

# Introduction to Android Window System

Chia-I Wu  
olv@0xlab.org

May 15, 2009

## Building Blocks

Overview

Interested Components

## Under the Hood

Random Topics

## Get Dirty

Development

Code

## Q & A

# Outline

## Building Blocks

Overview

Interested Components

## Under the Hood

Random Topics

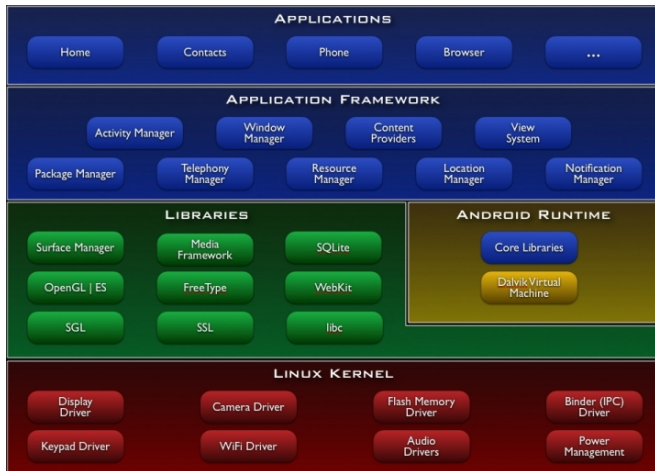
## Get Dirty

Development

Code

## Q & A

# System Architecture



# System Architecture



# System Architecture



# Building Blocks

There are more, but we focus on

- ▶ SurfaceManager
- ▶ WindowManager
- ▶ ActivityManager

# SurfaceManager

- ▶ `frameworks/base/libs/surfaceflinger/`
- ▶ a.k.a SurfaceFlinger
- ▶ Allocate surfaces. Backed by `ashmem/pmем/?`
- ▶ Composite surfaces

# WindowManager

- ▶ `frameworks/base/services/java/com/android/server/WindowManagerService.java`
- ▶ About 9000 SLOC in one file. Poorly documented, bad namings, ...
- ▶ (Ask SurfaceManager to) create/layout surfaces on behalf of the clients
- ▶ Dispatch input events to clients
- ▶ Transition animation
- ▶ WindowManagerPolicy

# ActivityManager

- ▶ `frameworks/base/services/java/com/android/server/am/`
- ▶ Manage lifecycles of activities
- ▶ Manage stacking of activities
- ▶ Dispatch intents
- ▶ Spawn processes

# Confusions

- ▶ An activity has one or more windows (e.g. dialogs)
- ▶ A window has one or more surfaces (e.g. surface views)
- ▶ However, in window manager, a window is called a session
- ▶ A surface is called a window
- ▶ And an activity becomes roughly a token

# Special Keys

- ▶ HOME key
- ▶ BACK key

# Outline

## Building Blocks

Overview

Interested Components

## Under the Hood

Random Topics

## Get Dirty

Development

Code

## Q & A

# Process View

- ▶ SurfaceManager, WindowManager, and SurfaceManager are threads of a single process (system\_server)
- ▶ Every application is usually a process of itself

# Zygote

- ▶ Is a process started on system initialization
- ▶ Preloads java classes and resources
- ▶ Forks system\_server
- ▶ Listens silently on `/dev/socket/zygote`

# Binder

- ▶ Early in the lifetime of an application process, thread(s) are created and blocked on `/dev/binder`
- ▶ Binder is used mainly for RPC
- ▶ Fragile

# Outline

## Building Blocks

Overview

Interested Components

## Under the Hood

Random Topics

## Get Dirty

Development

Code

## Q & A

# Build System

- ▶ `build/core/core/build-system.html`
- ▶ `. build/envsetup.sh`
- ▶ `showcommands`
- ▶ `export ANDROID_JAVA_HOME` if non-standard

# adb

- ▶ ADBHOST for transport over TCP/IP
- ▶ kill-server
- ▶ remount
- ▶ pull/push
- ▶ logcat
- ▶ shell

# hierarchyviewer

- ▶ Display view hierarchy
- ▶ Display view
- ▶ Invalidate/Relayout

# Graphics: Memory Management

- ▶ SurfaceFlinger has a SurfaceHeapManager
- ▶ Every client has a MemoryDealer, as returned by SurfaceHeapManager
- ▶ Every surface of a client also has dealer(s), from client or GPU

## Graphics: Memory Management cont.

- ▶ A dealer consists of a heap and an allocator
- ▶ A heap represents a sharable big chunk of memory
- ▶ An allocator is an algorithm
- ▶ Small chunks of memory from the heap are returned

## Graphics: Memory Management cont.

Real flow

- ▶ A client asks for a new surface, createSurface

## Graphics: Memory Management cont.

### Real flow

- ▶ A client asks for a new surface, `createSurface`
- ▶ `createSurface` calls `createNormalSurfaceLocked`

## Graphics: Memory Management cont.

### Real flow

- ▶ A client asks for a new surface, `createSurface`
- ▶ `createSurface` calls `createNormalSurfaceLocked`
- ▶ A layer is created and `setBuffers` is called to allocate buffers

## Graphics: Memory Management cont.

### Real flow

- ▶ A client asks for a new surface, `createSurface`
- ▶ `createSurface` calls `createNormalSurfaceLocked`
- ▶ A layer is created and `setBuffers` is called to allocate buffers
- ▶ Two dealers are created from client, one for front buffer and one for back buffer

## Graphics: Memory Management cont.

### Real flow

- ▶ A client asks for a new surface, `createSurface`
- ▶ `createSurface` calls `createNormalSurfaceLocked`
- ▶ A layer is created and `setBuffers` is called to allocate buffers
- ▶ Two dealers are created from client, one for front buffer and one for back buffer
- ▶ Two `LayerBitmaps` are created, initialized with the two dealers

## Graphics: Memory Management cont.

### Real flow

- ▶ A client asks for a new surface, `createSurface`
- ▶ `createSurface` calls `createNormalSurfaceLocked`
- ▶ A layer is created and `setBuffers` is called to allocate buffers
- ▶ Two dealers are created from client, one for front buffer and one for back buffer
- ▶ Two `LayerBitmaps` are created, initialized with the two dealers
- ▶ Heaps of dealers along with info about the layer are returned

# Hello World

- ▶ <http://people.debian.org.tw/~olv/surfaceflinger/demo.tar.gz>

# Many Buffers

- ▶ Surface is double buffered
- ▶ EGLDisplaySurface is double buffered
- ▶ Same technique; Different code pathes, different purposes

# Double Buffering

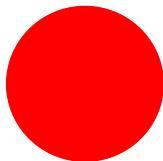
- ▶ Software v.s. Hardware
- ▶ Memory copy
- ▶ Page flipping

## Dirty Region

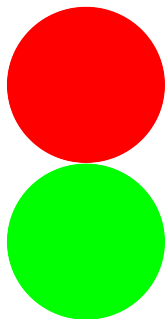
- ▶ Associate buffers with dirty regions
- ▶ Copy back

# Frame 0

# Frame 1



## Frame 2



# Outline

## Building Blocks

Overview

Interested Components

## Under the Hood

Random Topics

## Get Dirty

Development

Code

## Q & A

# Q & A

▶ Questions?