1/4 Featured Sessions,

Kandroid S/W Fundamentals Study Group

Android Graphics and Android EGL

www.kandroid.org 운영자

양정수 (yangjeongsoo@gmail.com)

목 차

Why review Android EGL?

- Google IO 2012 Project Butter
- Change history of Android Internal graphics architecture
- TLS based EGLContext issues

Android Graphics and EGL

- Understanding Android Graphics Internals
- EGLSurface and ANativeWindow
- EGLContext and Thread Local Storage
- EGL Implementation : SurfaceFlinger & HardwareRenderer

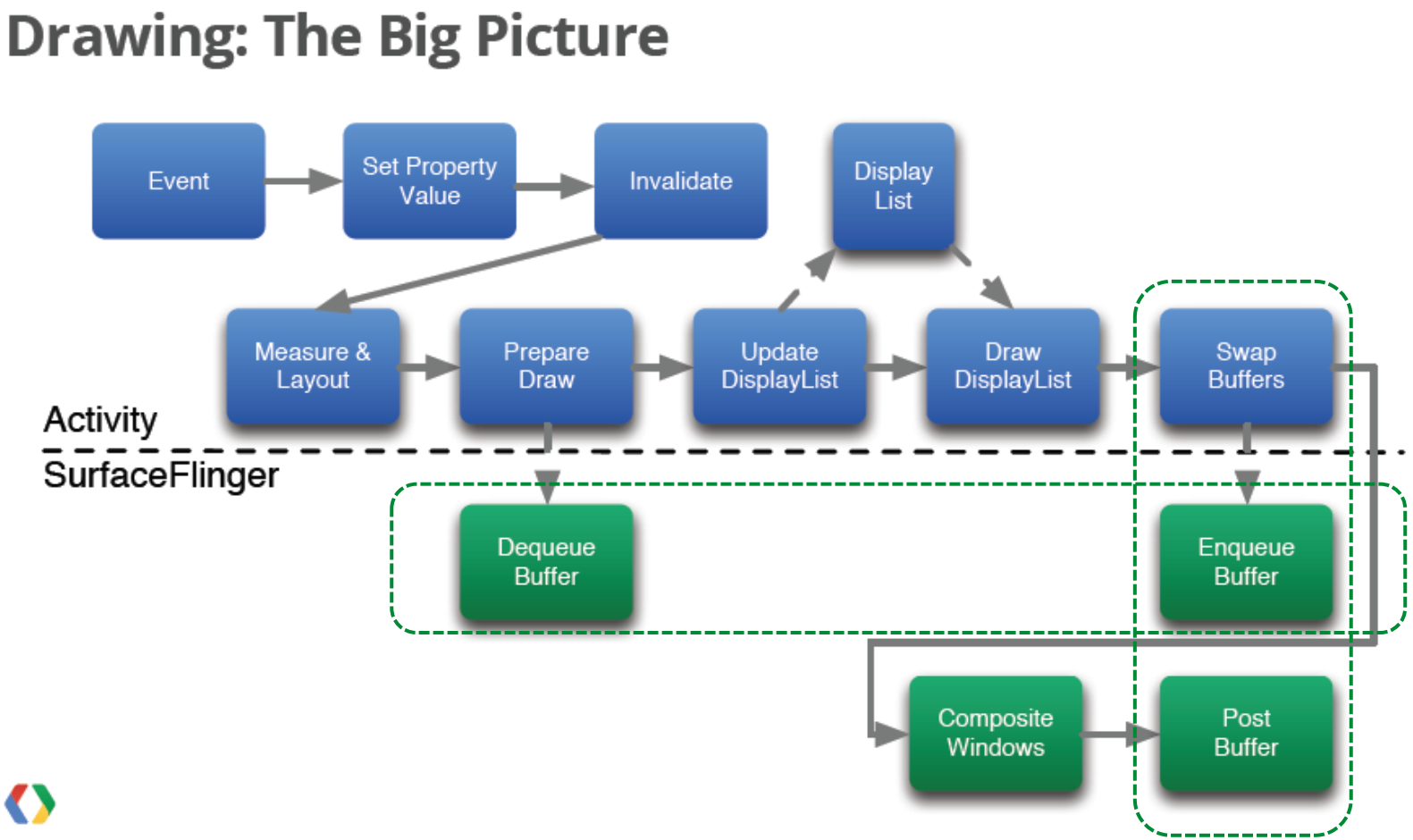
Useful Topics

- Performance Analysis Knowledge : Systrace, dumpsys
- OpenGL|ES Context Management in MT Environment
- Unified Memory Management

Why review Android EGL?

Google I/O 12

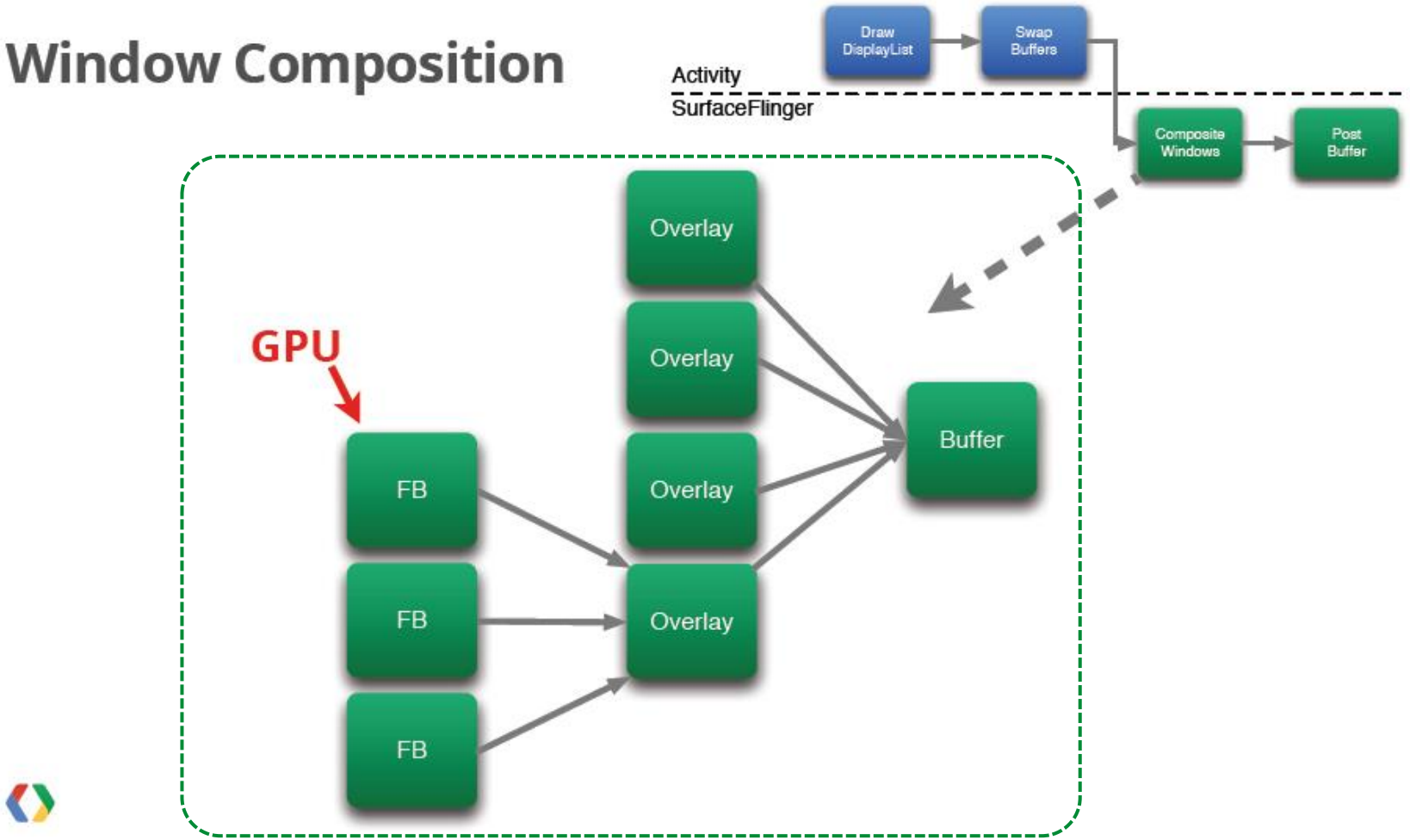
For Butter or Worse
Smoothing Out Performance in Android UIs



Why review Android EGL?

Google I/O 12

For Butter or Worse
Smoothing Out Performance in Android UIs



Why review Android EGL?

./frameworks/native/services/surfaceflinger

GLES2.0 SurfaceFlinger

Android 4.4 upgrades its SurfaceFlinger from OpenGL ES 1.0 to OpenGL ES 2.0. This boosts performance by using multi-texturing, and it improves color calibration and supports more advanced special effects.

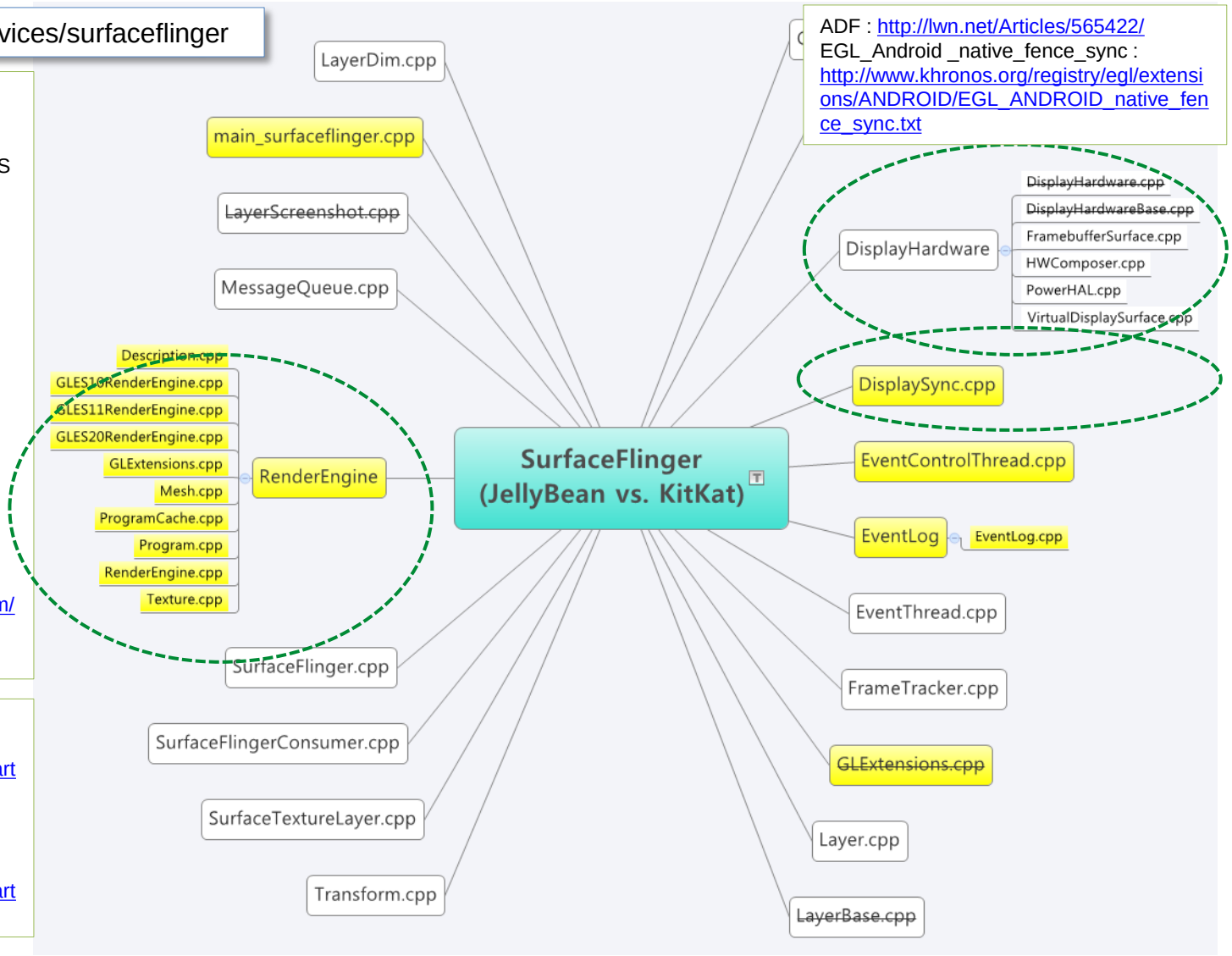
New Hardware Composer support for virtual displays
The latest version of Android Hardware Composer, HWComposer 1.3, supports hardware composition of one virtual display in addition to the primary, external (e.g. HDMI) display, and has improved OpenGL ES interoperability."

https://groups.google.com/forum/#!topic/android-developers/RQTTqLTJfUs

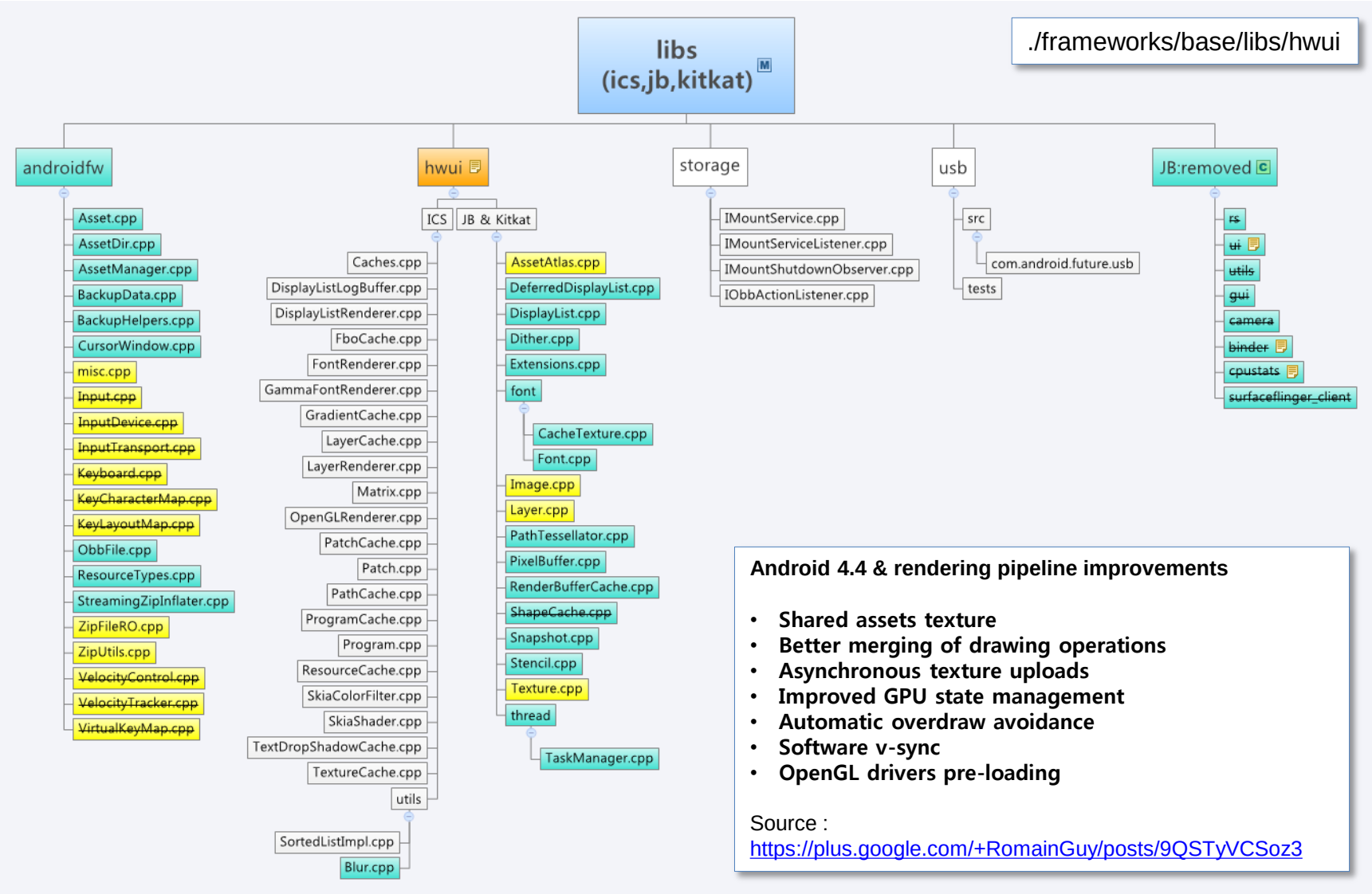
Android 4.4 (KitKat) Design Pattern-Graphics Subsystem
http://blog.csdn.net/jinzhuojun/article/details/17427491

Android 4.4 (KitKat) in virtualization VSync signal
http://blog.csdn.net/jinzhuojun/article/details/17293325

ADF : http://lwn.net/Articles/565422/
EGL_Android _native_fence_sync : http://www.khronos.org/registry/egl/extensions/ANDROID/EGL_ANDROID_native_fence_sync.txt



Why review Android EGL?

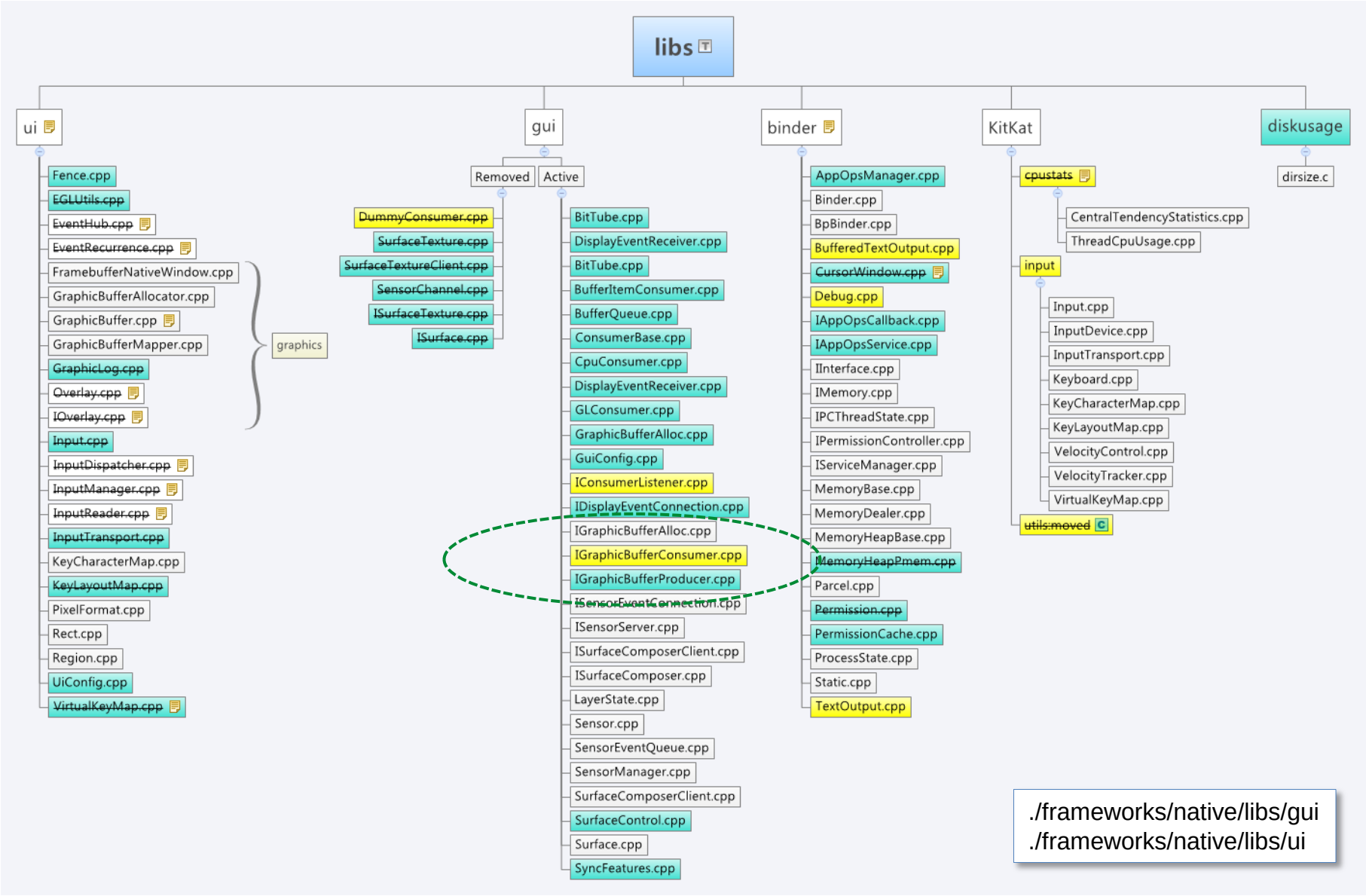


Android 4.4 & rendering pipeline improvements

- **Shared assets texture**
- **Better merging of drawing operations**
- **Asynchronous texture uploads**
- **Improved GPU state management**
- **Automatic overdraw avoidance**
- **Software v-sync**
- **OpenGL drivers pre-loading**

Source :
https://plus.google.com/+RomainGuy/posts/9QSTyVCSoz3

Why review Android EGL?



Why review Android EGL?

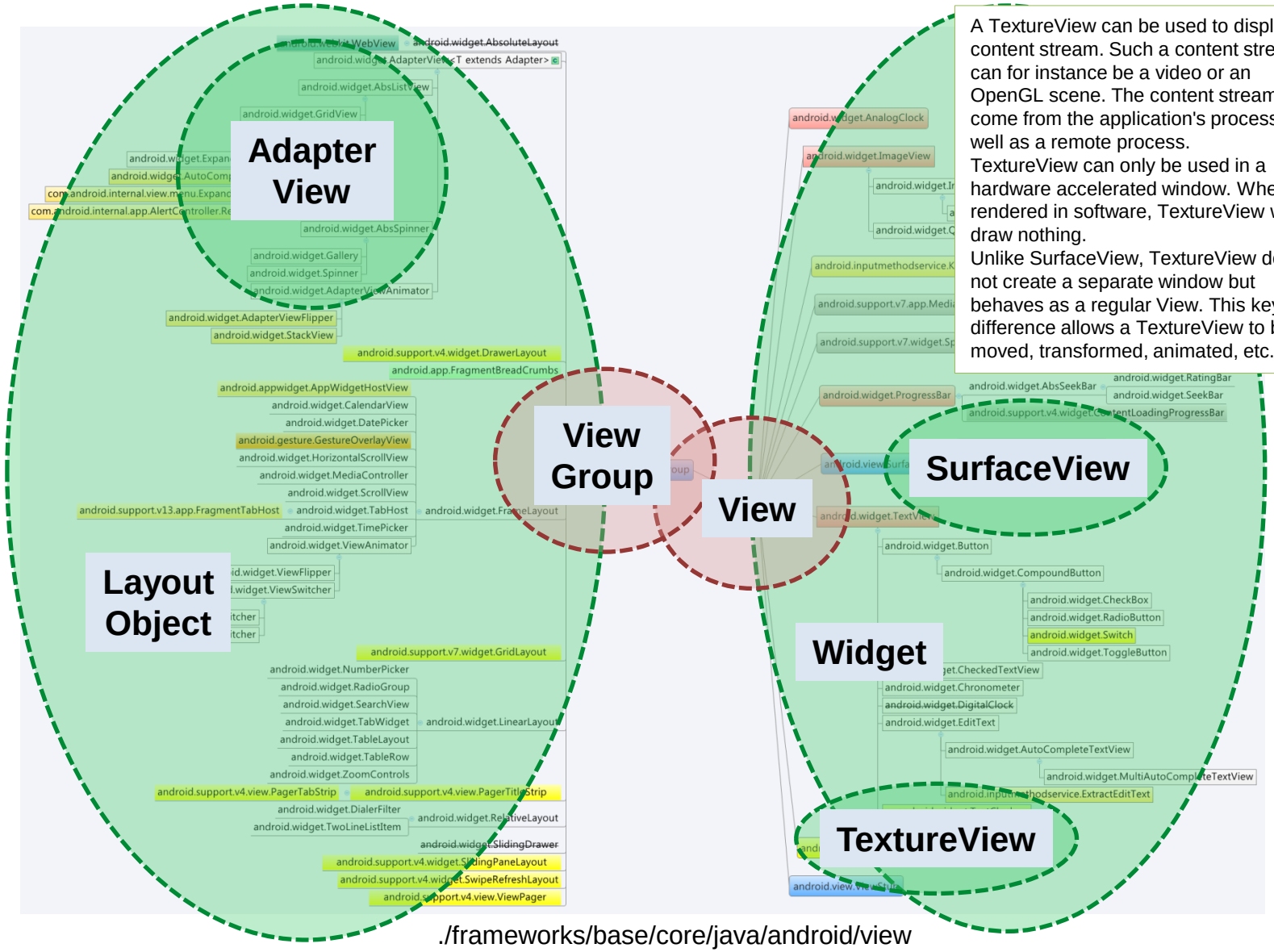
./frameworks/base/core/java/android/view/SurfaceView.java

```
public class SurfaceView extends View {

    final Surface mSurface = new Surface();

    private SurfaceHolder mSurfaceHolder = new SurfaceHolder() {

        public void addCallback(Callback callback) { }

        public void setType(int type) {
            // SURFACE_TYPE_NORMAL (SURFACE_TYPE_HARDWARE, SURFACE_TYPE_GPU)
            // SURFACE_TYPE_PUSH_BUFFERS
        }

        public Canvas lockCanvas() { }
        public void unlockCanvasAndPost(Canvas canvas) { }
        public Surface getSurface() { }

    }

}
```

Deprecated in API Level 5 (D)

Deprecated in API Level 11 (H)

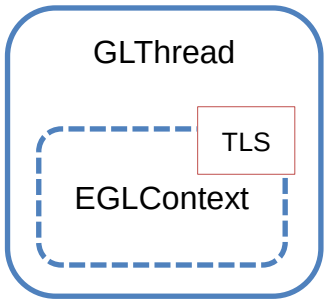
Why review Android EGL?

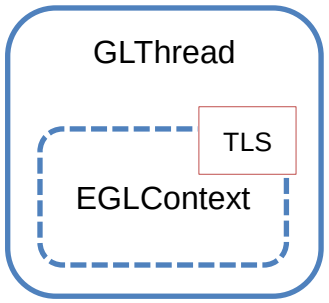


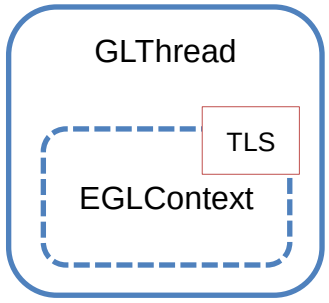
A TextureView can be used to display a content stream. Such a content stream can for instance be a video or an OpenGL scene. The content stream can come from the application's process as well as a remote process.
TextureView can only be used in a hardware accelerated window. When rendered in software, TextureView will draw nothing.
Unlike SurfaceView, TextureView does not create a separate window but behaves as a regular View. This key difference allows a TextureView to be moved, transformed, animated, etc.

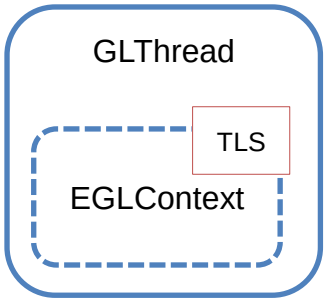
./frameworks/base/core/java/android/view

Why review Android EGL?









If a green dotted area each have GLSurfaceView in the Activity above, is there any problem?

Passion is not about *speed*,
but **DIRECTION** !

Why review Android EGL?

What is the Direction?

SW : Standard API (Java, NDK Stable API)
HW : OpenGLES, OpenSLES, OpenMAX

EGL™ is an interface between Khronos rendering APIs such as OpenGL ES or OpenVG and the underlying native platform window system

목 차

Why review Android EGL?

- Google IO 2012 Project Butter
- Change history of Android Internal graphics architecture
- TLS based EGLContext issues

Android Graphics and EGL

- Understanding Android Graphics Internals
- Android EGL Overview
- EGLSurface and ANativeWindow
- EGLContext and Thread Local Storage
- EGL Implementation : SurfaceFlinger & HardwareRenderer

Useful Topics

- Performance Analysis Knowledge : Systrace, dumpsys
- OpenGL|ES Context Management in MT Environment
- Unified Memory Management

Understanding Android Graphics Internals

Graphics Basics

- Surfaces and centralized surface composition
- What is a window on android? (ANativeWindow)
- The native graphic buffer handle
- The native graphic buffer ANativeWindowBuffer
- Flattenable
- GraphicBuffer class
- Surface class

The Graphics Surface Service Interfaces

- ISurfaceComposer
- ISurfaceComposerClient

Android 4.4 (KitKat) Design Pattern-Graphics Subsystem
http://blog.csdn.net/jinzhuojun/article/details/17427491

Graphics Buffer Handling Interfaces

- IGraphicBufferProducer
- IGraphicBufferAlloc

Android 4.4 (KitKat) in virtualization VSync signal
http://blog.csdn.net/jinzhuojun/article/details/17293325

Gralloc and HWComposer

- gralloc
- hwcomposer

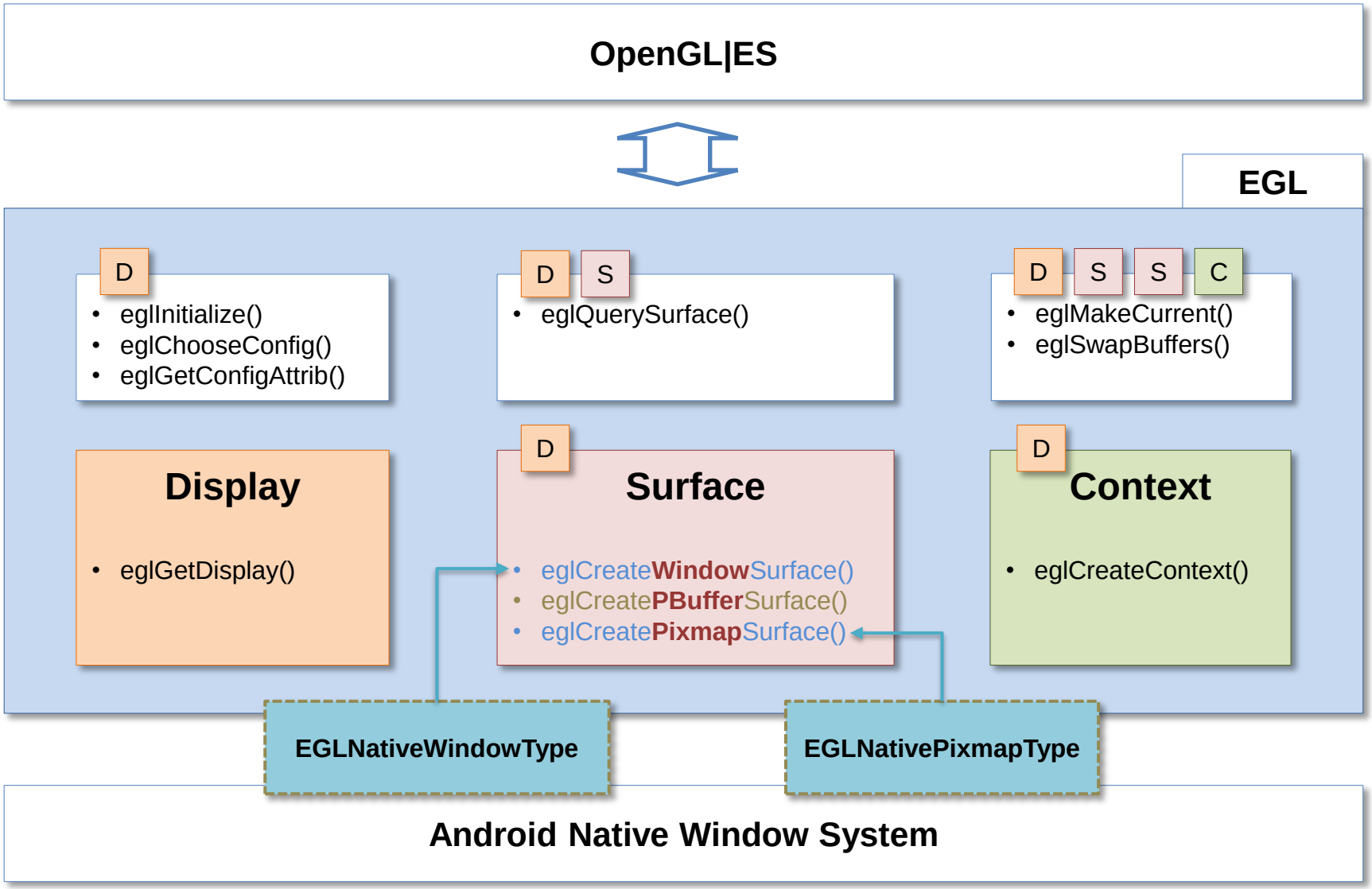
http://blog.csdn.net/yangwen123/article/details/22647255

SurfaceFlinger

- Periodical vsync event sources
- Spontaneous vsync event sources
- *BufferQueue*
- *BufferQueue::BufferSlot and BufferQueue::BufferSlot::BufferState*
- *SurfaceFlingerConsumer* and *GLConsumer* class
- *Layer* Class
- *FramebufferSurface* and *DisplayDevice*
- *HWComposer*
- *SurfaceFlinger*

Source : https://charleszblog.wordpress.com/category/android-2/graphics-android/

Android EGL Overview



Android EGL Overview

- eglGetDisplay()
- eglInitialize()
- eglChooseConfig()

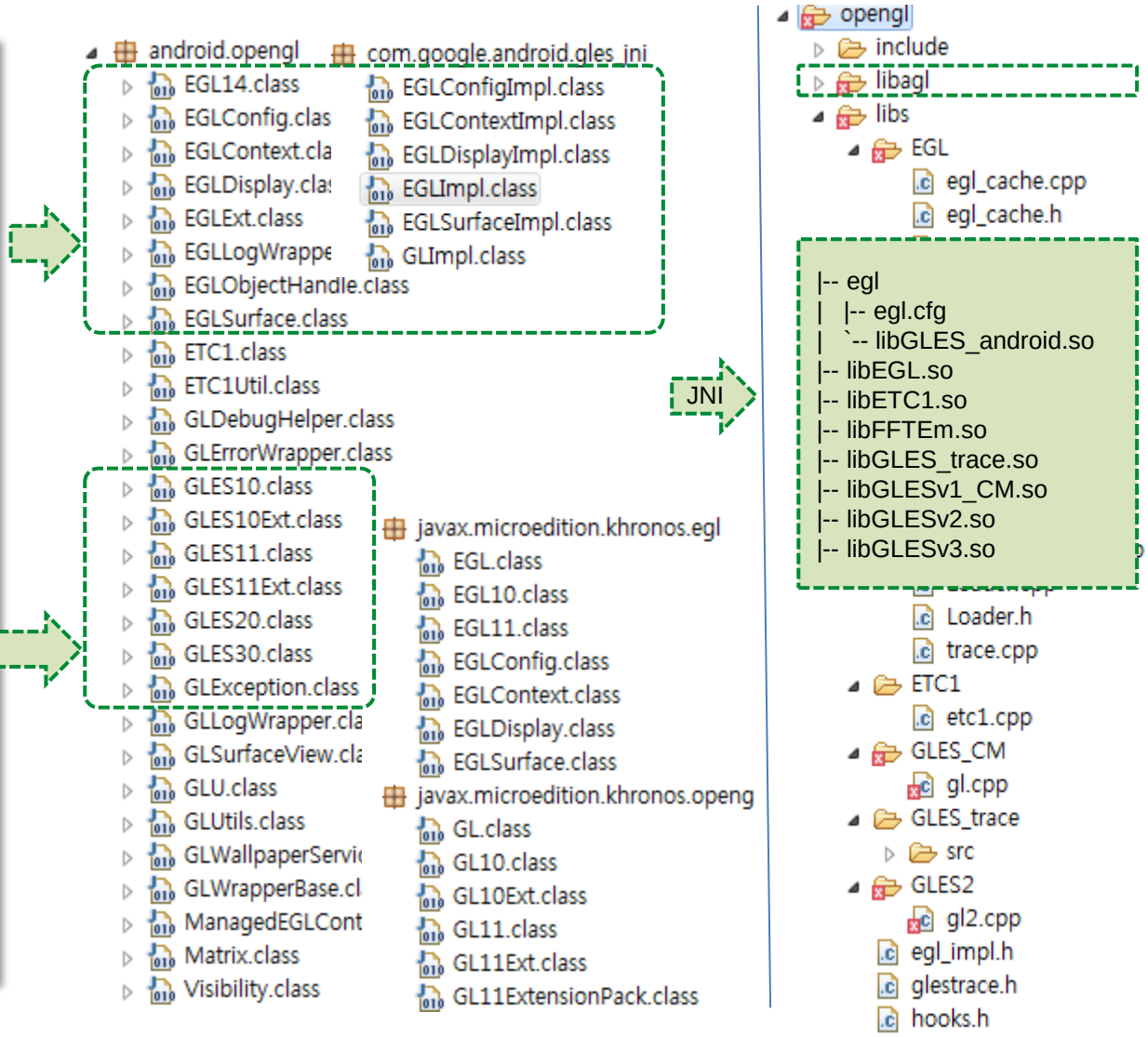
- eglCreateContext()

- eglCreateWindowSurface()
  eglCreatePBufferSurface()
  eglCreatePixmapSurface()

- eglMakeCurrent()

draw()

- eglSwapBuffers()



EGLSurface and ANativeWindow

- eglGetDisplay()
- eglInitialize()
- eglChooseConfig()

- eglCreateContext()

- *eglCreateWindowSurface()*
  eglCreatePBufferSurface()
  eglCreatePixmapSurface()

- eglMakeCurrent()

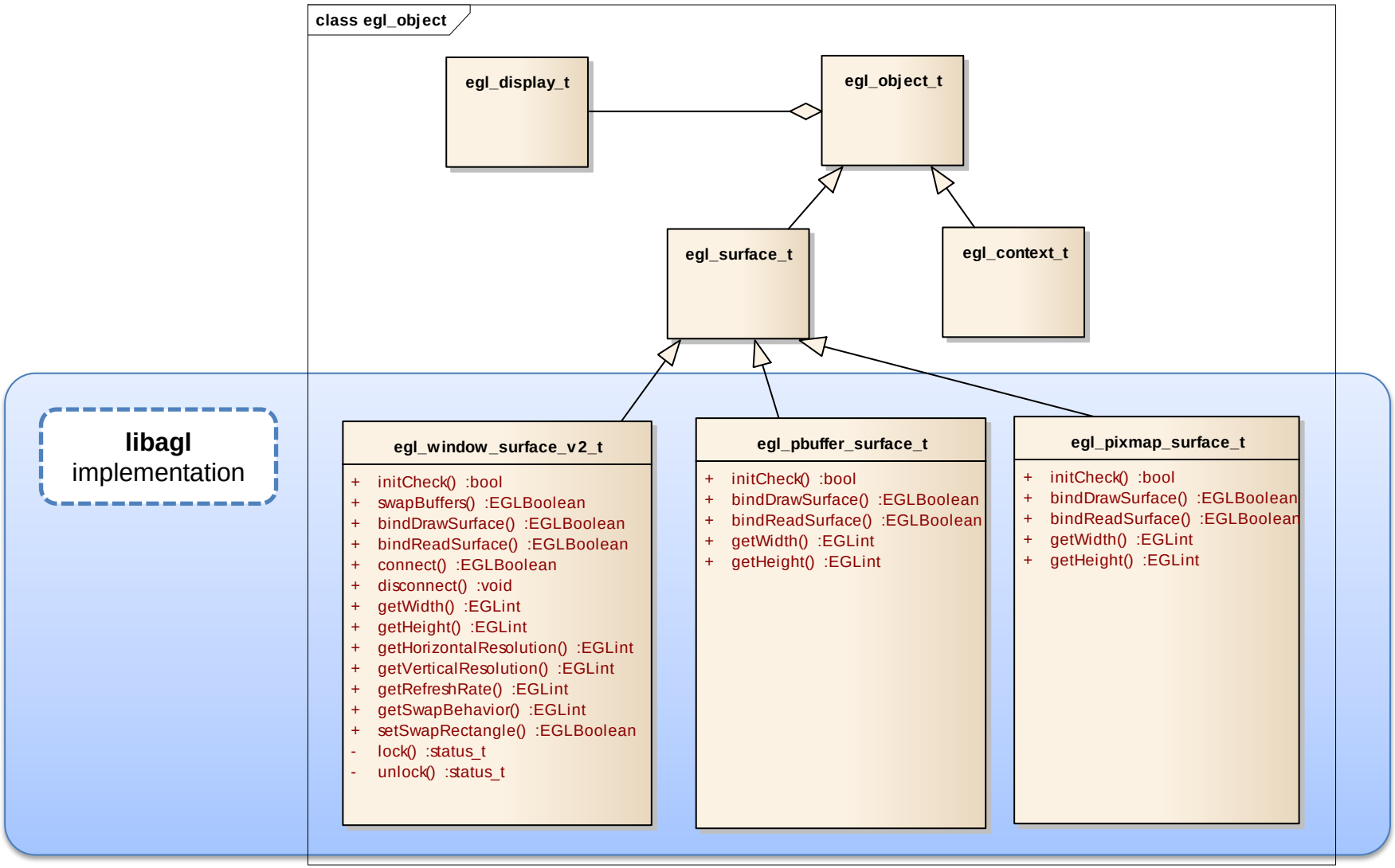
draw()

- eglSwapBuffers()

libagl

```
EGLSurface eglCreateWindowSurface(
                        EGLDisplay dpy,
                        EGLConfig config,
                        NativeWindowType window,
                        const EGLint *attrib_list)
{ …
    ANativeWindow* anw
      = reinterpret_cast<ANativeWindow*>(window);
    anw->setSwapInterval(anw, 1);

    EGLSurface surface = cnx->egl.eglCreateWindowSurface(
                iDpy, config, window, attrib_list);
    if (surface != EGL_NO_SURFACE) {
        egl_surface_t* s = new egl_surface_t(dp.get(), config, window,
                                             surface, cnx);
        return s;
    }
}

EGLSurface eglCreatePixmapSurface(
                        EGLDisplay dpy,
                        EGLConfig config,
                        NativePixmapType pixmap,
                        const EGLint *attrib_list) { … }

EGLSurface eglCreatePbufferSurface(
                        EGLDisplay dpy,
                        EGLConfig config,
                        const EGLint *attrib_list) { … }
```

EGLSurface and ANativeWindow

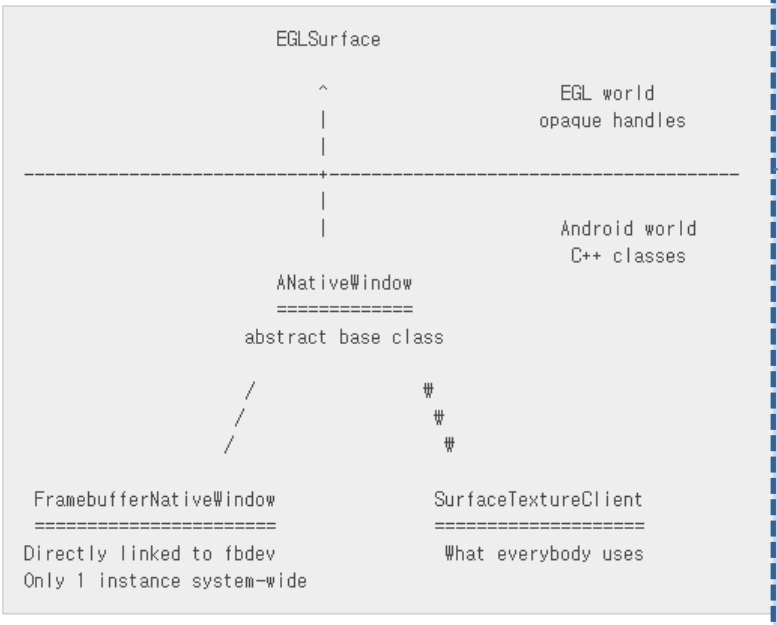


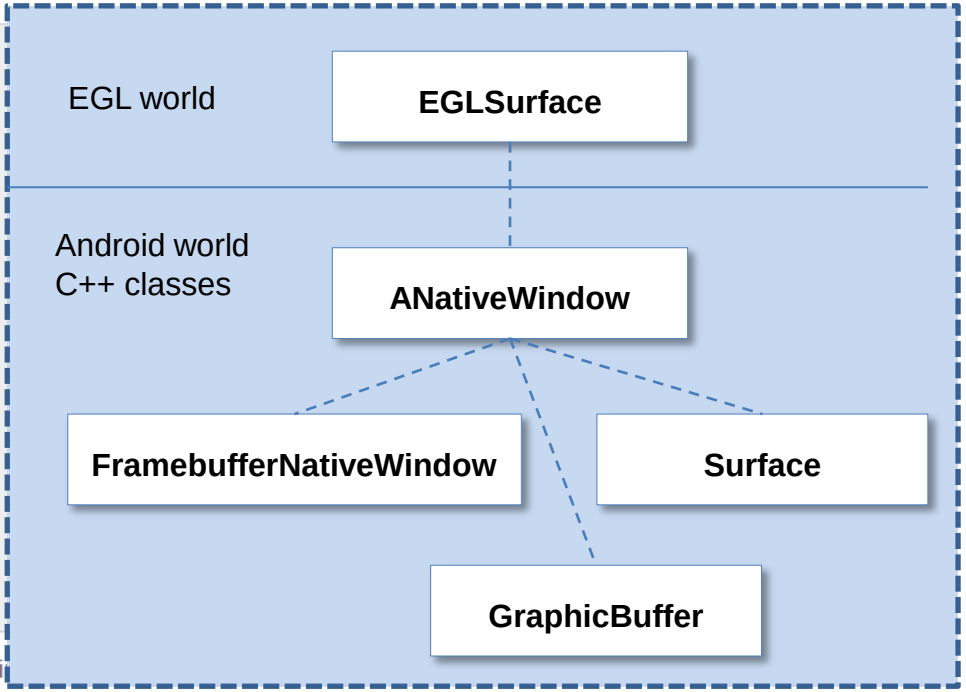
EGLSurface and ANativeWindow

Some terminology: EGLSurface, ANativeWindow, etc.

EGLSurface is a portable EGL abstraction for a possibly multi-buffered render target.

ANativeWindow is the Android-specific abstraction for a possibly multi-buffered render target. The eglCreateWindowSurface function allows to create an EGLSurface from an ANativeWindow. There are two concrete implementations of ANativeWindow in Android: **FramebufferNativeWindow** and **SurfaceTextureClient**.





While ANativeWindow abstracts a possibly multi-buffered render target, the i
ANativeWindowBuffer.

The concrete implementation of ANativeWindowBuffer is **GraphicBuffer**, the class discussed above in this document.

Source : https://wiki.mozilla.org/Platform/GFX/Gralloc

EGLContext and Thread Local Storage

- eglGetDisplay()
- eglInitialize()
- eglChooseConfig()

- eglCreateContext()

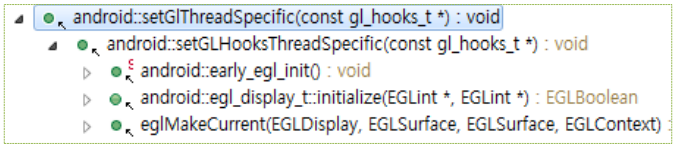
- eglCreateWindowSurface()
  eglCreatePBufferSurface()
  eglCreatePixmapSurface()

- *eglMakeCurrent()*

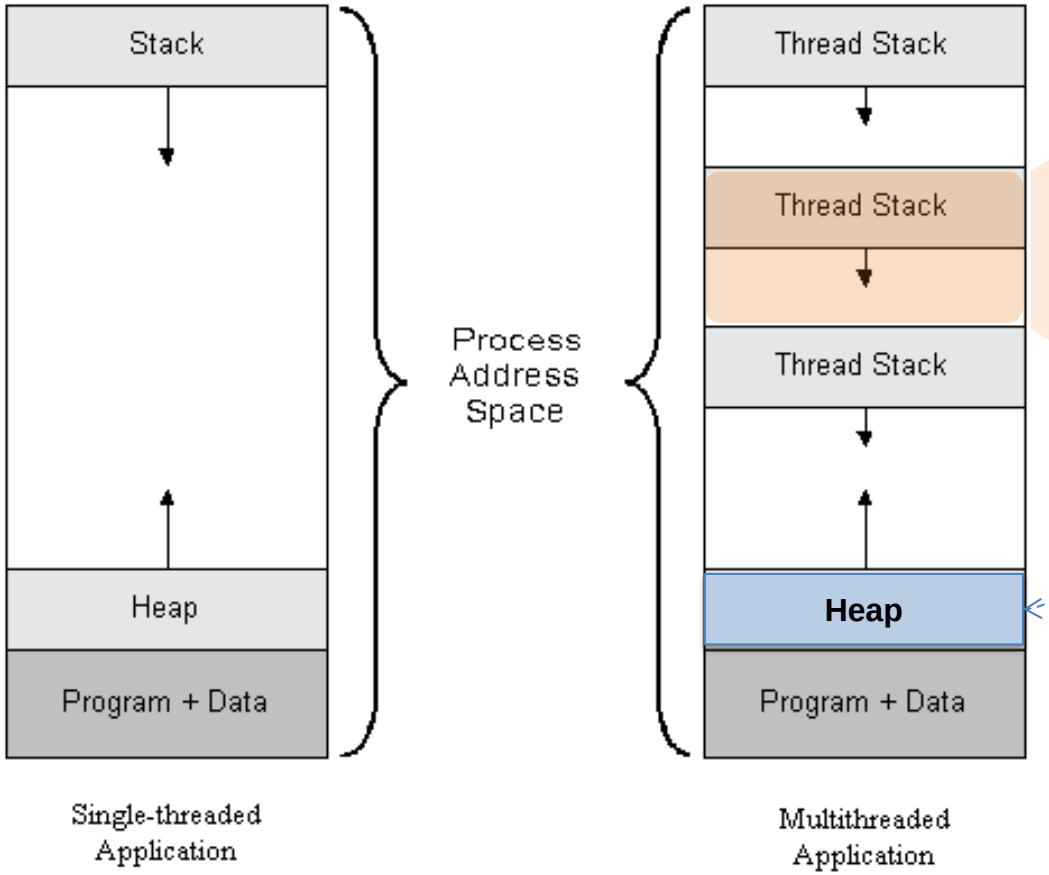
draw()

- eglSwapBuffers()

```
EGLBoolean eglMakeCurrent(
                      EGLDisplay dpy,
                      EGLSurface draw,
                      EGLSurface read,
                      EGLContext ctx)
{ ...

   EGLBoolean result = dp->makeCurrent(c, cur_c,
                        draw, read, ctx,
                        impl_draw, impl_read, impl_ctx);

   if (result == EGL_TRUE) {
     if (c) {
        setGLHooksThreadSpecific(c->cnx->hooks[c->version]);
        egl_tls_t::setContext(ctx);
        _c.acquire();
        _r.acquire();
        _d.acquire();
     } else {
        setGLHooksThreadSpecific(&gHooksNoContext);
        egl_tls_t::setContext(EGL_NO_CONTEXT);
     }
   }
   ...
   return result;
}
```

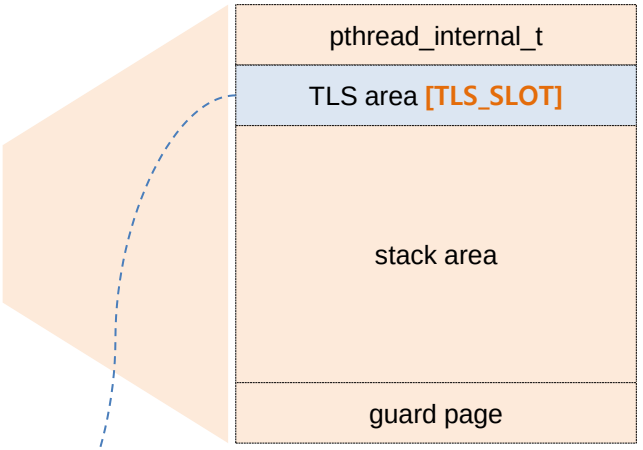


EGLContext and Thread Local Storage



Source : http://www.roguewave.com/portals/0/products/legacy-hpp/docs/thrug/3-9.html

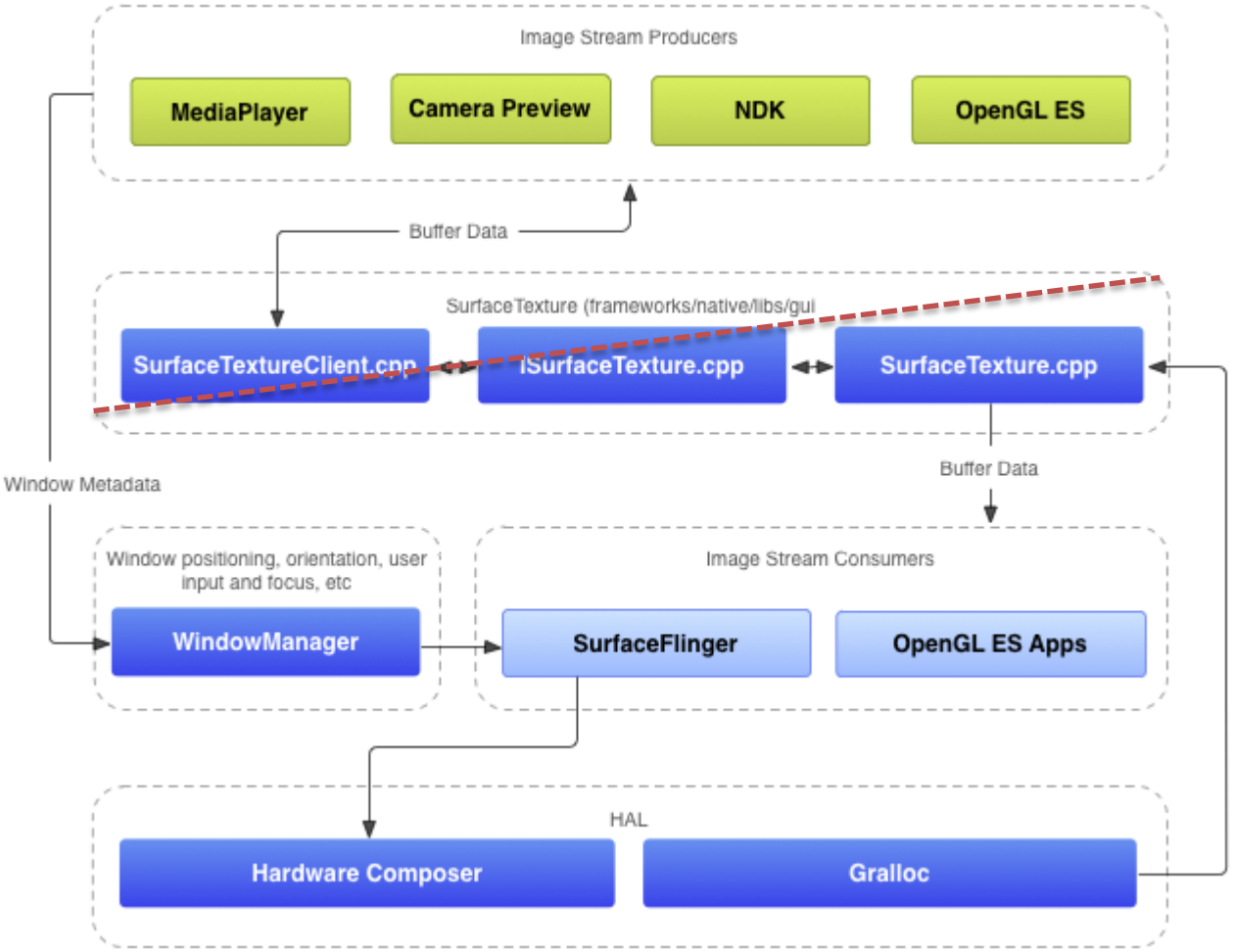
The layout of the thread's stack



Source : bionic/libc/bionic/pthread.c

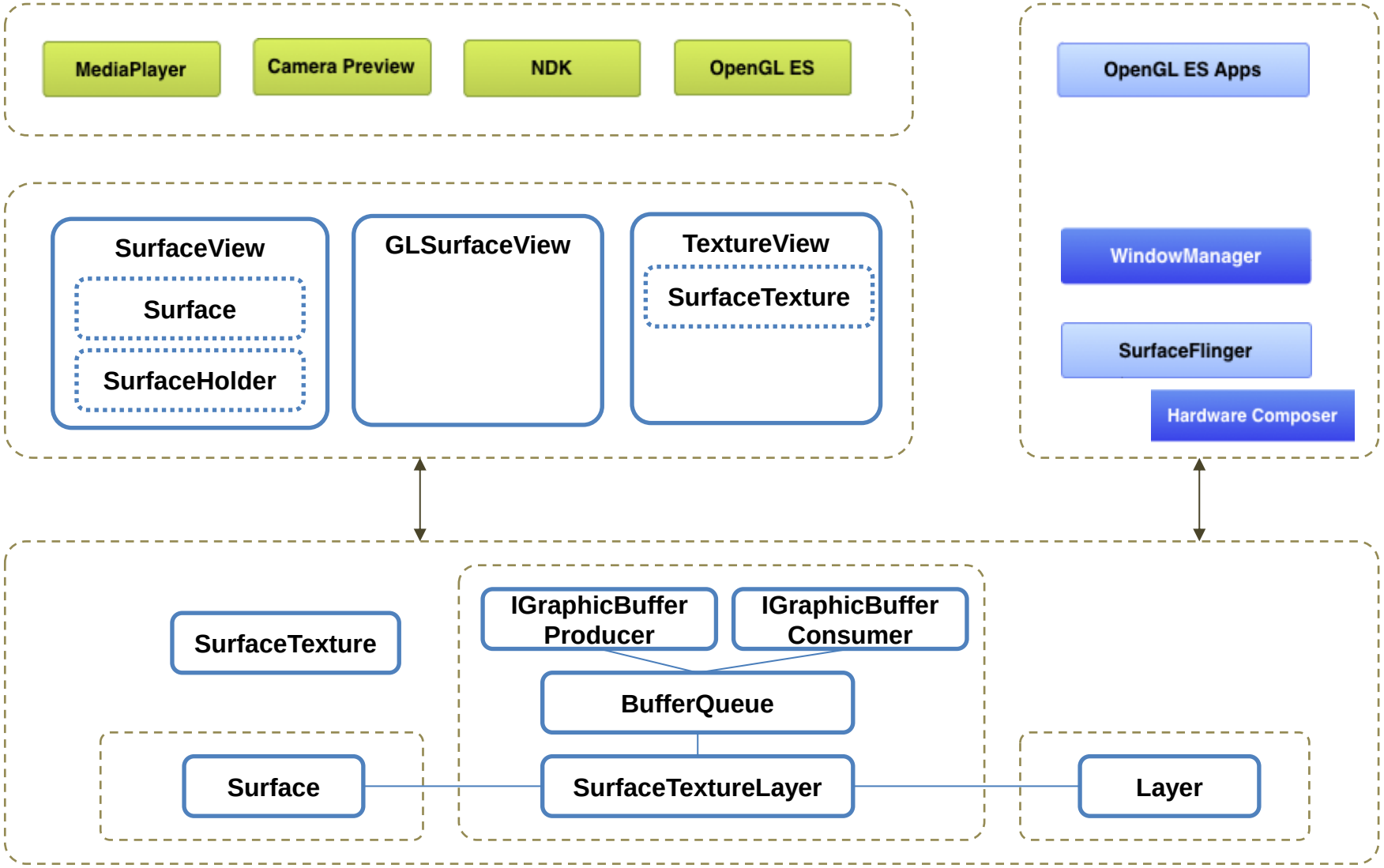
```
enum {
  TLS_SLOT_SELF = 0,
  TLS_SLOT_THREAD_ID,
  TLS_SLOT_ERRNO,
  TLS_SLOT_OPENGL_API = 3,
  TLS_SLOT_OPENGL = 4,
  TLS_SLOT_BIONIC_PREINIT
     = TLS_SLOT_OPENGL_API,
  TLS_SLOT_STACK_GUARD = 5,
  TLS_SLOT_DLERROR,
  TLS_SLOT_FIRST_USER_SLOT
}; // bionic_tls.h
```

EGLImplementation : HWRenderer and SurfaceFlinger



Source : https://source.android.com/devices/graphics.html

EGLImplementation : HWRenderer and SurfaceFlinger



EGLImplementation : HWRenderer and SurfaceFlinger

screensavers : com.android.dreams.basic.ColorsGLRenderer : SurfaceTexture
Gallery2 : com.android.photos.views.BlockingGLTextureView : SurfaceTexture
Camera2 : com.android.camera.SurfaceTextureRenderer : SurfaceTexture
Nfc : com.android.nfc.FireflyRenderer : SurfaceTexture
Camera : com.android.camera.MosaicPreviewRenderer : SurfaceTexture
Launcher3 : com.android.photos.views.BlockingGLTextureView : SurfaceTexture

frameworks : com.android.systemui.ImageWallpaper : SurfaceHolder
frameworks : com.android.photos.views.BlockingGLTextureView : SurfaceTexture
frameworks :com.android.server.power.ElectronBeam : Surface
frameworks : android.view.HardwareRenderer.GlRenderer : Surface
frameworks : android.opengl.GLSurfaceView.DefaultWindowSurfaceFactory : SurfaceHolder
frameworks : android.opengl.EGLLogWrapper : Surface

eglCreateWindowSurface()
- **android.opengl.EGL14**
- **com.google.android.gles_jni.EGLImpl implements EGL10**

```
EGLSurface surface;

if (sur != null) {
    surface = _eglCreateWindowSurface(dpy, config, sur, attrib_list, offset);
}
else if (win instanceof SurfaceTexture) {
    surface = _eglCreateWindowSurfaceTexture(dpy, config, win, attrib_list,
                                                                    offset);
}
```

목 차

Why review Android EGL?

- Google IO 2012 Project Butter
- Change history of Android Internal graphics architecture
- TLS based EGLContext issues

Android Graphics and EGL

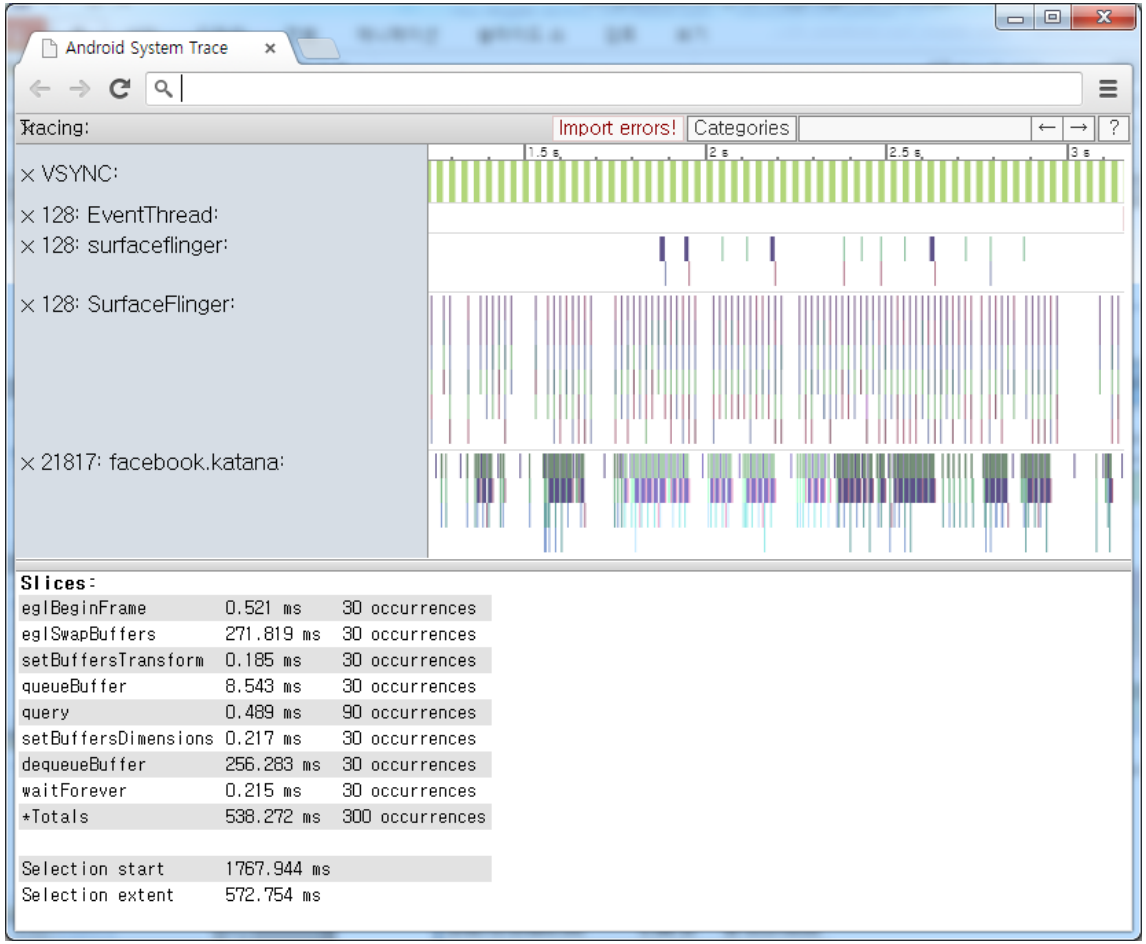
- Understanding Android Graphics Internals
- EGLSurface and ANativeWindow
- EGLContext and Thread Local Storage
- EGL Implementation : SurfaceFlinger & HardwareRenderer

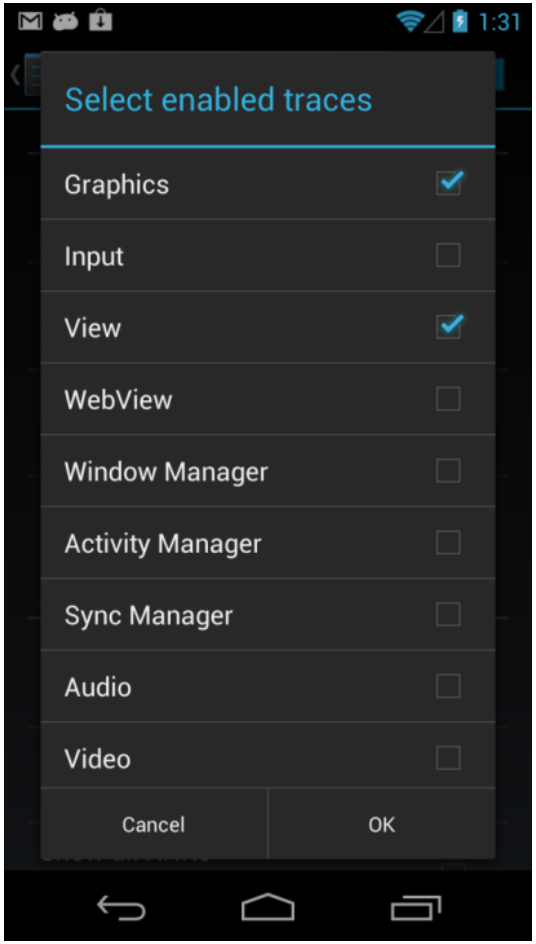
Useful Topics

- Performance Analysis Knowledge : Systrace, dumpsys
- OpenGL|ES Context Management in MT Environment
- Unified Memory Management

Performance Analysis Knowledge : systrace

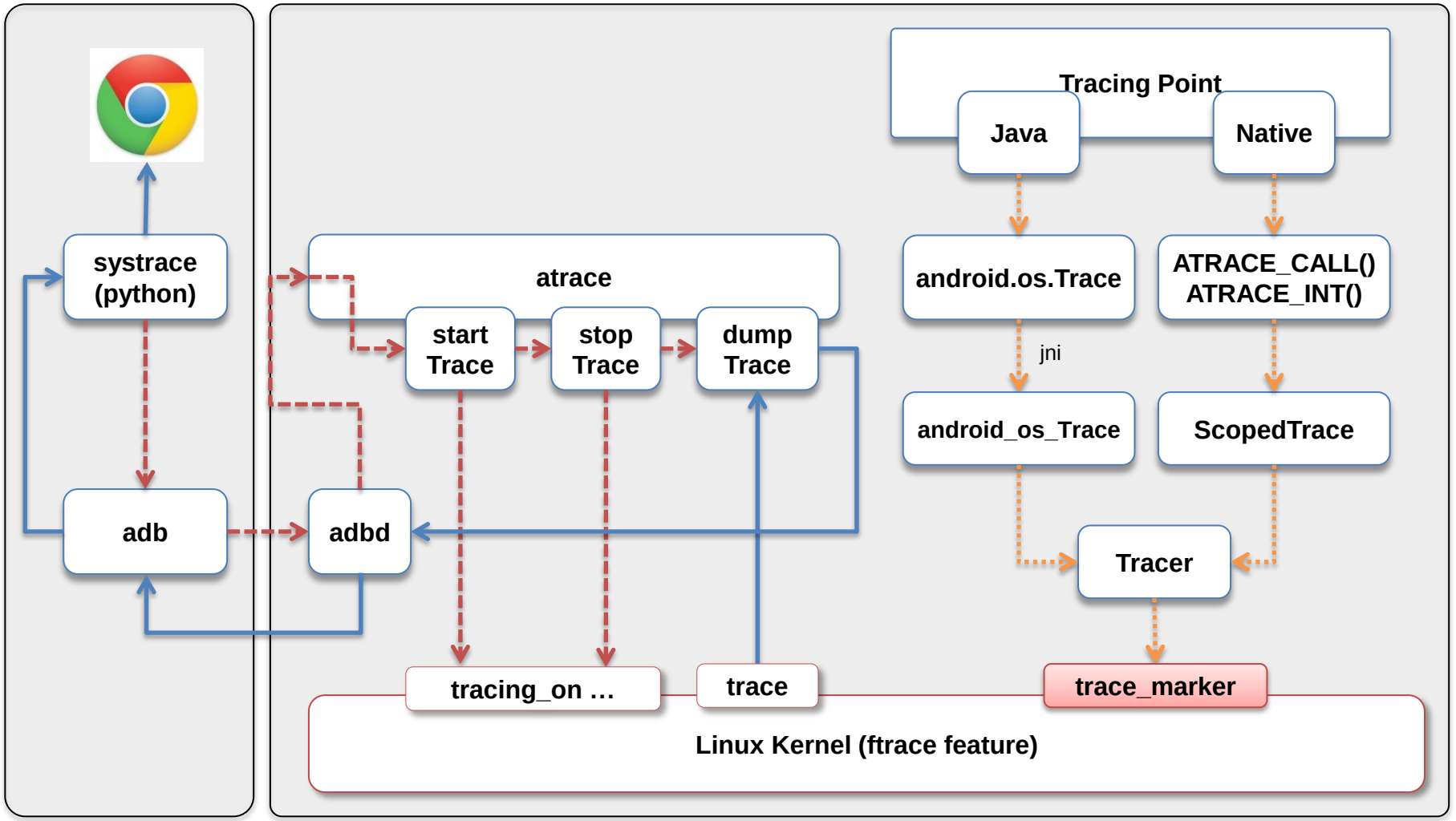
http://developer.android.com/tools/debugging/systrace.html





Performance Analysis Knowledge : systrace

https://code.google.com/p/trace-viewer/

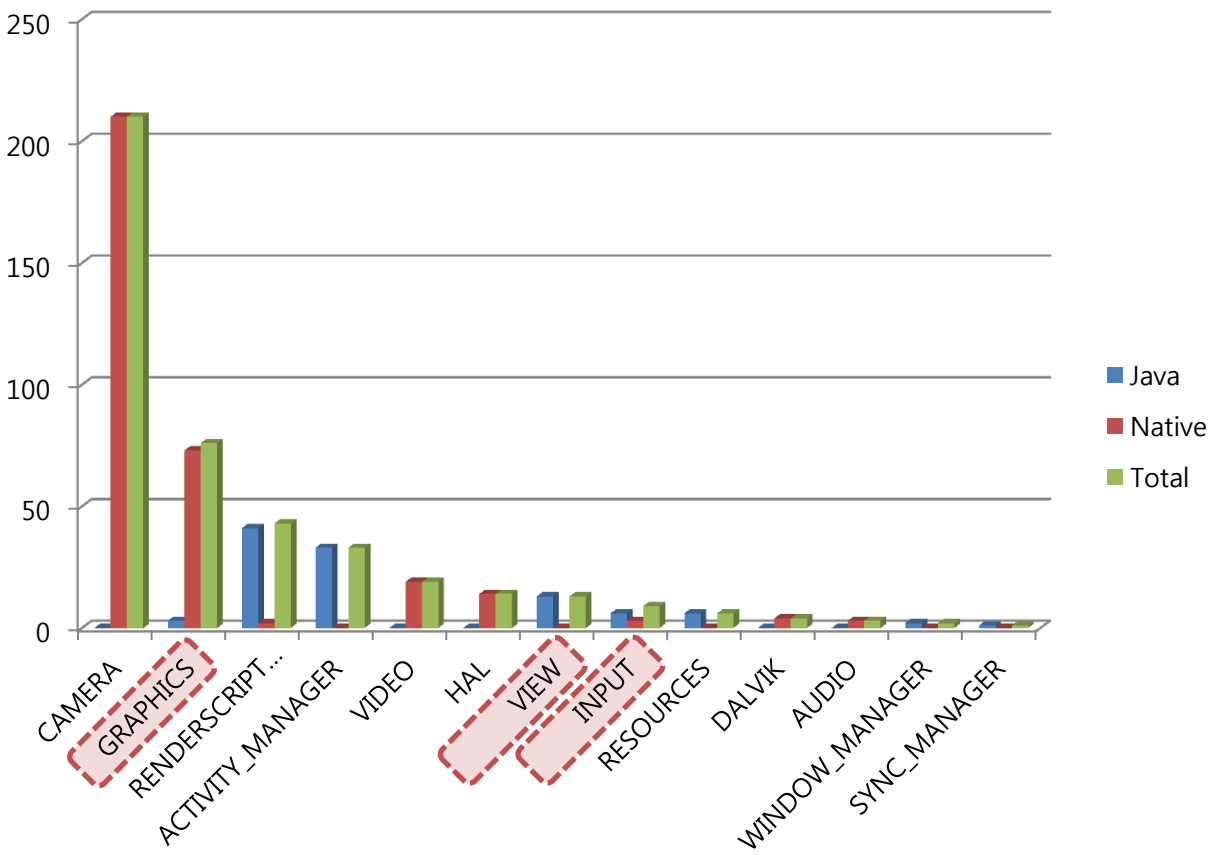


- atrace_args = ['adb', 'shell', 'setprop', 'debug.atrace.tags.enableflags', hex(flags)]
- atrace_args = ['adb', 'shell', 'atrace', '-z'] + args

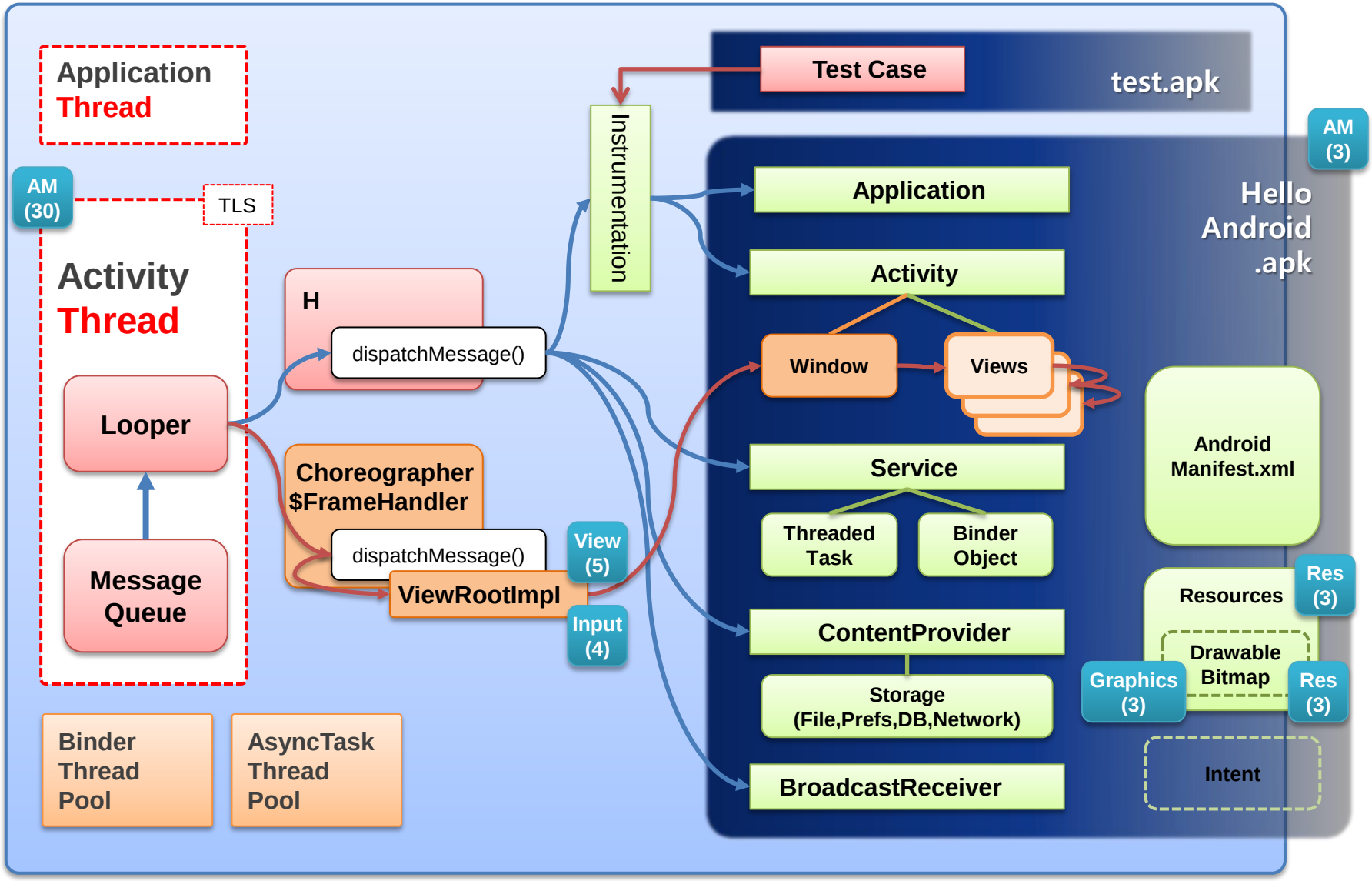
Performance Analysis Knowledge : systrace

ATRACE_TAG

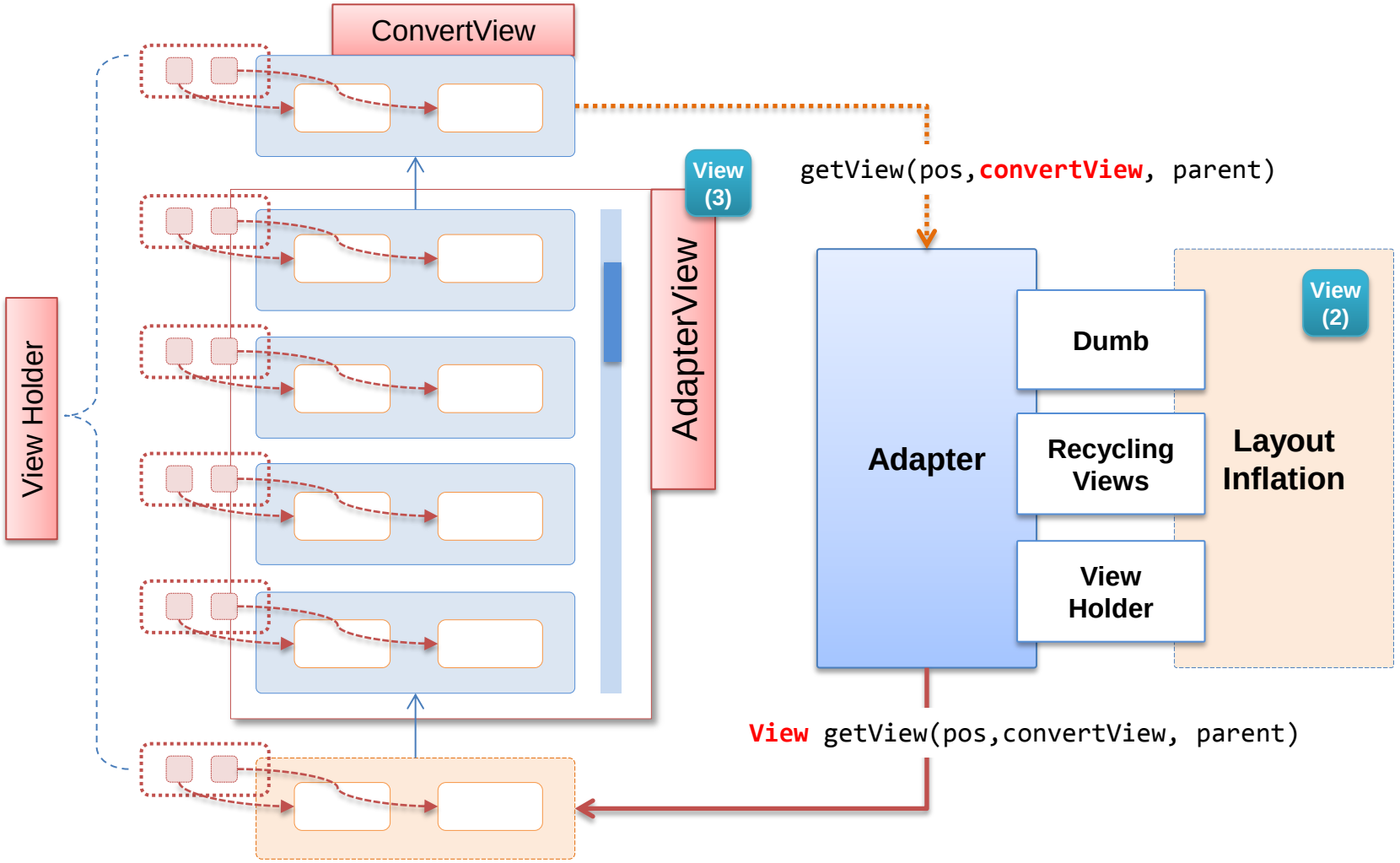
- *GRAPHICS*
- *INPUT*
- *VIEW*
- *WEBVIEW*
- *WINDOW_MANAGER*
- *ACTIVITY_MANAGER*
- *SYNC_MANAGER*
- *AUDIO*
- *VIDEO*
- *CAMERA*
- *HAL*
- *APP*
- *RESOURECS*
- *DALVIK*
- *RS*



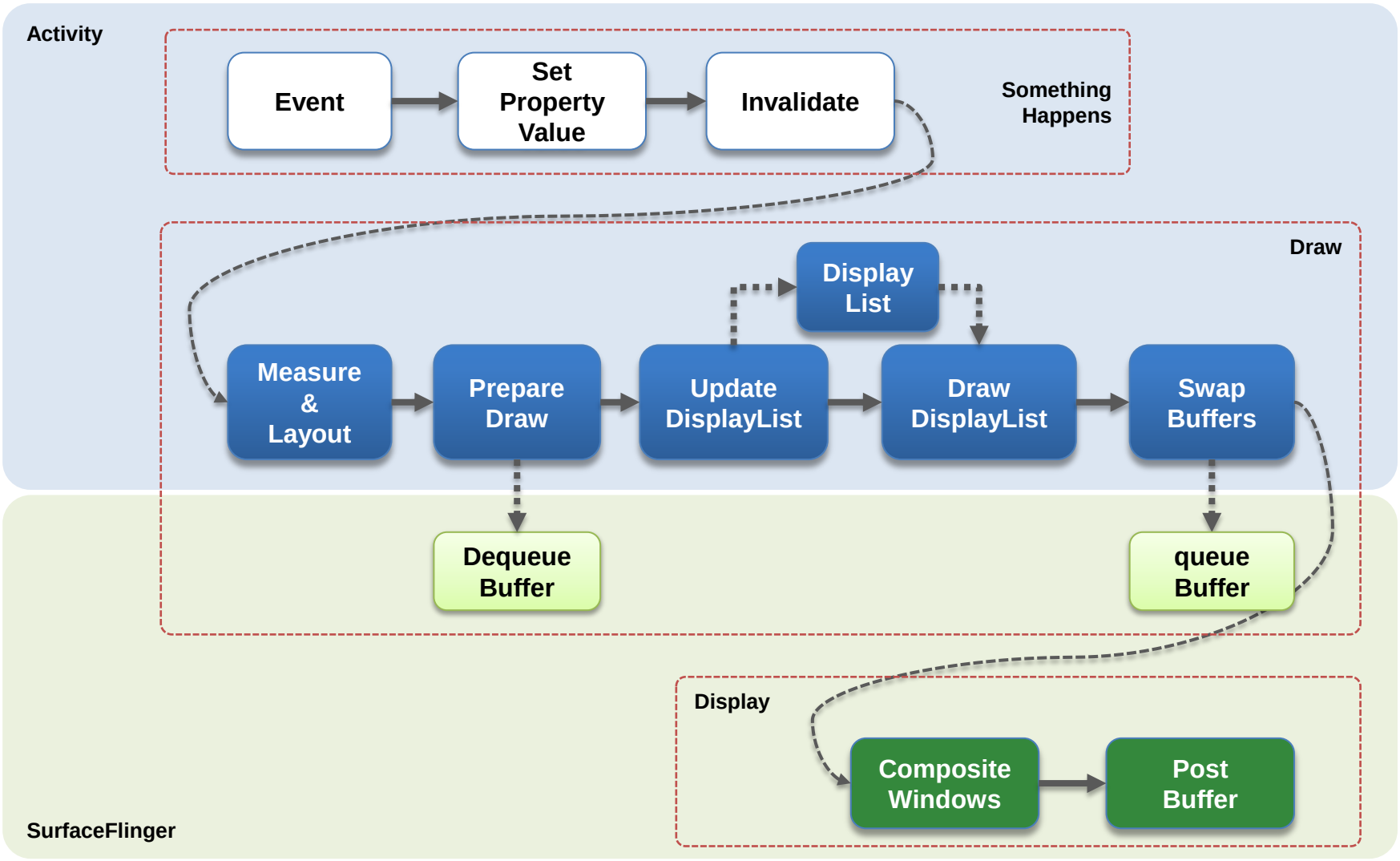
Performance Analysis Knowledge : systrace



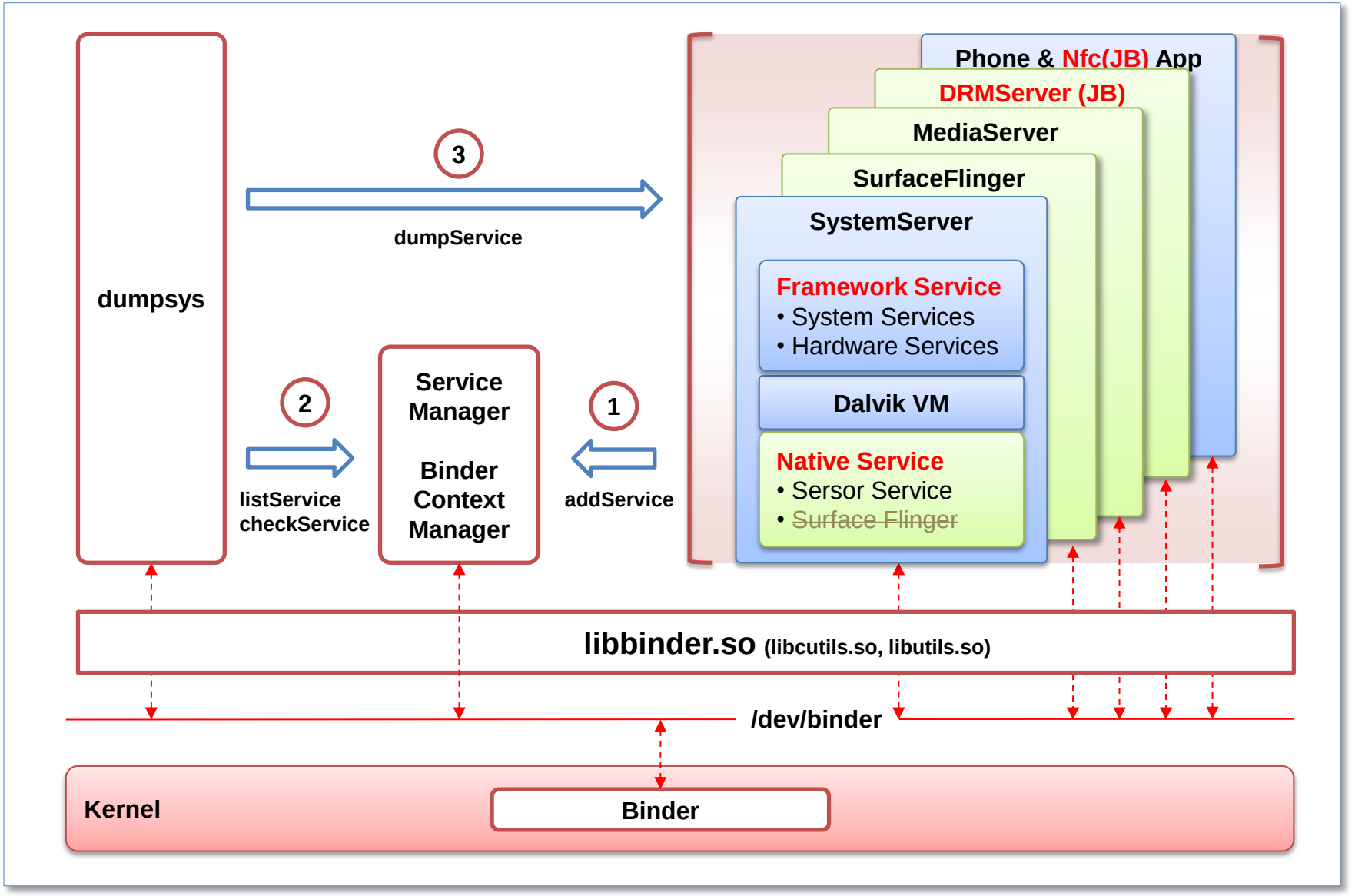
Performance Analysis Knowledge : systrace



Performance Analysis Knowledge : systrace



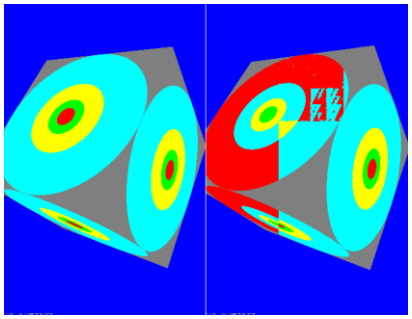
Performance Analysis Knowledge : dumpsys



Performance Analysis Knowledge : dumpsys

SurfaceFlinger accessibility account **activity** alarm android.security.keystore appops appwidget **assetatlas** audio backup battery batterypropreg batterystats clipboard commontime_management connectivity consumer_ir content country_detector cpuinfo	dbinfo device_policy devicestoragemonitor diskstats **display** dreams drm.drmManager dropbox entropy **gfxinfo** hardware **input** input_method iphonesubinfo isms location lock_settings media.audio_flinger media.audio_policy media.camera	media.player media_router **meminfo** mount netpolicy netstats network_management notification package permission phone power print procstats samplingprofiler scheduling_policy search sensorservice serial servicediscovery simphonebook	statusbar telephony.registry textservices uimode updatelock usagestats usb user vibrator wallpaper wifi wifip2p **window**

OpenGL|ES Context Management in MT Environment



fencing on vs. fencing off

OpenGL ES 3.0 sync objects

- GLsync glFenceSync(GLenum condition, GLbitfield flags);
- GLenum glClientWaitSync(GLsync sync, GLbitfield flags, GLuint64 timeout);
- void glWaitSync(GLsync sync, GLbitfield flags, GLuint64 timeout);
- void glDeleteSync(GLsync sync);
- void glGetSynciv(GLsync sync, GLenum pname, GLsizei bufSize, GLsizei *length, GLint *values);

OpenGL ES Context Management in Multi-threaded Environment

Working in OpenGL ES within a multi-threaded environment requires some minor additional work in your application. In order to work with multiple threads in OpenGL ES, we must follow these rules:

- **Only one rendering context can be current per thread.**
- **A given context can be current to only one thread.**

In the case of multiple threads sharing a single rendering context, we must unbind the context from the thread where it is current before making it current to any other thread. The order of operations is as follows:

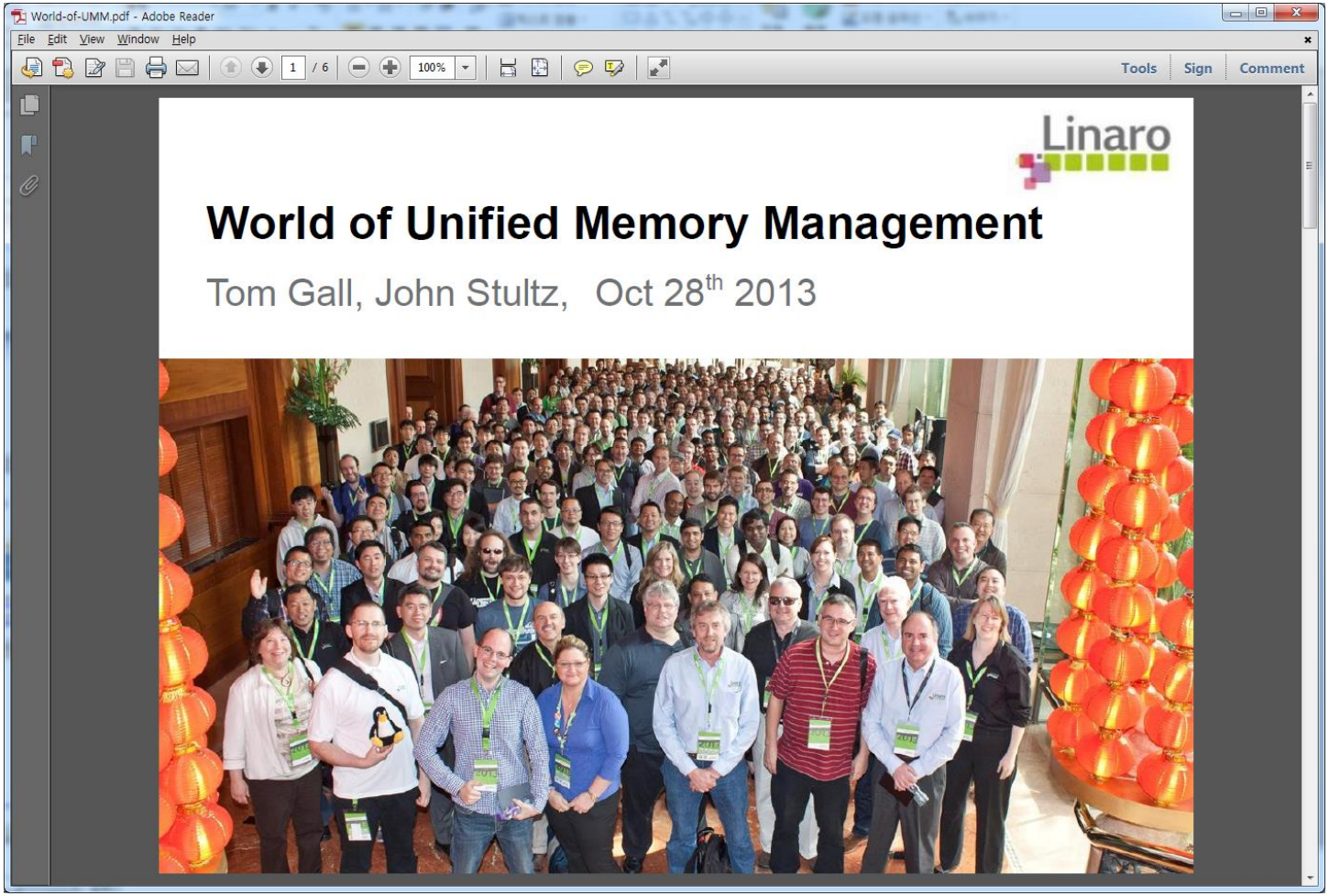
Thread 1	Thread 2

eglMakeCurrent(display, surface, surface, context);
Perform OpenGL ES operations
eglMakeCurrent(display, EGL_NO_SURFACE, EGL_NO_SURFACE, EGL_NO_CONTEXT);	

	eglMakeCurrent(display, surface, surface, context); Perform OpenGL ES operations

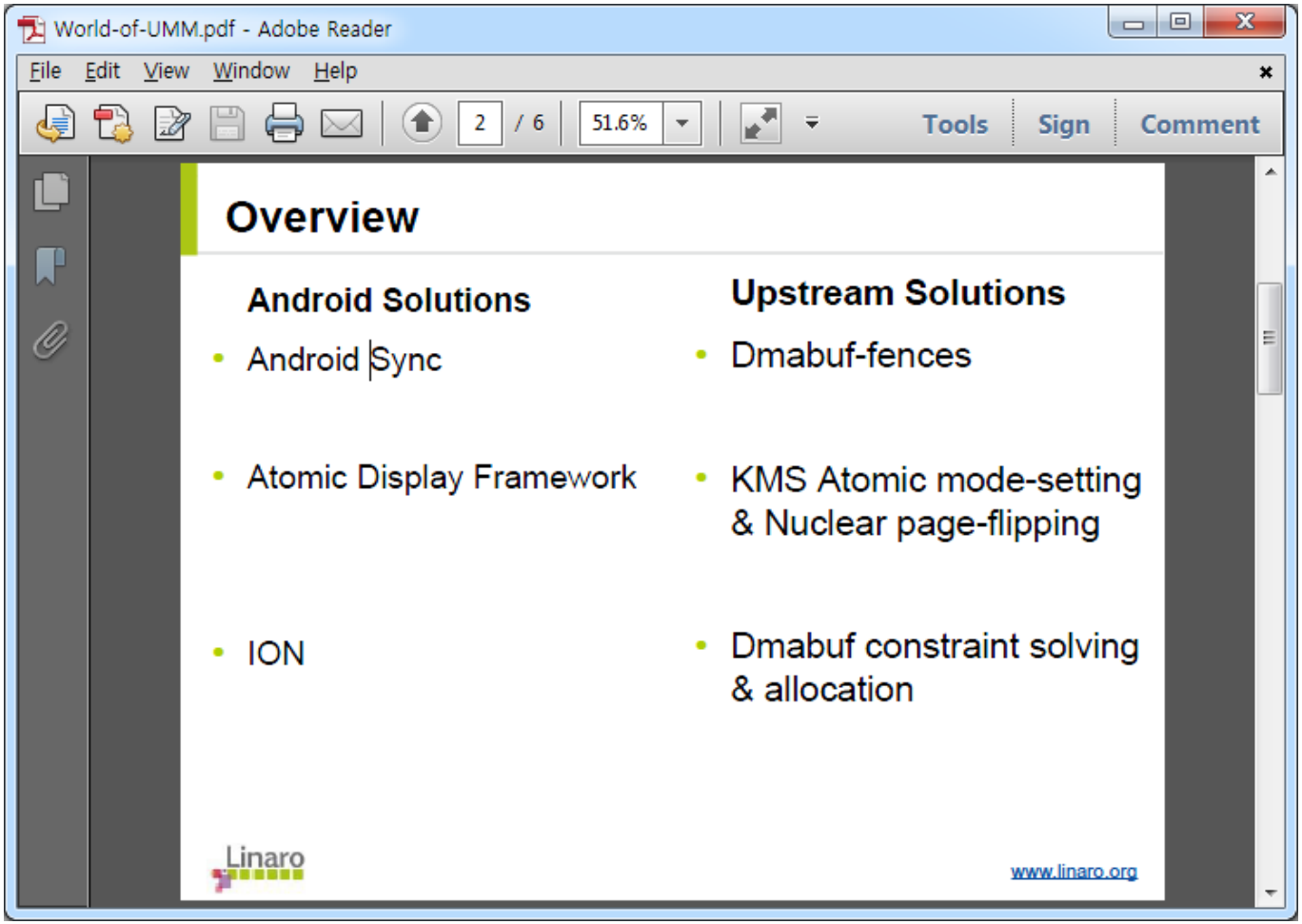
Source : http://malideveloper.arm.com/downloads/deved/tutorial/SDK/android/thread_sync.html

Unified Memory Management



Source : http://people.linaro.org/linaro-connect/lcu13/presentations/World-of-UMM.pdf
Ref : http://www.slideshare.net/linaroorg/android-graphicsupstreamingv2

Unified Memory Management



Source : http://people.linaro.org/linaro-connect/lcu13/presentations/World-of-UMM.pdf
Ref : http://www.slideshare.net/linaroorg/android-graphicsupstreamingv2

1/4 Featured Sessions,

Kandroid S/W Fundamentals Study Group

Q & A

www.kandroid.org 운영자

양정수 (yangjeongsoo@gmail.com)