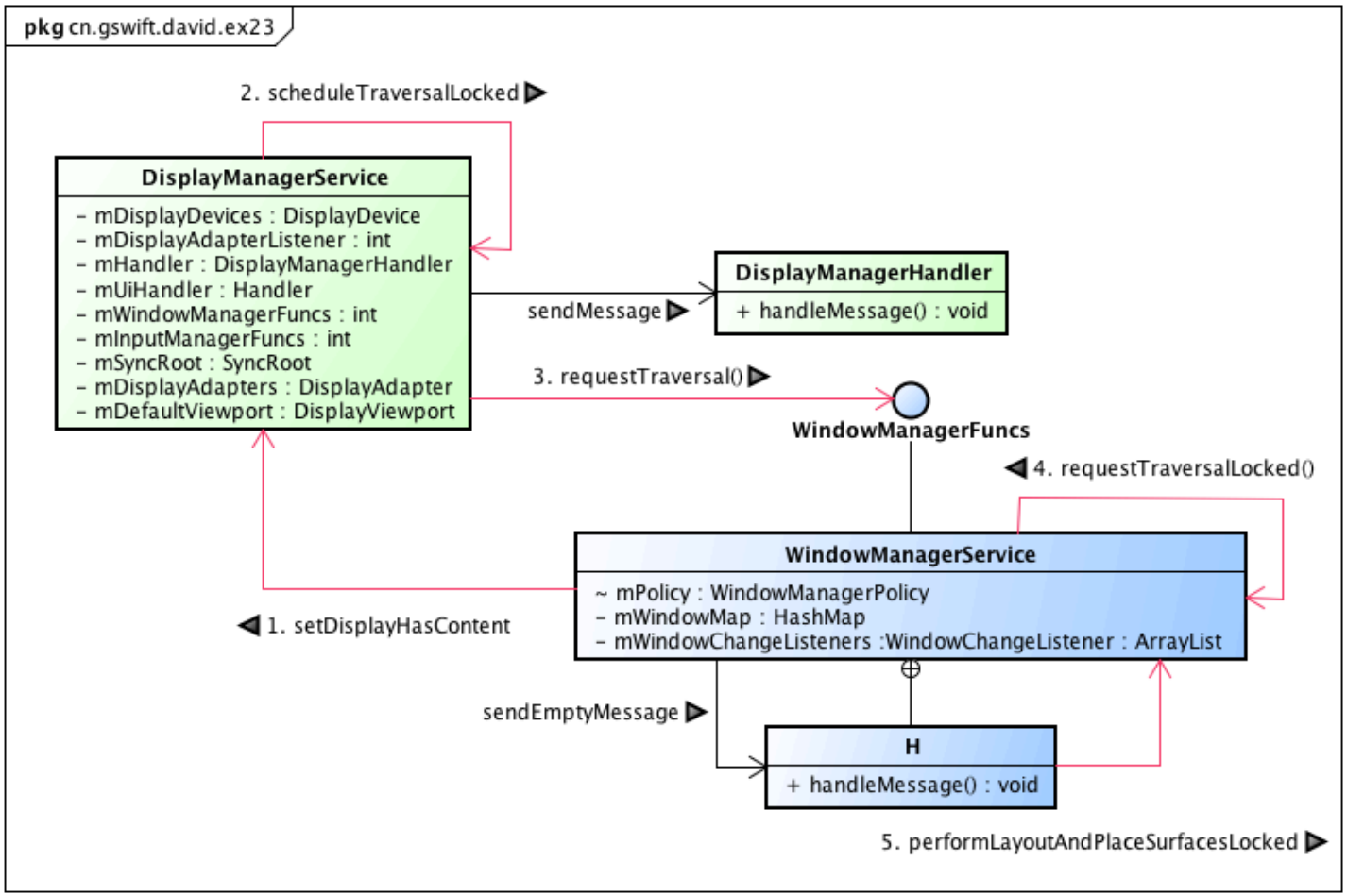
WMS &IMS


pkg cn.gswift.david.ex50
setWindowManagerCallbacks()
InputManagerService
- callbacks : Callbacks
- mContext : Context
- messageQueue : MessageQueue
+ monitorInput(inputChannelName : String) : InputChannel
- nativeStart() : void
- nativeInit() : void
WindowManagerCallbacks
ServerThread
+ run() : void
<<new>>
WindowManagerService
- mInputManager : InputManagerService
+ addWindow() : int
InputMonitor
- mInputWindowHandles : int
+ notifyANR() : void
+ notifyConfigurationChanged() : void
+ waitForInputDevicesReady() : void
+ notifyLidSwitchChanged() : void

WMS&DMS


pkg cn.gswift.david.ex23
DisplayManagerHandler
+ handleMessage() : void
2. requestTraversal()
1. sendEmptyMessage()
DisplayManagerService
- mDisplayDevices : DisplayDevice
- mDisplayAdapterListener : int
- mHandler : DisplayManagerHandler
- mUiHandler : Handler
- mWindowManagerFuncs : int
- mInputManagerFuncs : int
- mSyncRoot : SyncRoot
- mDisplayAdapters : DisplayAdapter
WindowManagerFuncs
2.1 requestTraversal()
4 performTraversalInTransactionFromWindowManager
WindowManagerService
~ mPolicy : WindowManagerPolicy
- mWindowMap : HashMap
- mWindowChangeListeners :WindowChangeListener : ArrayList
3.3.3 performLayoutAndPlaceSurfacesLockedInner
3.3.2 performLayoutAndPlaceSurfacesLockedLoop
3.3.1performLayoutAndPlaceSurfacesLocked()
3.1 requestTraversalLocked
3.2 sendEmptyMessage
H
+ handleMessage() : void

WMS&DMS


pkg cn.gswift.david.ex23
2. scheduleTraversalLocked
DisplayManagerService
- mDisplayDevices : DisplayDevice
- mDisplayAdapterListener : int
- mHandler : DisplayManagerHandler
- mUiHandler : Handler
- mWindowManagerFuncs : int
- mInputManagerFuncs : int
- mSyncRoot : SyncRoot
- mDisplayAdapters : DisplayAdapter
- mDefaultViewport : DisplayViewport
sendMessage
DisplayManagerHandler
+ handleMessage() : void
3. requestTraversal()
WindowManagerFuncs
4. requestTraversalLocked()
WindowManagerService
~ mPolicy : WindowManagerPolicy
- mWindowMap : HashMap
- mWindowChangeListeners :WindowChangeListener : ArrayList
1. setDisplayHasContent
sendEmptyMessage
H
+ handleMessage() : void
5. performLayoutAndPlaceSurfacesLocked

参考资料：

Android4.2 http://code.metager.de/source/xref/android/4.2/

Android Graphics Architecture I ,himmele,http://himmele.googlecode.com

Inside Android's UI -Embedded Linux Conference Europe 2012 ,
Karim Yaghmour ,2012，http://www.opersys.com/community/docs

About Me

- I have been working as a product-designer specializing in software/Web application design and development. I am passionate about mobile application development and became interested in Android programming when the platform was launched by Google. Thus I was not programming on Android projects, I spent spare time reading technical blogs, researching, analyzing, and testing mobile applications, as a software consultancy specialized in android technologies.
- In my product-design time, in the developing, I've encountered too many program manage troubles that suffer due to poor communication and code design, I know that help them to understand the system framework is very important. I amd experienced in system and application layers, my goal is simple: help someone who wishes to better understand the **Android framework** in java、JNI and C/C++ libraries.

- Please also check my article and slides on this http://blog.sina.com.cn/gswift

Contact: Zhiyong.liu@aliyun.com

http://weibo.com/gswift