Applied Computer Science Concepts in Android

Jim Huang (黃敬群)

Developer & Co-Founder, 0xlab

jserv@0xlab.org

Dec 31, 2010 / CSIE, NTU

Applied Computer Science **Concepts** in **Android**

Android = A complete operating system for mobile computing, open source'd

Rights to copy

http://0xlab.org/

contact@0xlab.org

creative commons
COMMONS DEED

Attribution – ShareAlike 3.0

Corrections, suggestions, contributions and translations are welcome!

Latest update:Dec 31, 2010

You are free

- to copy, distribute, display, and perform the work
- to make derivative works
- to make commercial use of the work

Under the following conditions

BY: **Attribution**. You must give the original author credit.

Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.

Your fair use and other rights are in no way affected by the above.

License text: http://creativecommons.org/licenses/by-sa/3.0/legalcode

Agenda

(1) Android Internals

(2) **Compiler**: 2D/3D Graphics

(3) **OS**: Telephony

(4) **Virtual Machine**: Database

In this presentation, the unaware or indirect applications of essential computer science concepts are dicussed as showcase.

Android Internals

... or, the low-level parts ...

Android Internals

- Android **Low-Level** system
  - Architecture View
  - Everything from “Zygote”
  - Key design concepts in Android
  - System components
- Hardware Abstraction Layer (**HAL**)
  - GPS, RIL, Graphics (2D/3D), ...

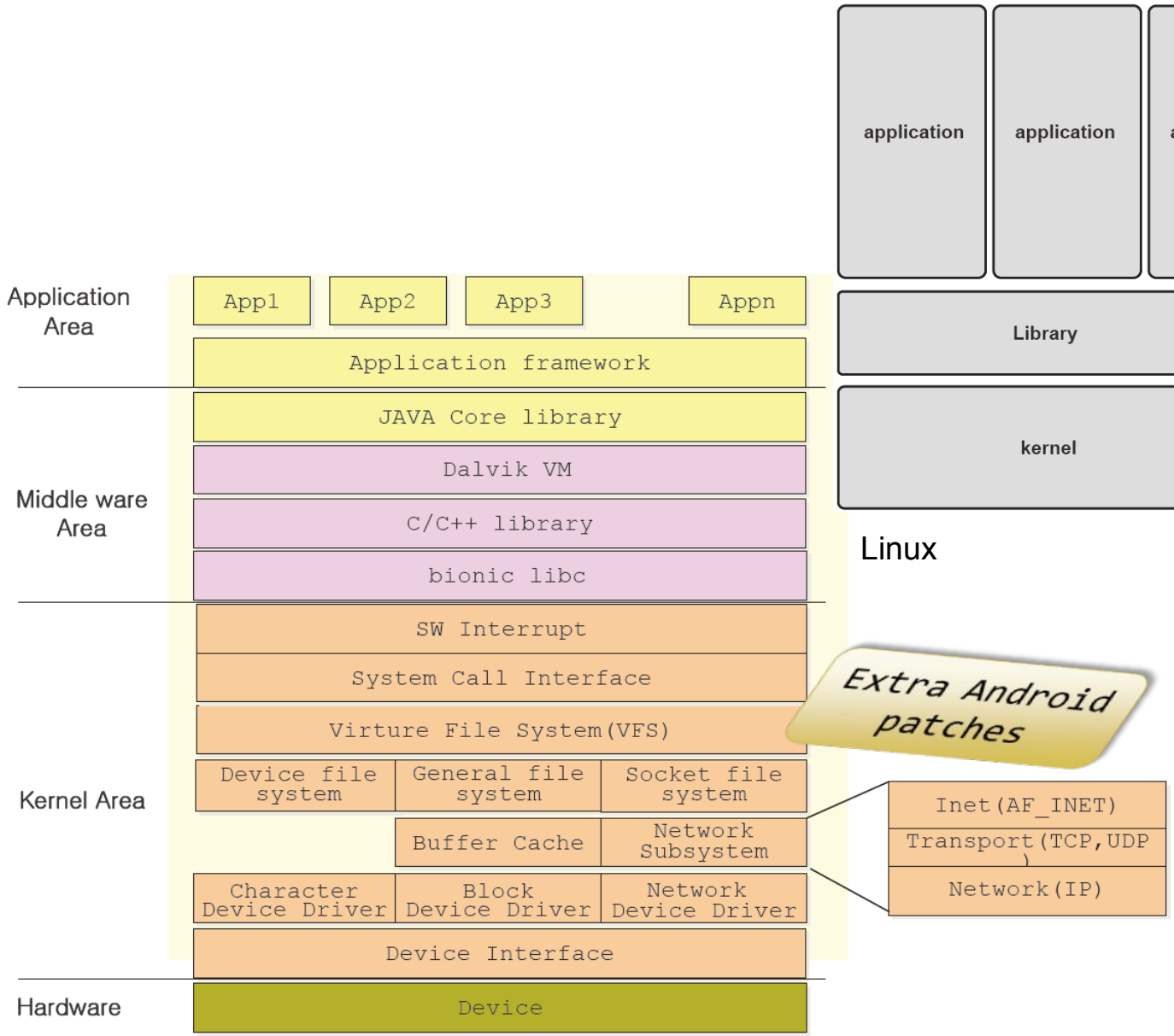
Android Low-level system

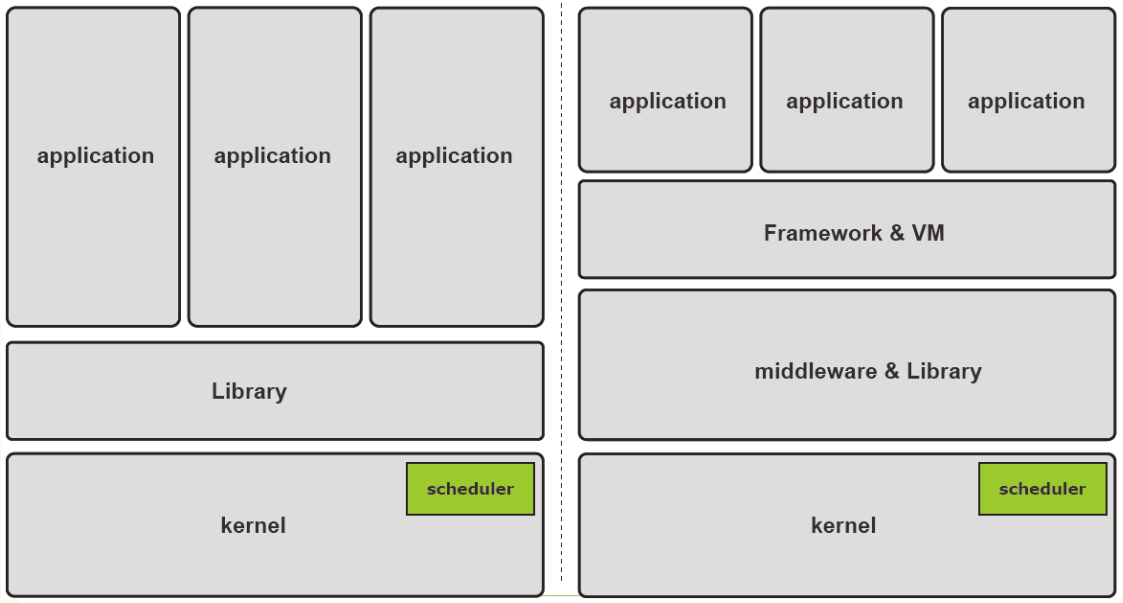
- Architecture
- Everything from “Zygote”
- Key design concepts in Android

Android Low-Level System

[Architecture]

GNU/Linux vs. Android



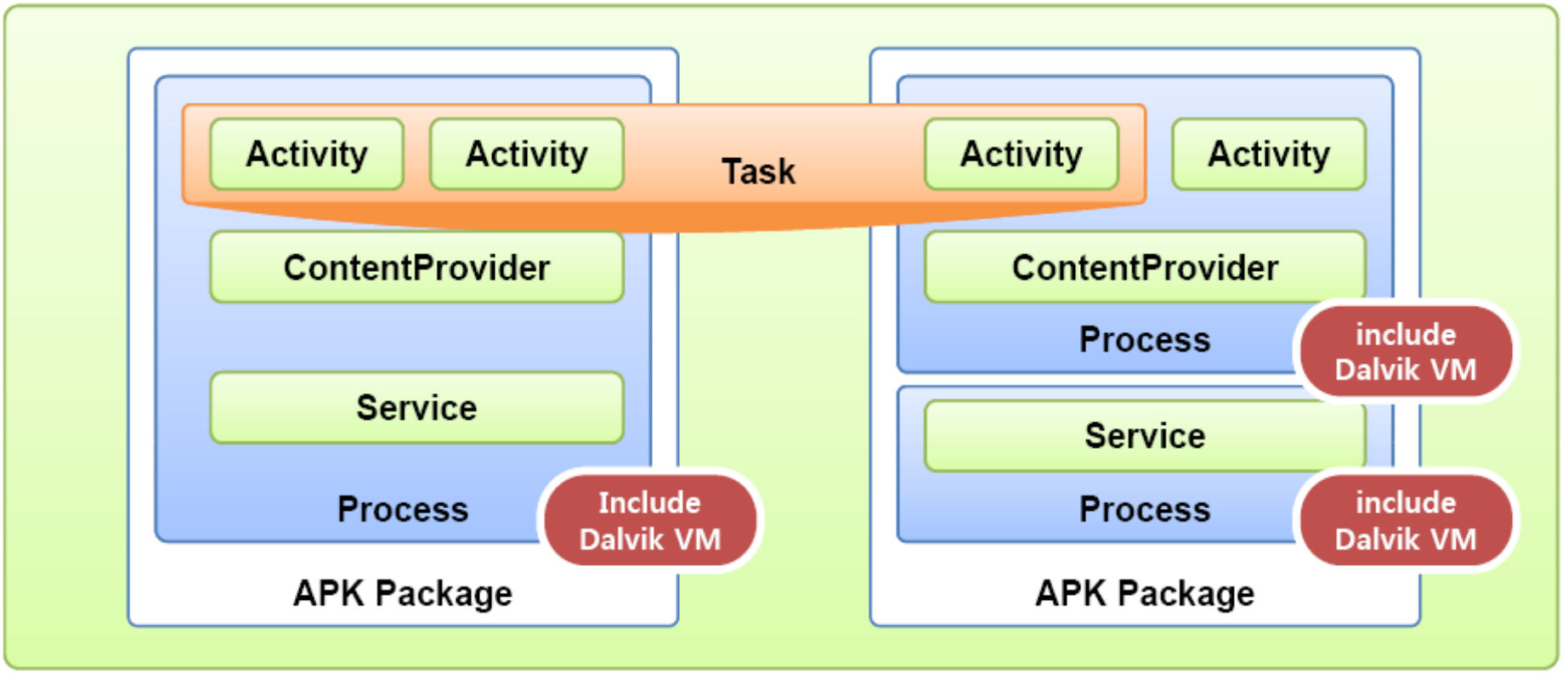


Android = (patched) Linux Kernel + special user-space

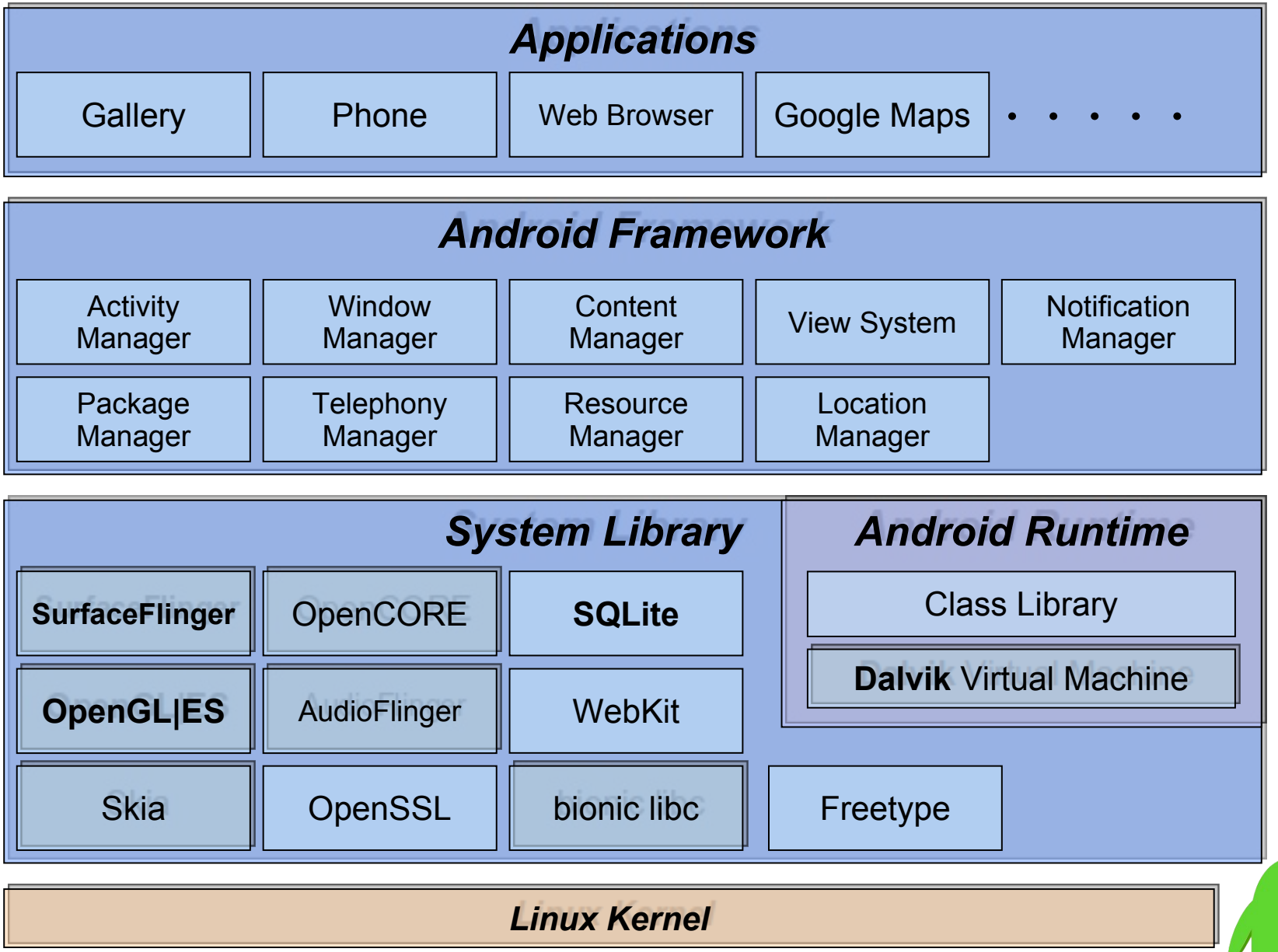
Application Building Block

- AndroidManifest.xml
- Activity [User Interaction] : (MIDlet)
- ContentProvider [Data Provider]
- Service [Service Provider]
- BroadcastReceiver (Push Registry)

Intent : Component Activation Method
- Explicit Method : Call Class
- Implicit Method : IntentFilter
  - Action, Data, Category
  - Declared at AndroidManifest.xml



Functional View



Android Low-Level System

[Everything from Zygote]

Android Boot Sequence

Low level initialization

High level perspective (Android specific)

ARM based
Target Board

Android bootloader
USB loader

Load Linux

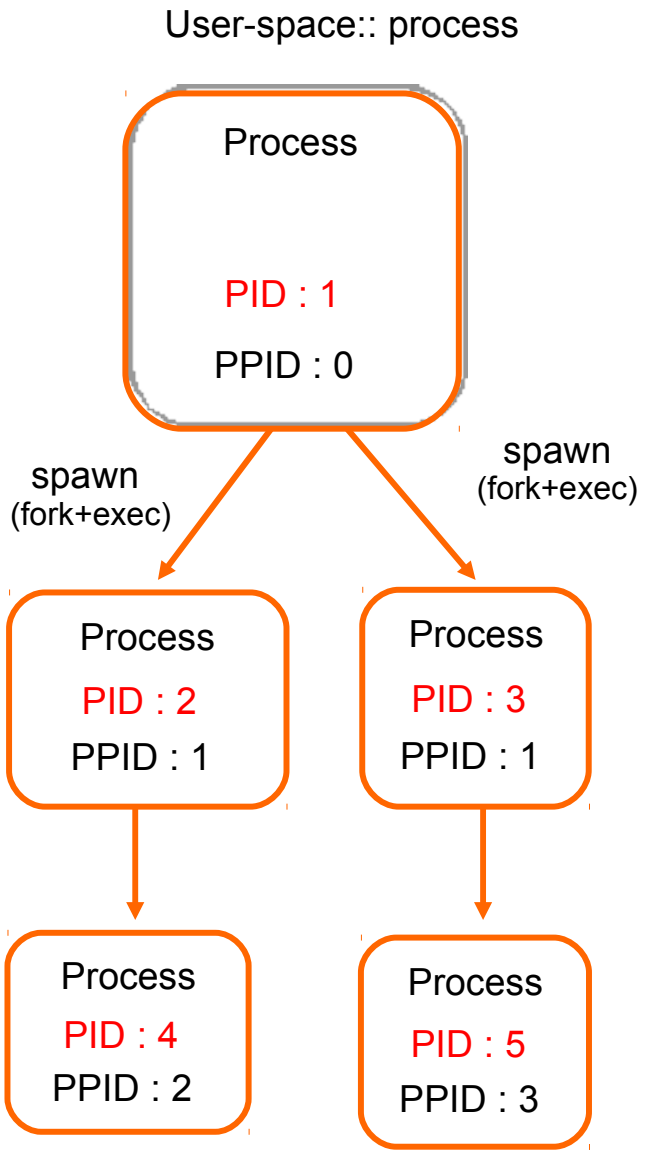
Start kernel
- /init/main.c

Initialization process
/etc/init.rc

Start Android Service

console
adbd
service manager
mountd
debuggerd
zygote
mediaserver
installd
flash_recovery

Process View




```
# ps
ps
USER     PID   PPID  VSIZE  RSS     WCHAN    PC         NAME
root      1     0     280    188     c008de04 0000c74c S /init
root      2     0     0      0       c004b334 00000000 S kthreadd
root      3     2     0      0       c003cf68 00000000 S ksoftirqd/0
root      4     2     0      0       c00486b8 00000000 S events/0
root      5     2     0      0       c00486b8 00000000 S khelper
root      10    2     0      0       c00486b8 00000000 S suspend
root      42    2     0      0       c00486b8 00000000 S kblockd/0
root      45    2     0      0       c00486b8 00000000 S cqueue
root      47    2     0      0       c016f13c 00000000 S kseriod
root      51    2     0      0       c00486b8 00000000 S kmmcd
root      95    2     0      0       c0065c7c 00000000 S pdflush
root      96    2     0      0       c0065c7c 00000000 S pdflush
root      97    2     0      0       c006990c 00000000 S kswapd0
root      99    2     0      0       c00486b8 00000000 S aio/0
root      267   2     0      0       c016c884 00000000 S mtdblockd
root      302   2     0      0       c00486b8 00000000 S rpciod/0
root      536   1     740    312     c0141bb0 afe0c1bc S /system/bin/sh
system    537   1     808    264     c01654b4 afe0c45c S /system/bin/servicemanager
root      538   1     836    364     c008e3f4 afe0c584 S /system/bin/vold
root      539   1     668    264     c0192c20 afe0cdec S /system/bin/debuggerd
radio     540   1     5392   684     ffffffff afe0cacc S /system/bin/rild
root      541   1     72416  20868   c008e3f4 afe0c584 S zygote
media     542   1     17720  3528    ffffffff afe0c45c S /system/bin/mediaserver
root      543   1     800    324     c01f3b04 afe0c1bc S /system/bin/installd
root      546   1     840    356     c00ae7b0 afe0d1dc S /system/bin/qemud
root      549   1     4432   180     ffffffff 0000e8f4 S /sbin/adbd
system    565   541   194228 28308   ffffffff afe0c45c S system_server
app_2     610   541   103148 18872   ffffffff afe0d3e4 S android.process.acore
radio     612   541   106840 17592   ffffffff afe0d3e4 S com.android.phone
app_15    636   541   94508  13884   ffffffff afe0d3e4 S com.android.mms
app_4     656   541   95368  13592   ffffffff afe0d3e4 S android.process.media
app_0     664   541   94288  12884   ffffffff afe0d3e4 S com.android.alarmclock
app_18    711   541   102160 12920   ffffffff afe0d3e4 S org.kandroid.sample
root      718   549   740    328     c003aa1c afe0d08c S /system/bin/sh
root      762   718   884    336     00000000 afe0c1bc R ps
```

1 2 3 4 5

fork()
Copy on Write

Role of Zygote

Zygote

App. Framework
- System Services
- Hardware Services

Dalvik VM

Native Server
- Audio Flinger
- Surface Flinger

System Server

libc

Home

Dalvik VM

Home

libc

daemons
- usbd
- adbd
- debuggerd
- rild

libc

Service Manager

runtime

libc

Zygote

libc

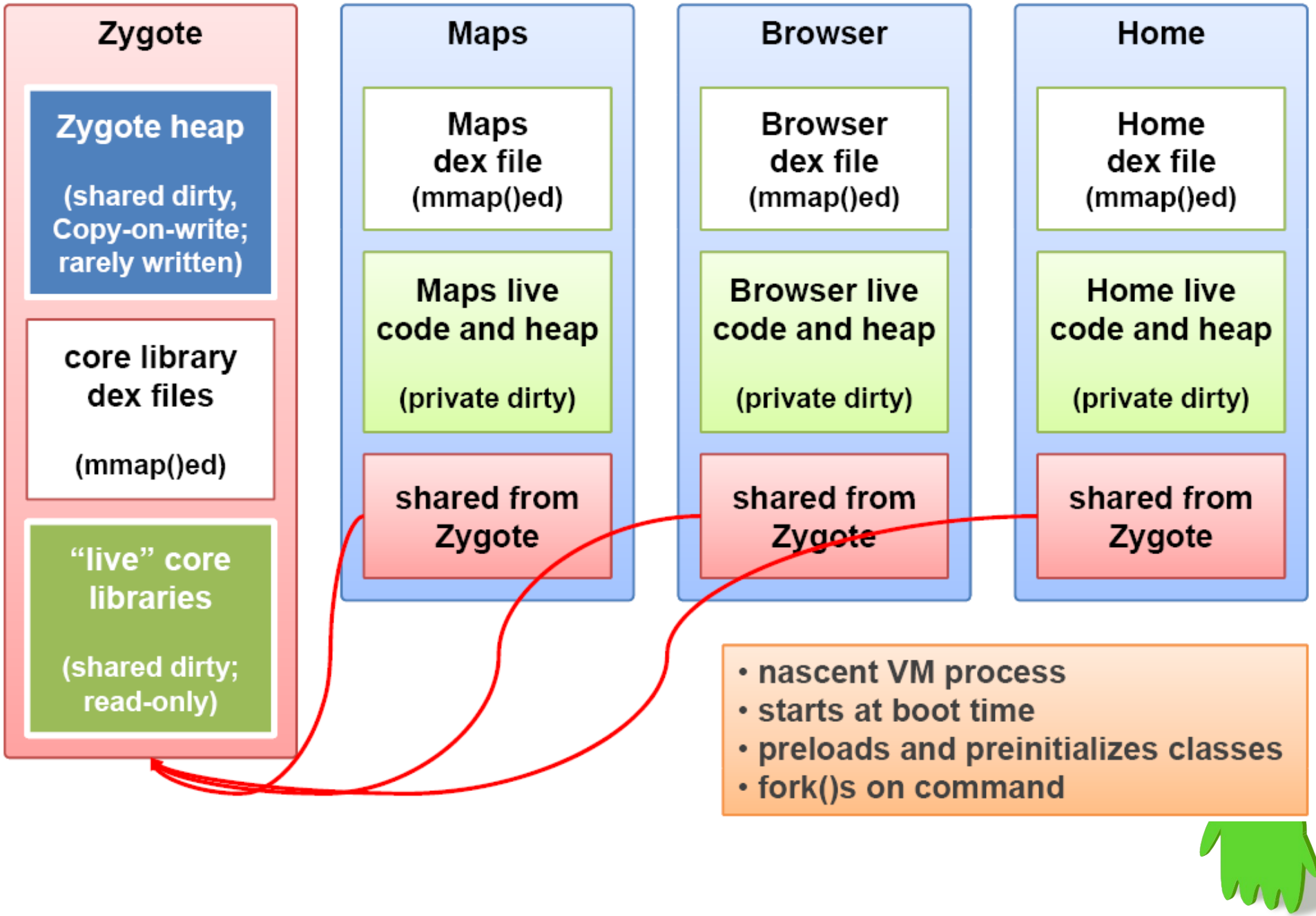
Init

Kernel

Binder Driver

Zygote is the parent process of the services in Android Framework

Everything spawn from Zygote



Android Low-Level System

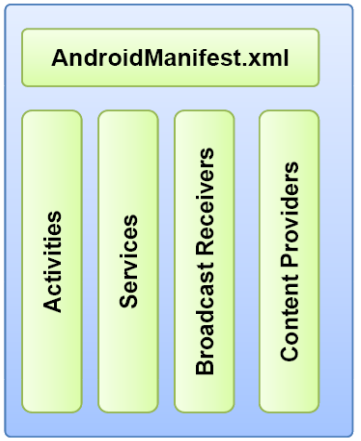
[Key Design Concepts]

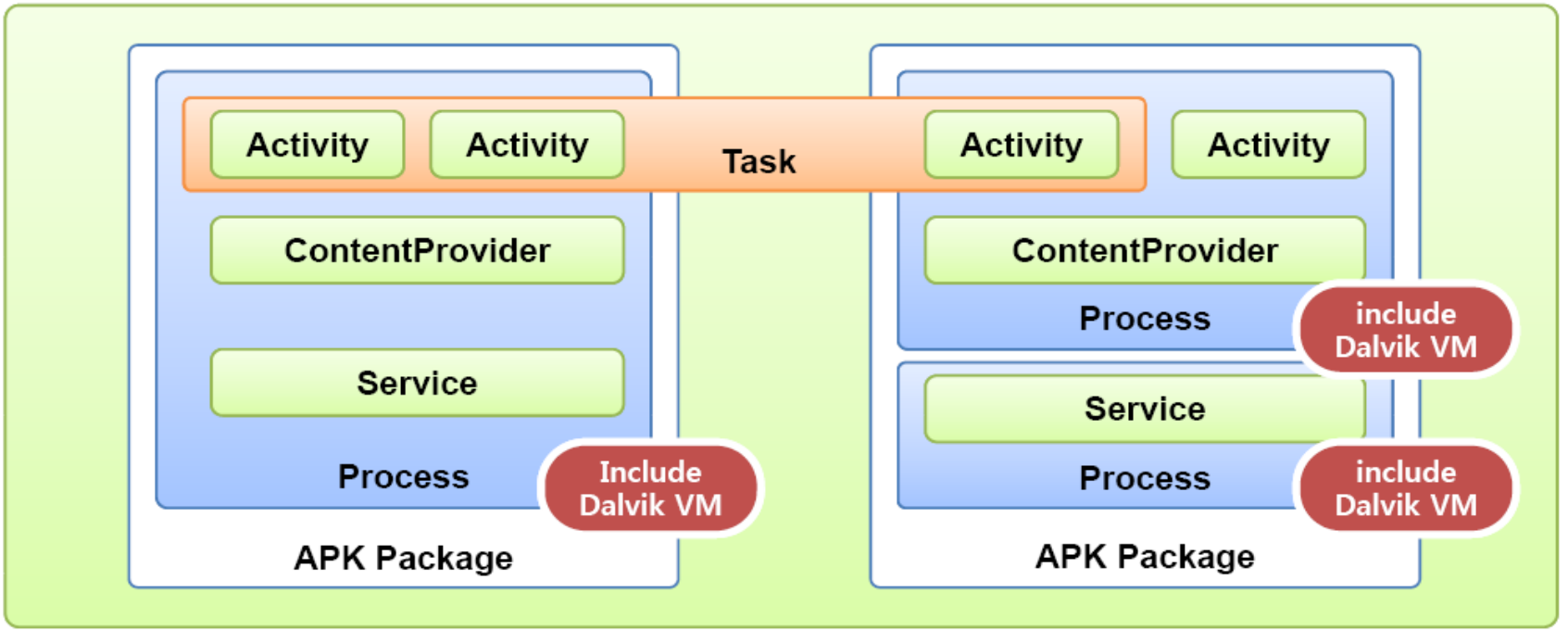
System Interaction

Application Building Block

- AndroidManifest.xml
- Activity [User Interaction] :
- ContentProvider [Data Provider]
- Service [Service Provider]
- BroadcastReceiver

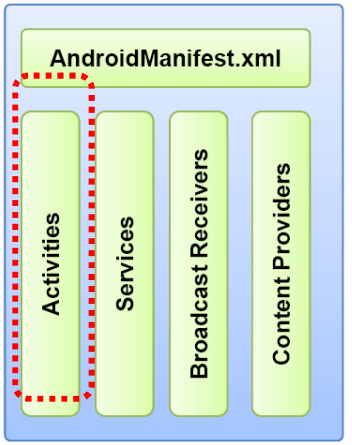
Intent : Component Activation Method
- Explicit Method : Call Class
- Implicit Method : IntentFilter
  - Action, Data, Category
  - Declared at AndroidManifest.xml

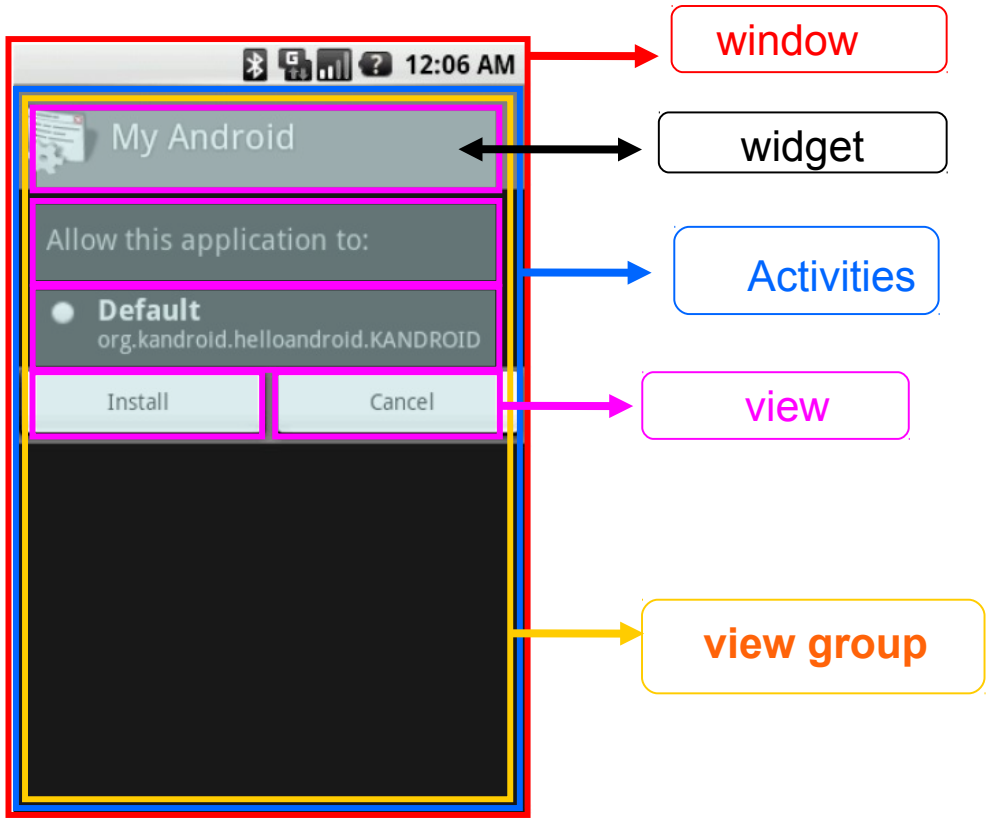


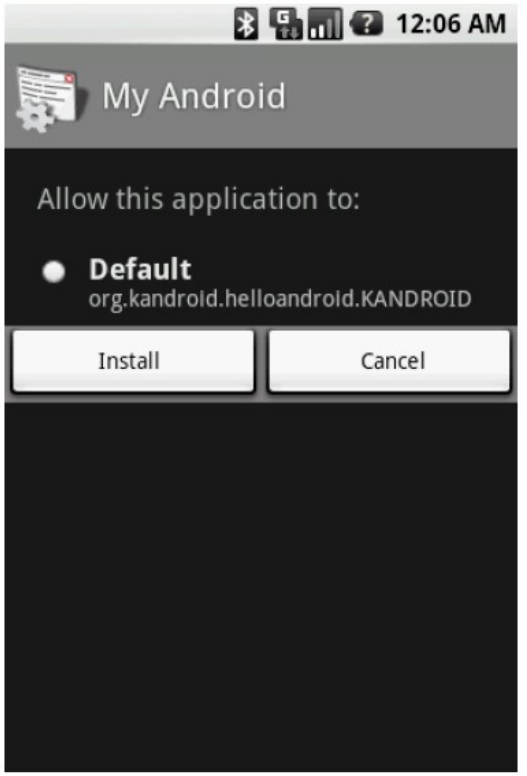


IPC: Inter-Process Communication

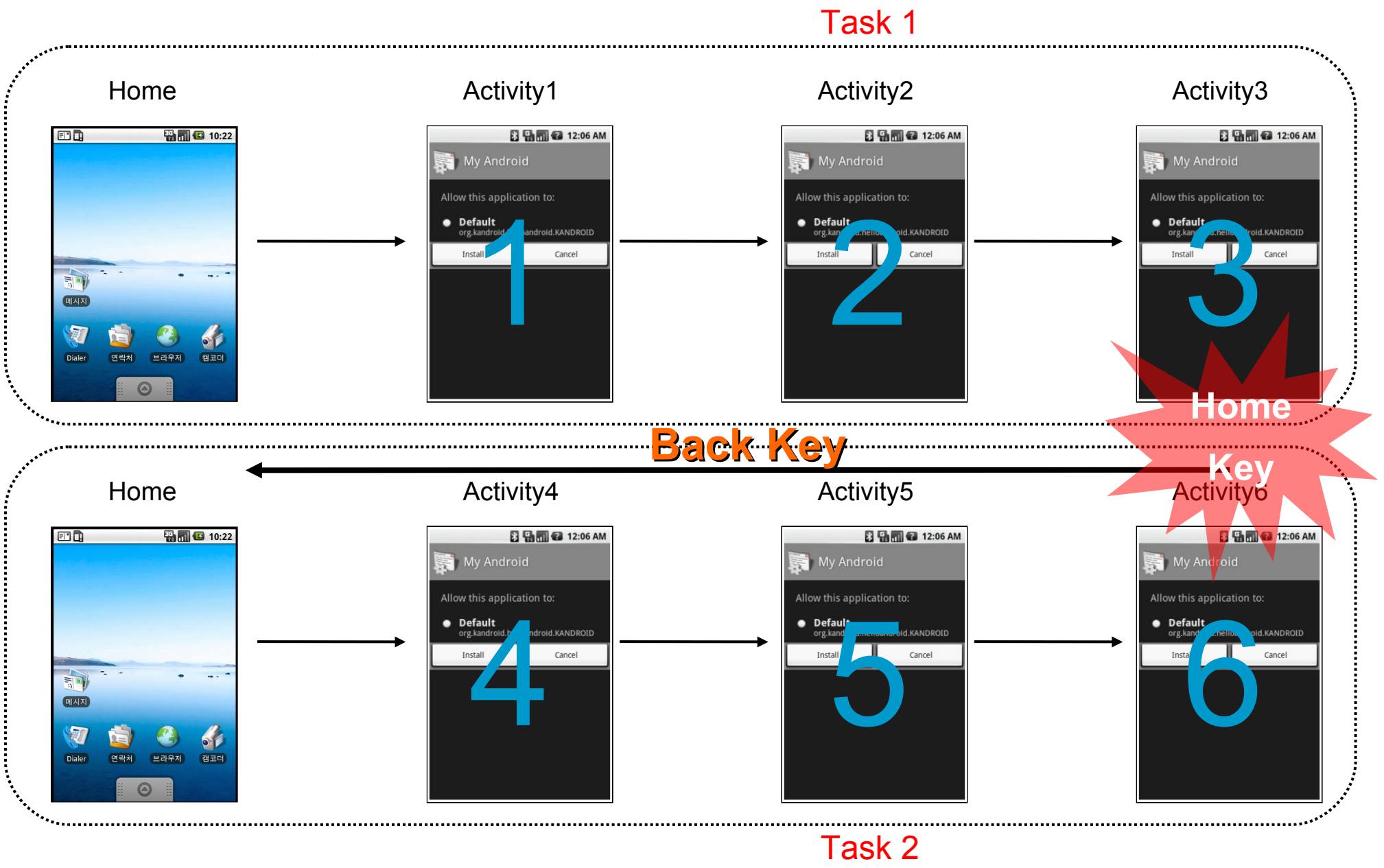
Activities


AndroidManifest.xml
Activities
Services
Broadcast Receivers
Content Providers


12:06 AM
My Android
Allow this application to:
Default
org.kandroid.helloandroid.KANDROID
Install
Cancel
window
widget
Activities
view
view group

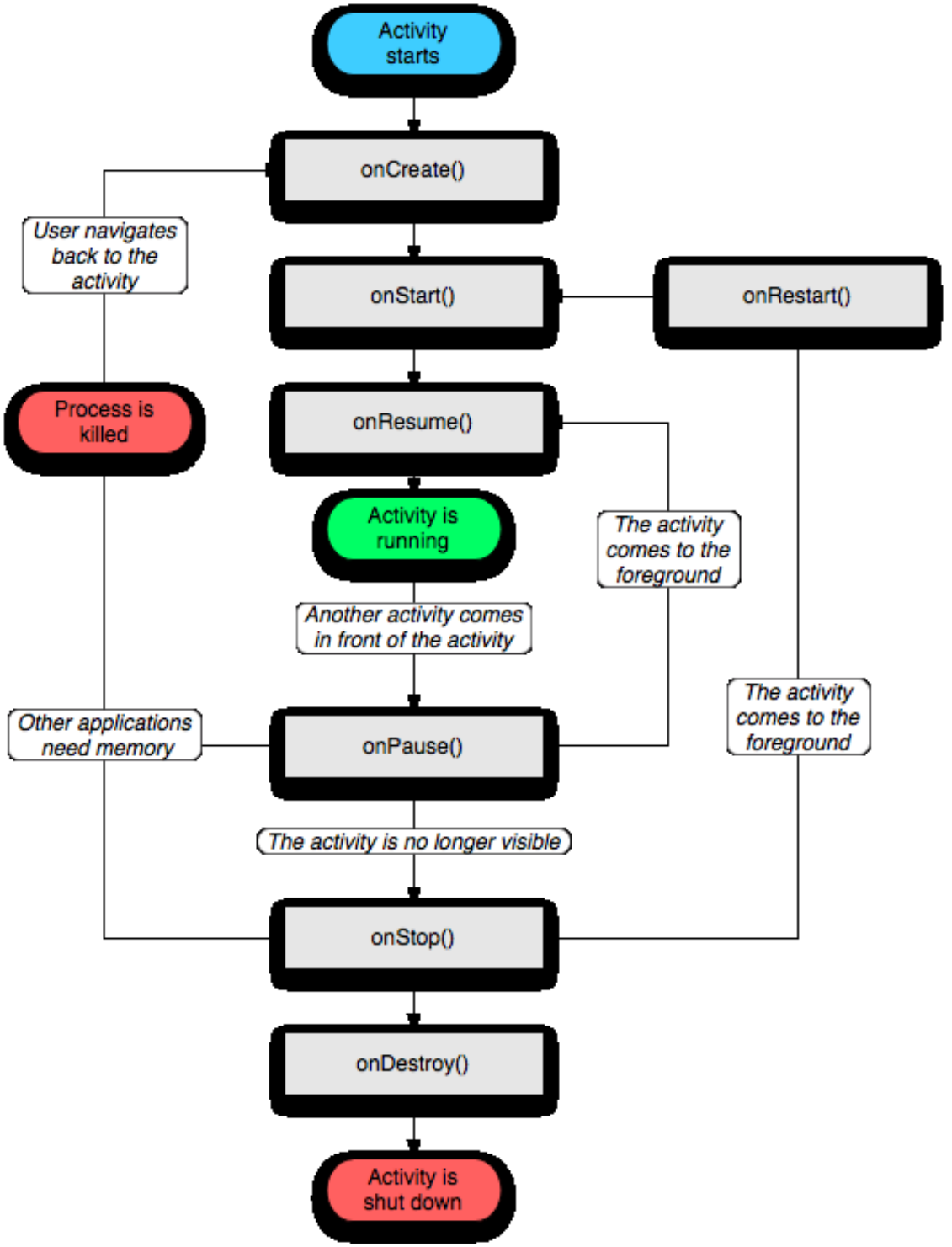

12:06 AM
My Android
Allow this application to:
Default
org.kandroid.helloandroid.KANDROID
Install
Cancel

Activity state transition

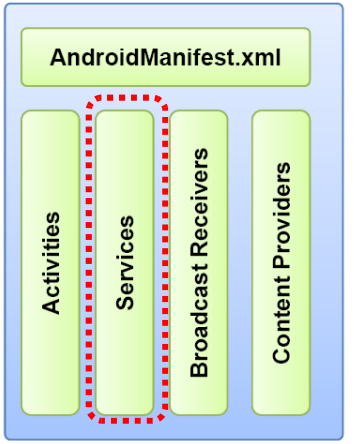


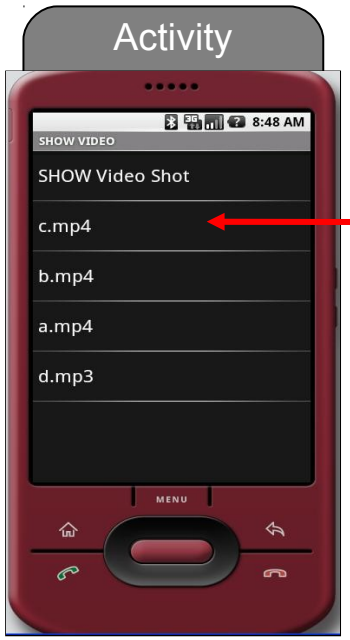
Activity life cycle

- Entire lifetime
- Visible lifetime
- Foreground lifetime

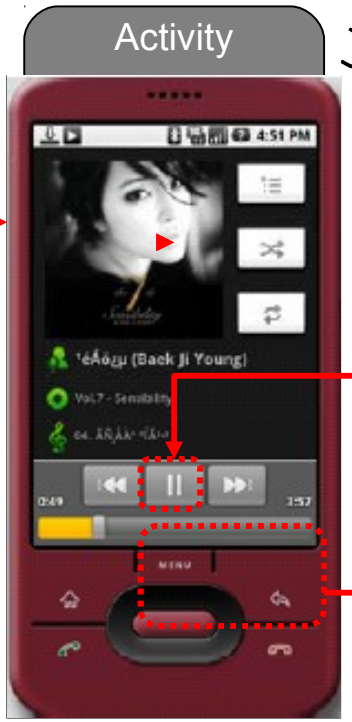


Services

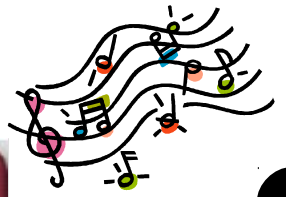




Activity1



Activity2



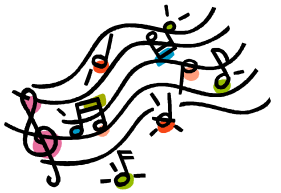
Service

Invoking specific service
(provided by Media server)

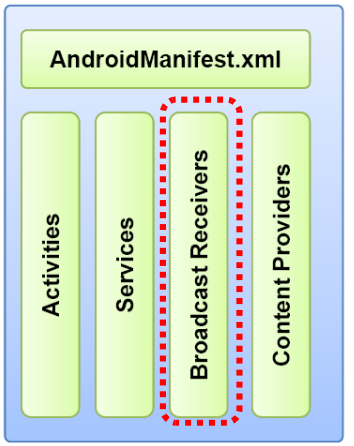
Home

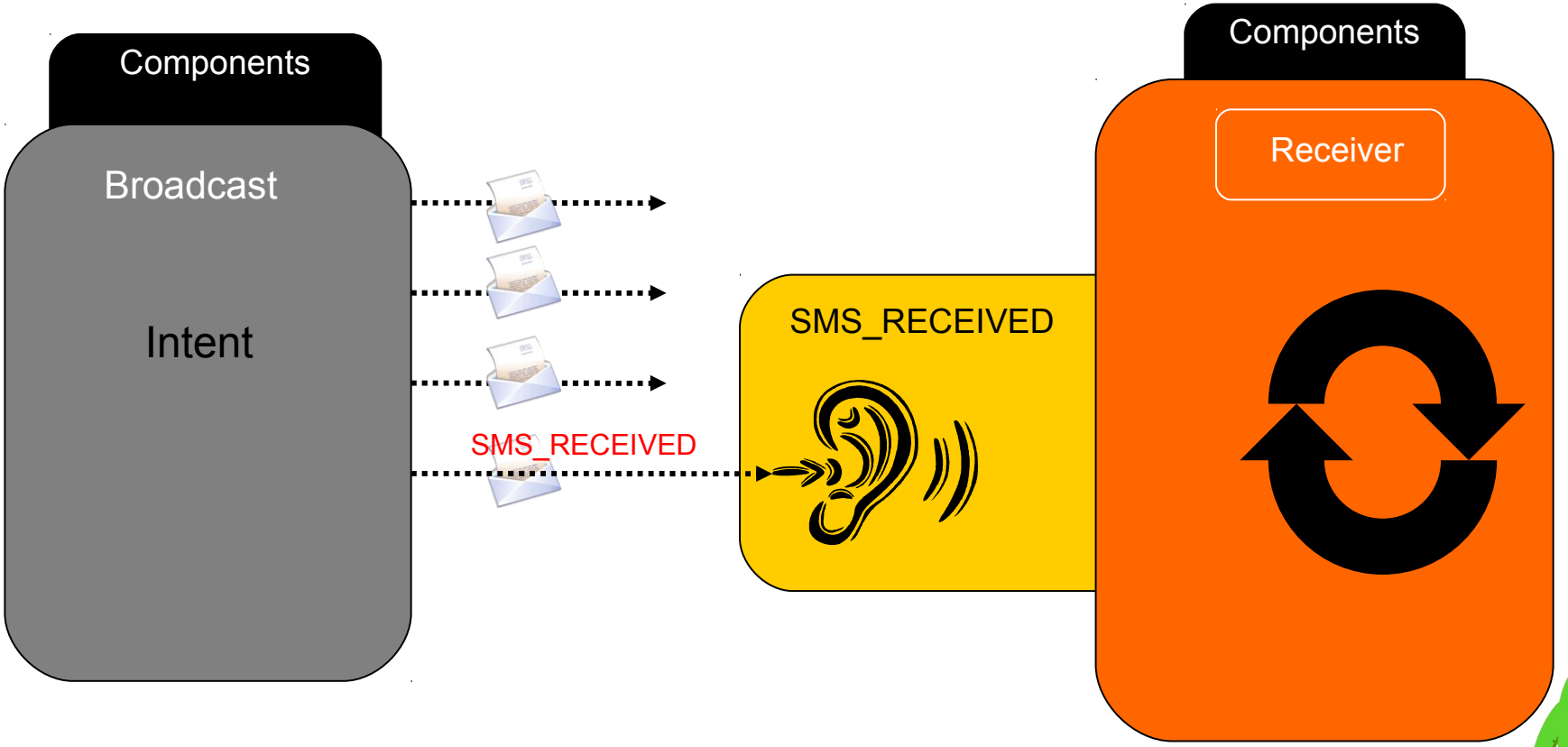
or

Back

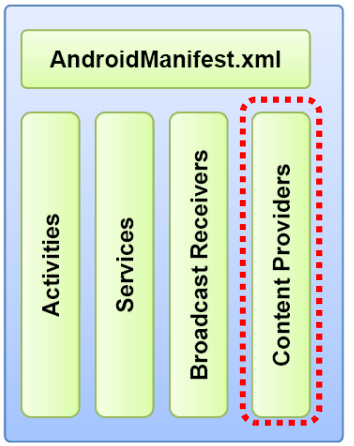


Broadcast Receivers





Content Providers



Components

Content Providers

Interface

Content Provider Class

Preferences

Files

SQLite(DB)

Network

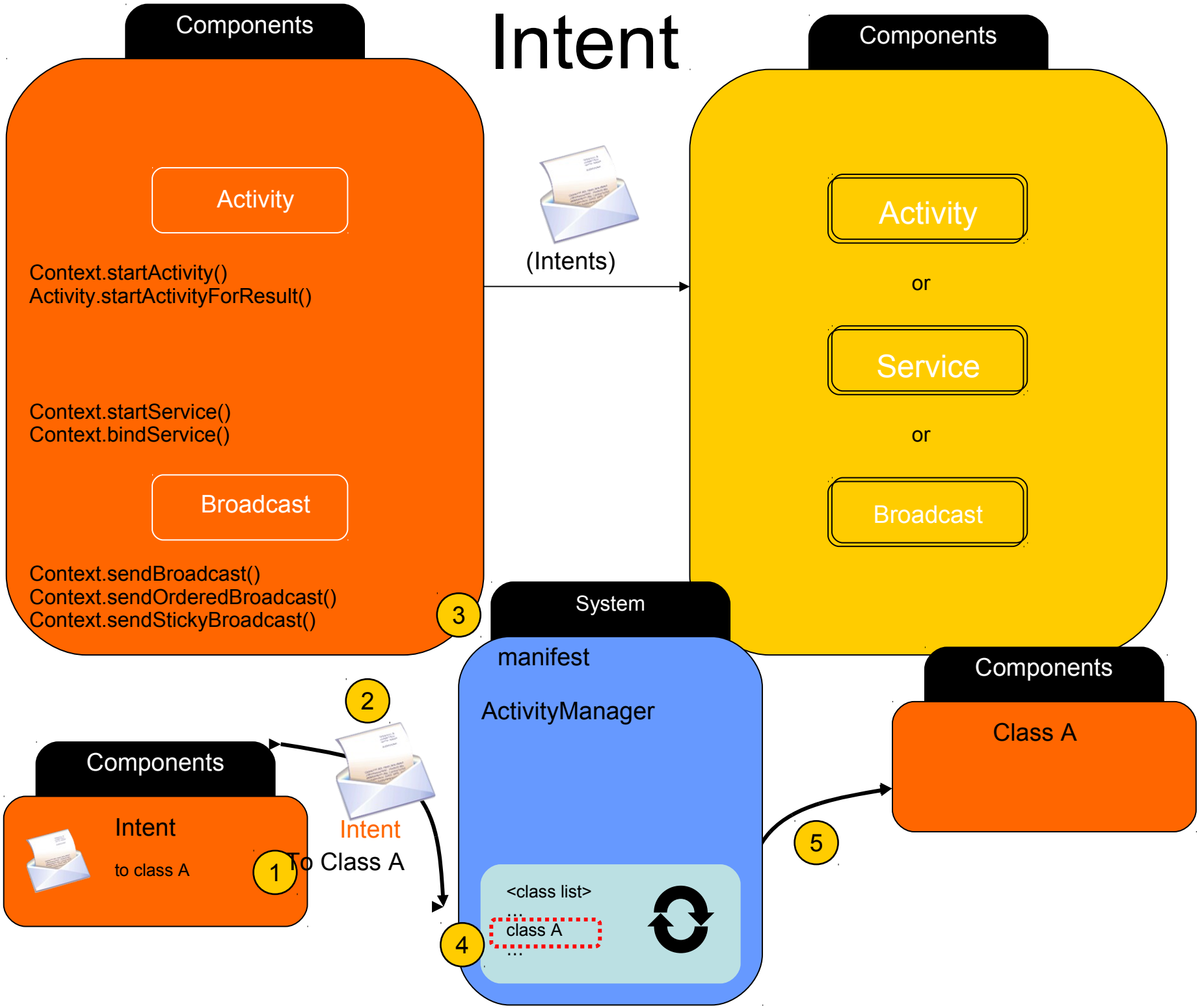
Components A

Content Resolver

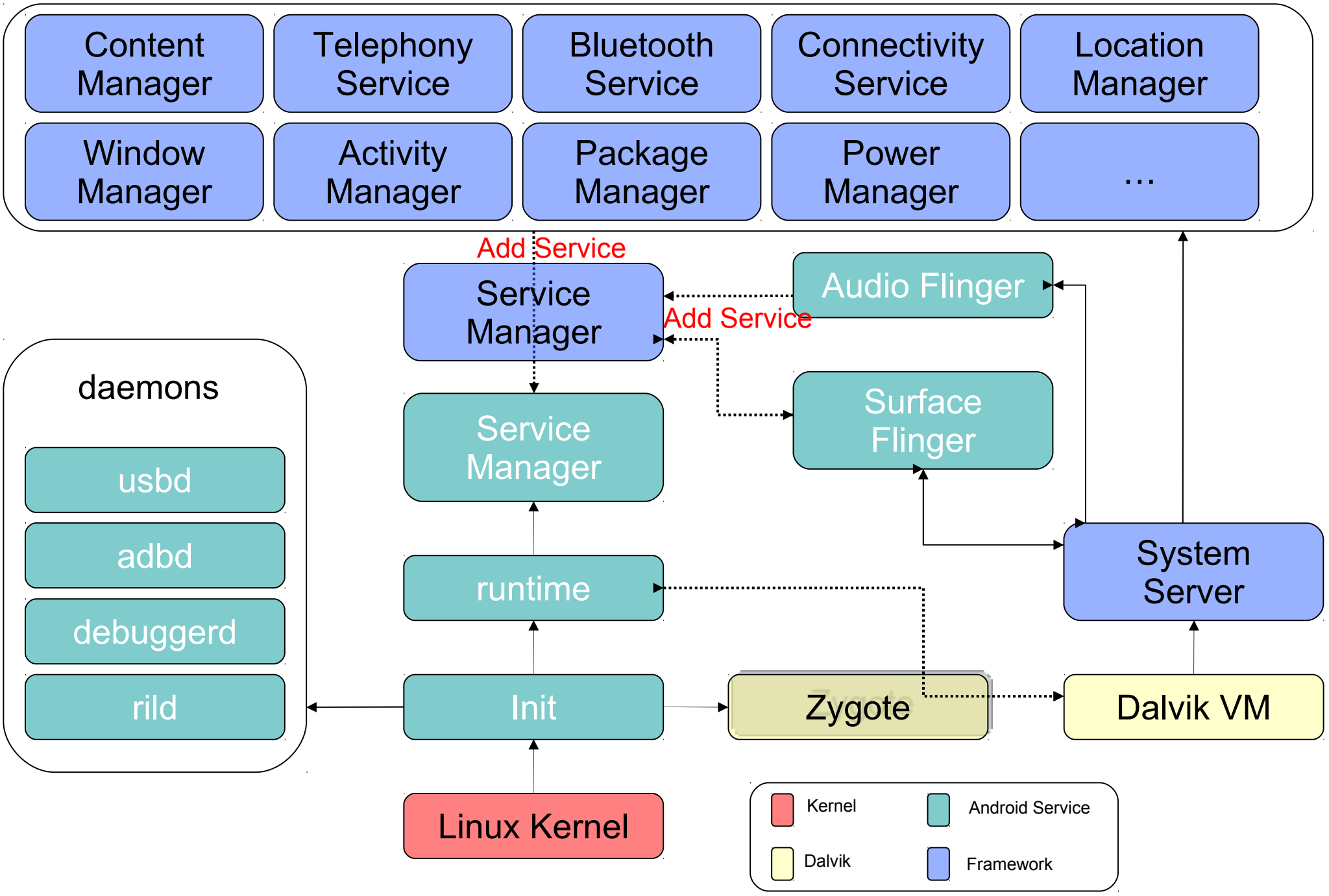
Components B

Content Resolver

Content Providers, Intents, Resolver

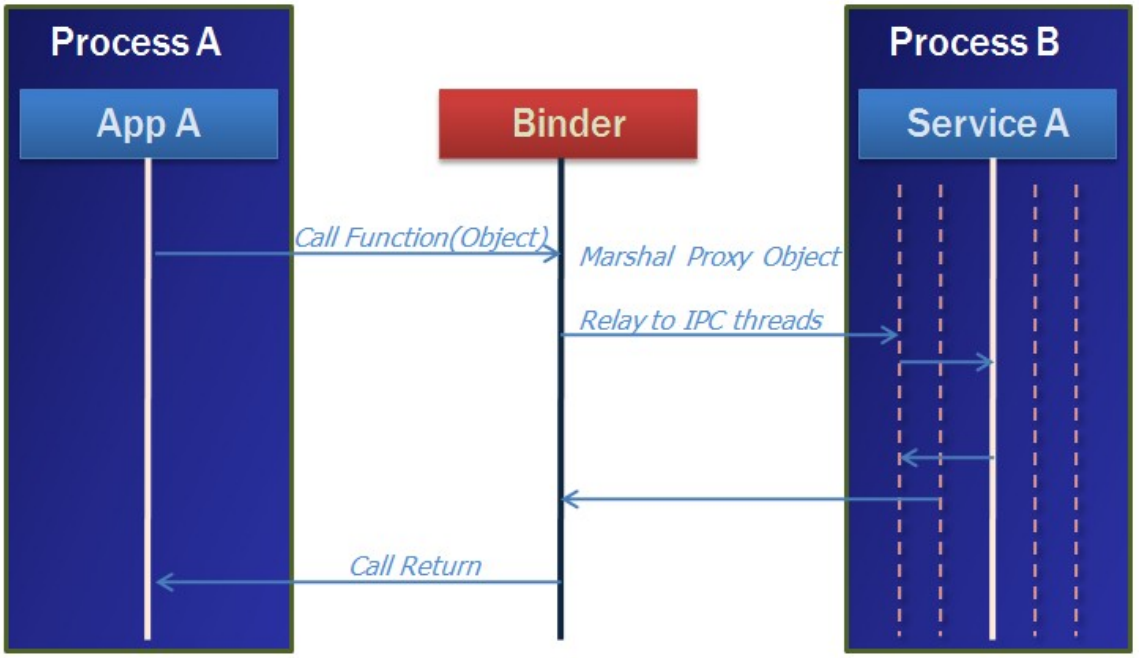

Intent
Components
Activity
Context.startActivity()
Activity.startActivityForResult()
Context.startService()
Context.bindService()
Broadcast
Context.sendBroadcast()
Context.sendOrderedBroadcast()
Context.sendStickyBroadcast()
(Intents)
Components
Activity
or
Service
or
Broadcast
System
manifest
ActivityManager
<class list>
...
class A
...
Components
Intent
to class A
Intent
To Class A
Components
Class A
1
2
3
4
5

Android Services in Action



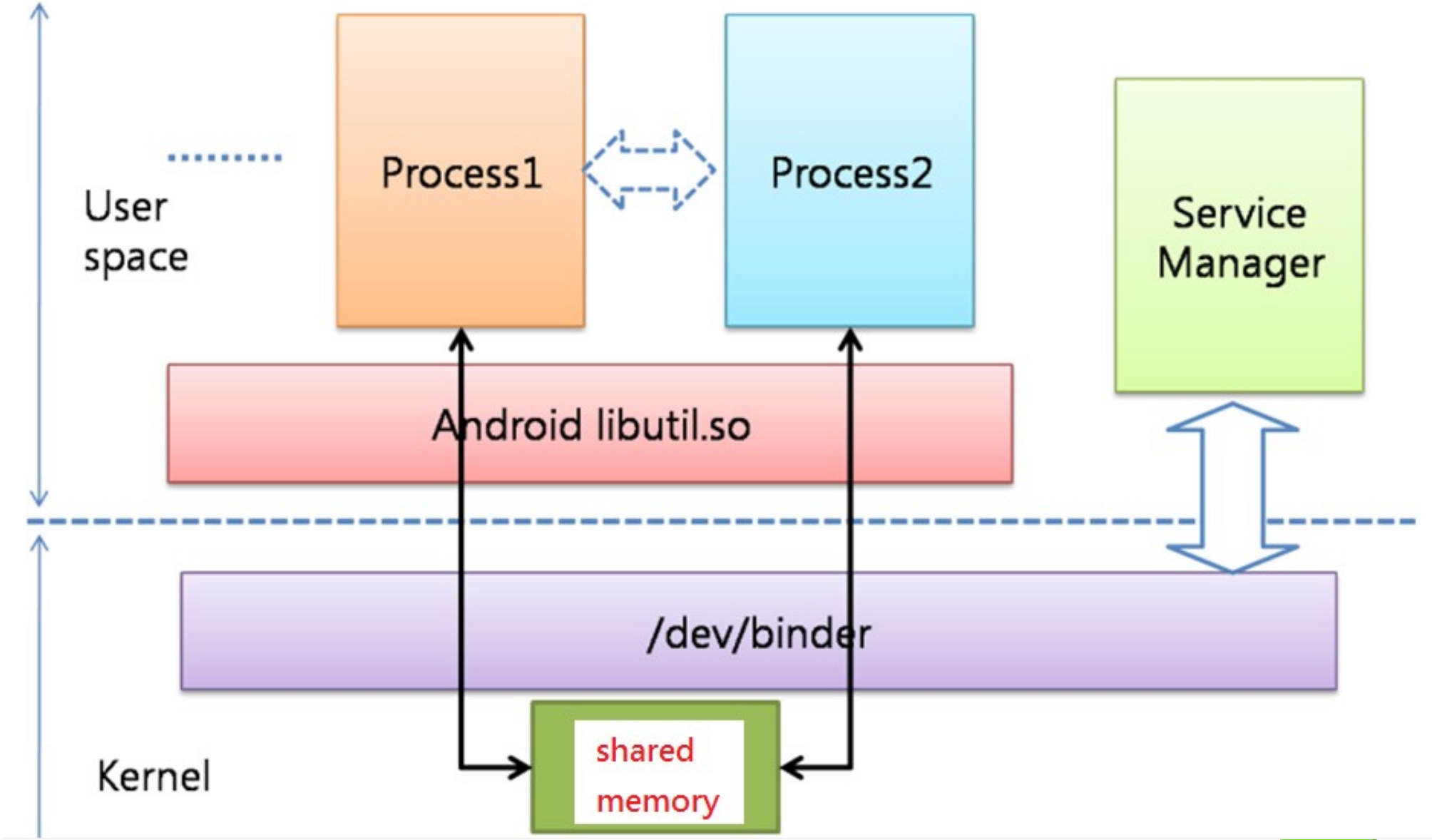
IPC: Binder(1)

- Binder in Action



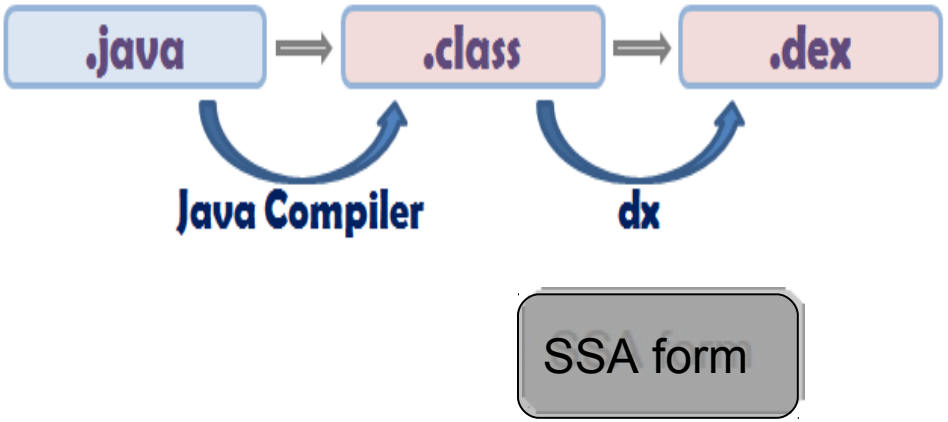
- ✓ A pool of threads is associated to each service application to process incoming IPC (Inter-Process Communication).
- ✓ Binder performs mapping of object between two processes.
- ✓ Binder uses an object reference as an address in a process's memory space.
- ✓ Synchronous call, reference couting

IPC: Binder(2)


User space
Process1
Process2
Service Manager
Android libutil.so
/dev/binder
shared memory
Kernel

Understanding

- Essential components
  - Compiler: Java
  - Virtual Machine: Dalvik
  - OS: Linux Kernel



- Anything else?

Use the Source, Luke!

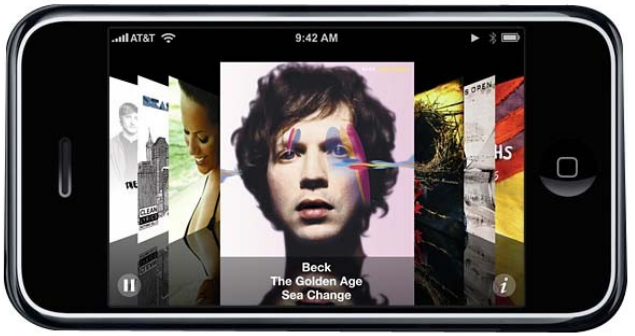
Many resources and tricks on the Internet find you will, but solutions to all technical issues only in the Source lie.

Thanks to LucasArts

Compilers

Applied in 2D/3D Graphics

OpenGL Anywhere



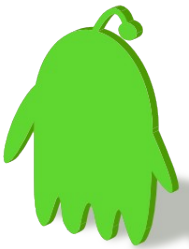
ARM® Cortex™- A8 with NEON VFP

PowerVR SGX Graphics

C64x+ DSP

Other Peripherals

Display Subsystem



OpenGL Graphics Pipeline

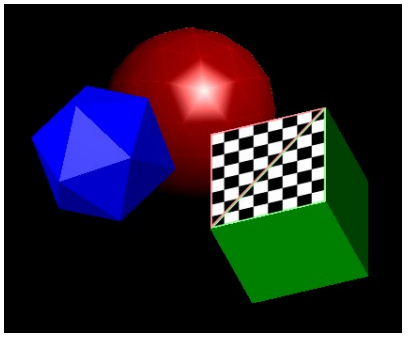
State Variables

Camera Position
Light Sources
Etc.

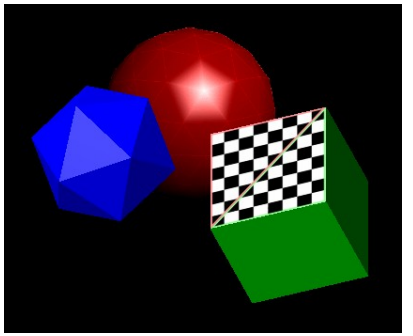
Geometry Pipeline

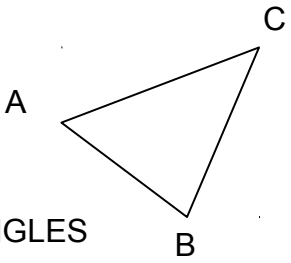
Fragment Pipeline

Front Buffer

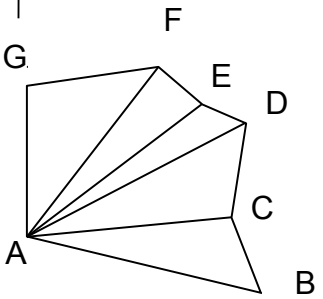


Back Buffer

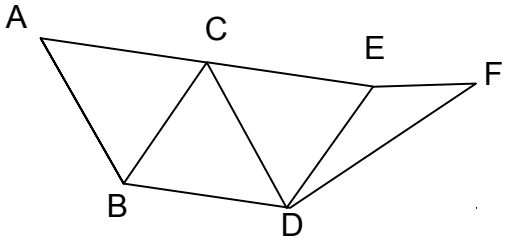




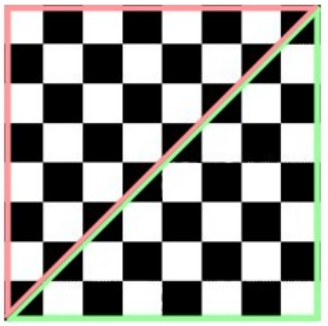
GL_TRIANGLES



GL_TRIANLE_FAN

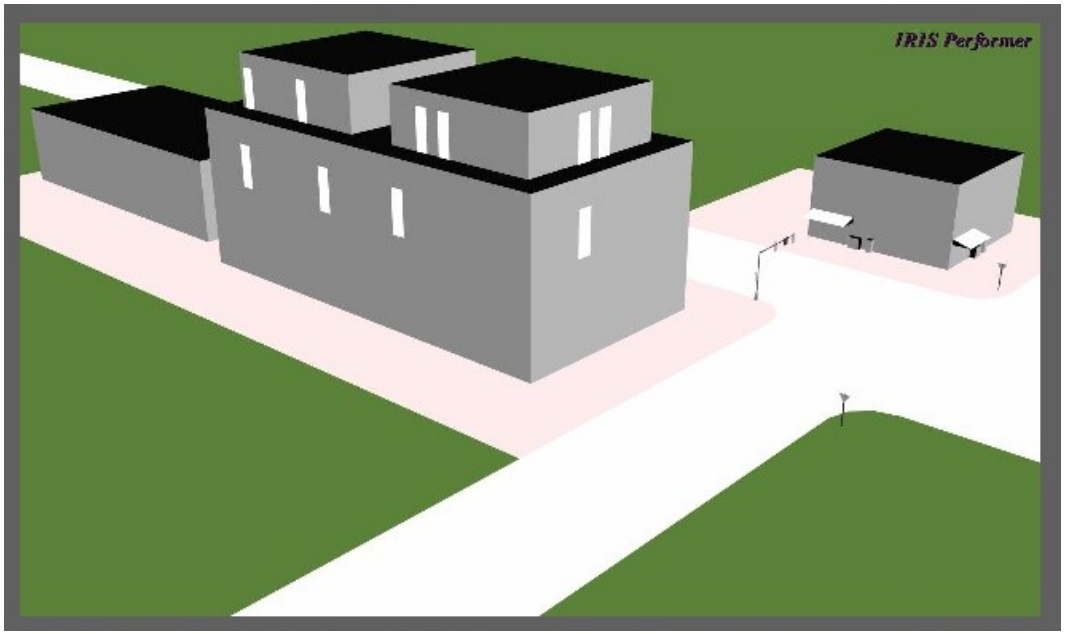


GL_TRIANGLE_STRIP

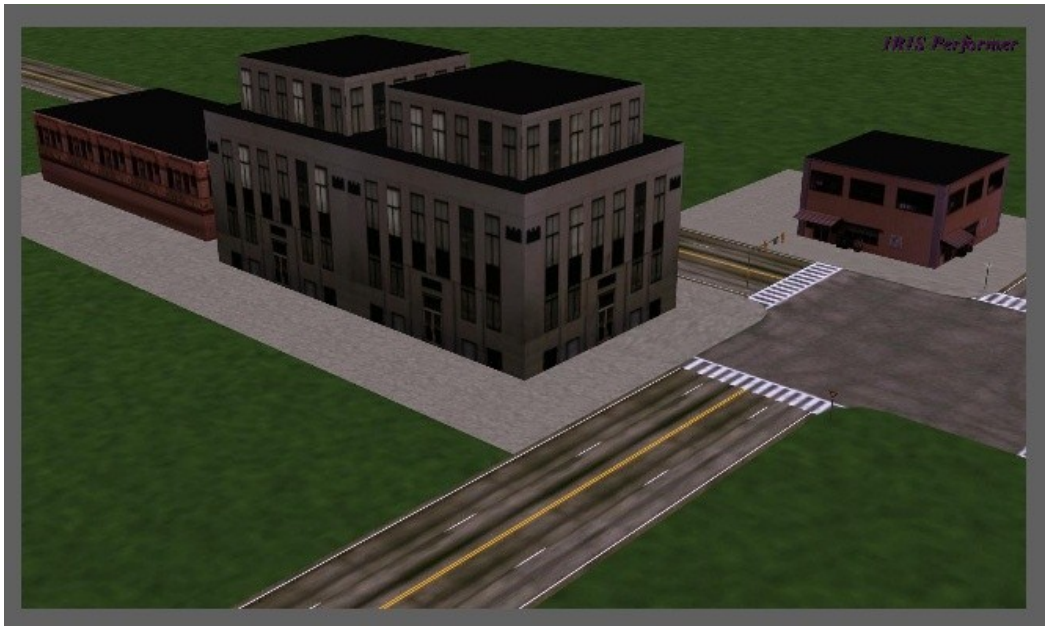


Texture Maps

Texturing Example

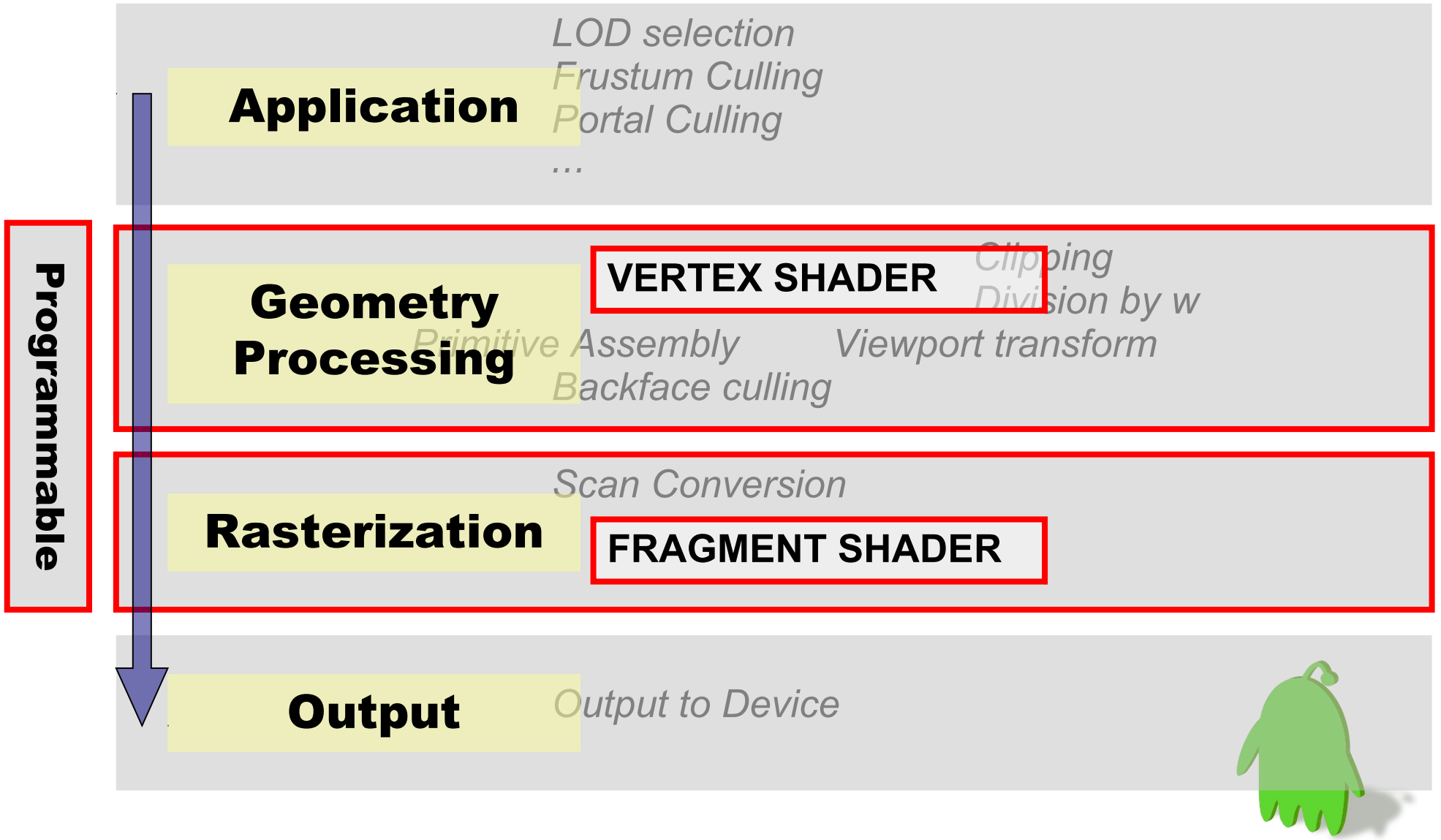


Before Texturing

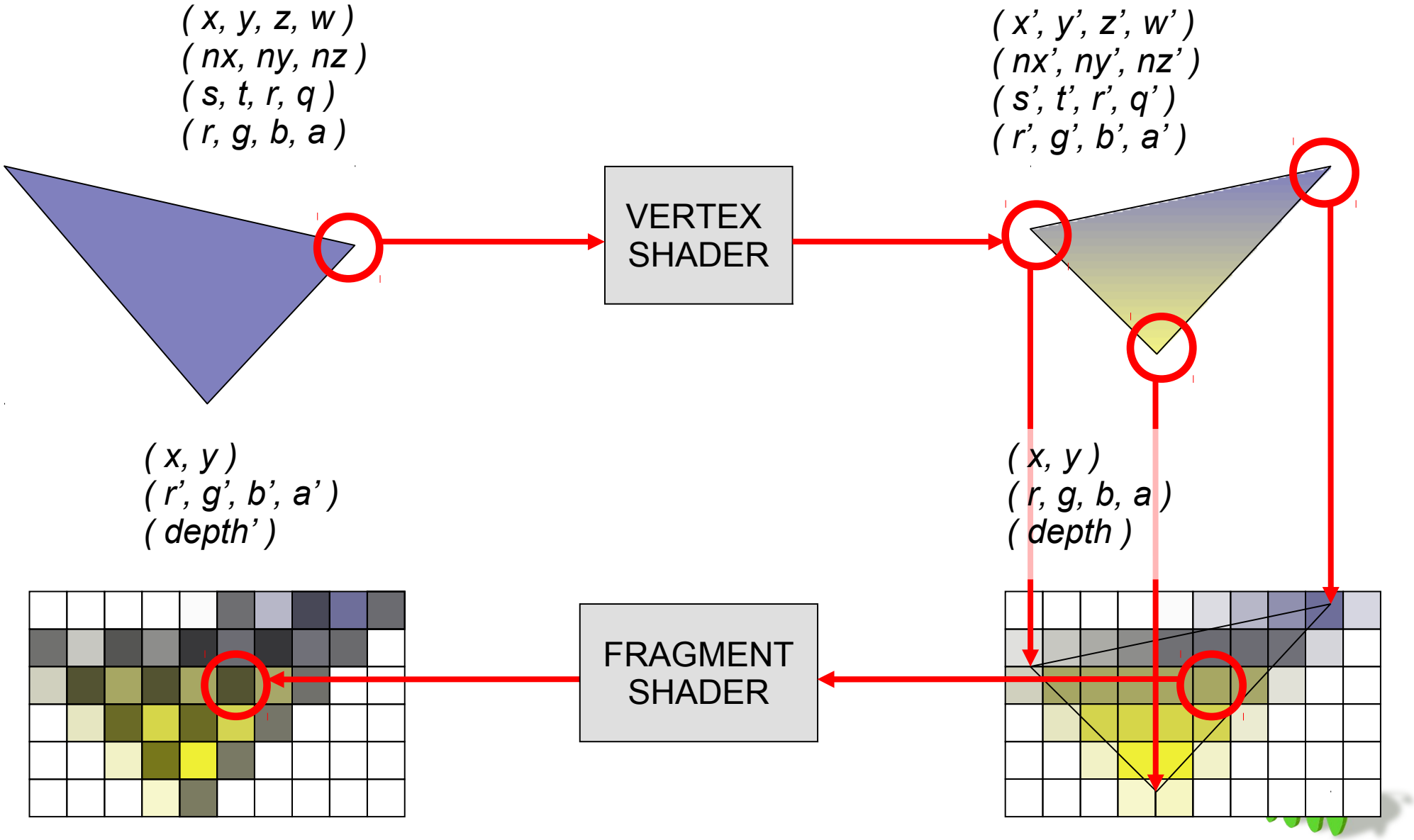


After Texturing

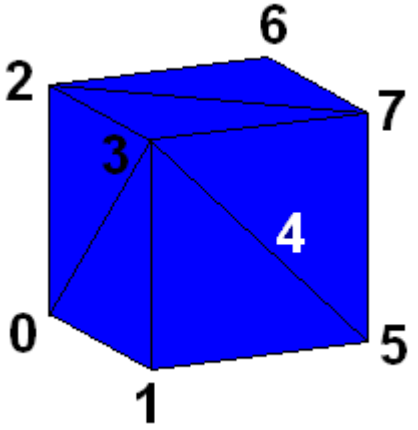
Graphics Pipeline



Vertex and Fragment Shaders



Embedded code with no lighting

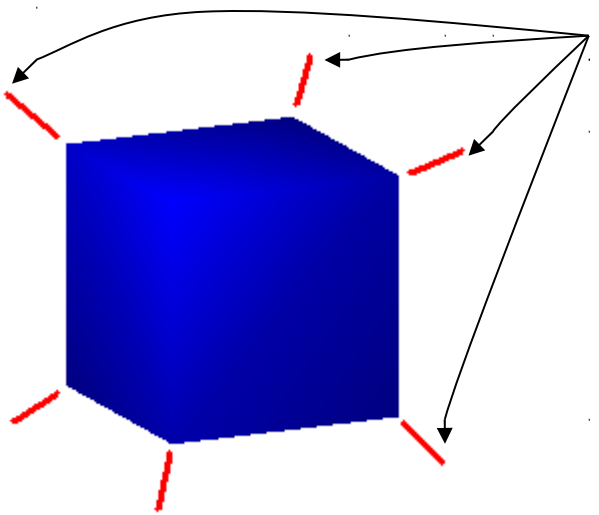


OpenGL|ES

- Added
  - Fixed-point and byte data
- Retained
  - Vertex Transforms and Lighting (mostly)
  - Multi-texturing (2D only)
  - Full Scene Antialiasing via Multisampling
  - Alpha blending

Embedded code with lighting

(Smooth Shaded)



OpenGL|ES 1.1 Example

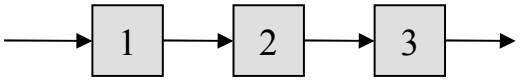
```
// Enable fixed-function shading (smooth or flat)
    glShadeModel(GL_SMOOTH);

// Define the appearance of triangle surfaces
    glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, fMaterialAmbient);
    glMaterialf(GL_FRONT_AND_BACK, GL_SHININESS, Shininess);

// Define the appearance and position of a light source
    glLightfv(GL_LIGHT0, GL_AMBIENT, fLightAmbient);
    glLightModelfv(GL_LIGHT_MODEL_AMBIENT, fAmbient);
    glLightfv(GL_LIGHT0, GL_POSITION, fLightPosition);

// Set pointers to geometry and other attributes and draw it
    glVertexPointer(3, GL_FLOAT, 0, Vertices);
    glTexCoordPointer(2, GL_FLOAT, 0, TexCoords);
    glNormalPointer(GL_FLOAT, 0, NormalsPerVertex);
    glDrawArrays(GL_TRIANGLES, 0, Count);
```

OpenGL|ES 2.0 Example

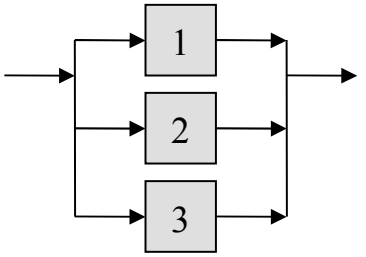
```
// Create a vertex shader object, load source code and compile it
    hVertexShader = glCreateShader(GL_VERTEX_SHADER);
    glShaderSource(hVertexShader, 1, pVertexShaderSourceCode, NULL);
    glCompileShader(hVertexShader);

// Create a shader program and attach the fragment and vertex shaders to it
    hProgram = glCreateProgram();
    glAttachShader(hProgram, hFragmentShader);
    glAttachShader(hProgram, hVertexShader);

// Link and load the new shader programs into the PowerVR SGX
    glLinkProgram(hProgram);
    glUseProgram(hProgram);

// Set pointers to geometry and other attributes and send to the vertex shader
    glVertexAttribPointer(Index, 3, GL_FLOAT, GL_TRUE, Stride, pAttributes);
    glDrawArrays(GL_TRIANGLES, 0, Count);
```

GPU = Graphics Processing Unit

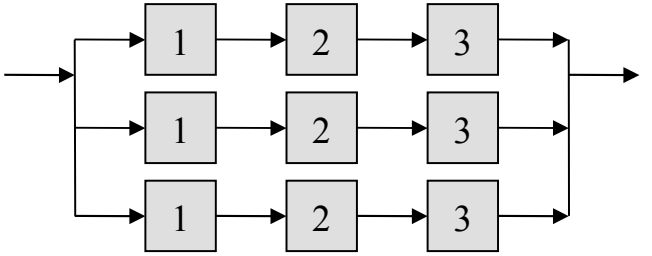
- Pipelining
  - Number of stages



- Parallelism
  - Number of parallel processes



- Parallelism + pipelining
  - Number of parallel pipelines

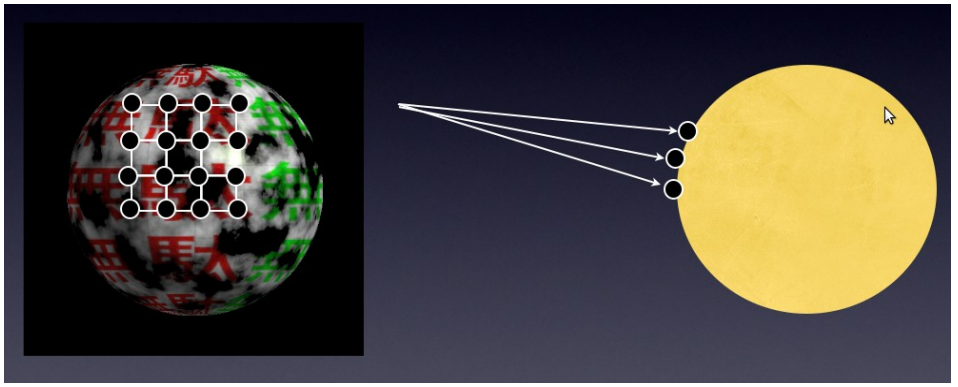


- Operates on 4 tuples
  - Position *(x, y, z, w)*
  - Color *(red, green, blue, alpha)*
  - Texture Coordinates *(s, t, r, q)*

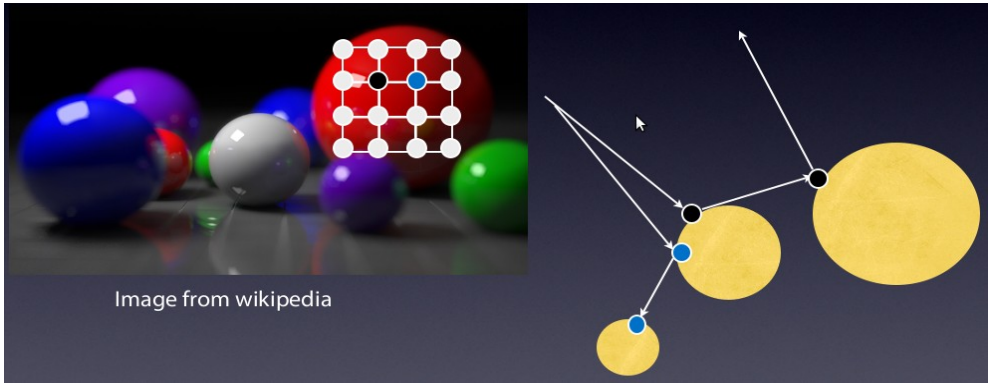
4 tuple ops, 1 clock cycle

- SIMD [Single Instruction Multiple Data]

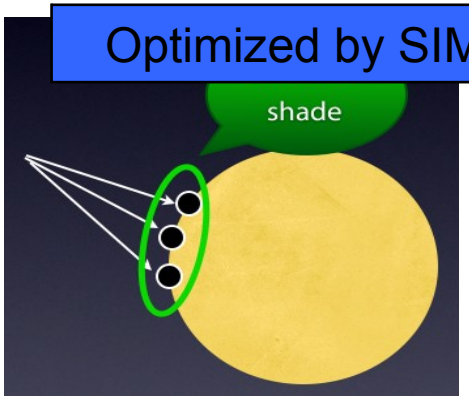
Reyes(scanline,polygon)

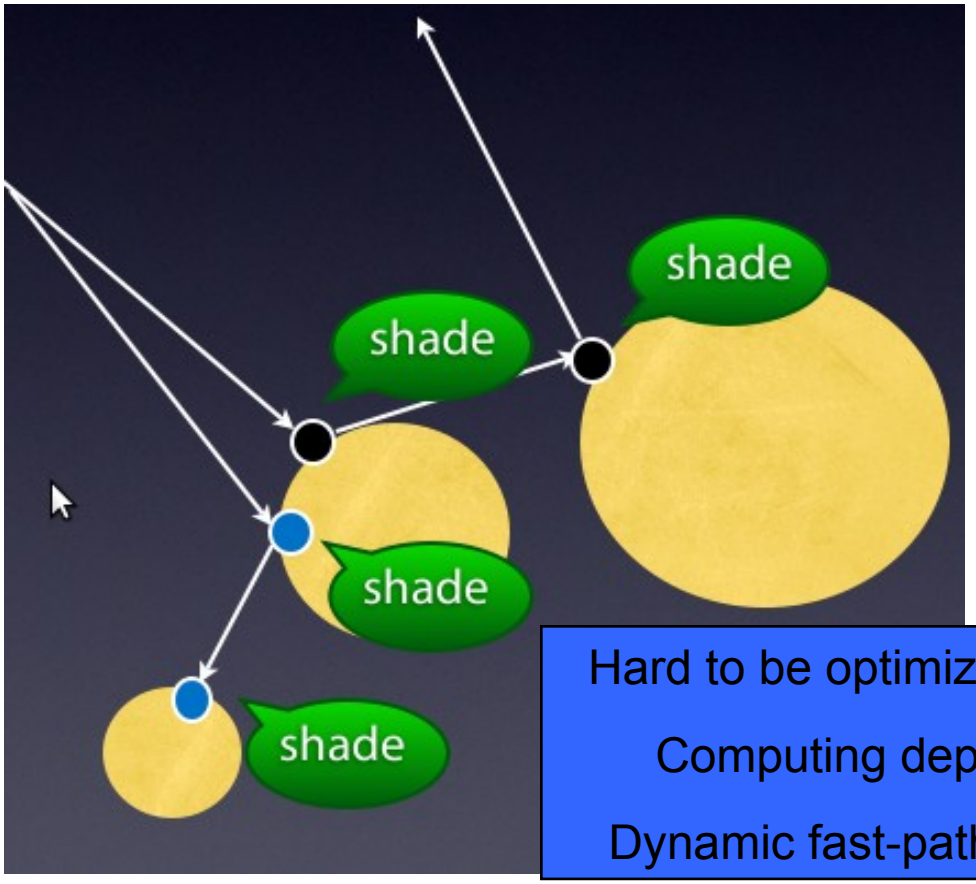


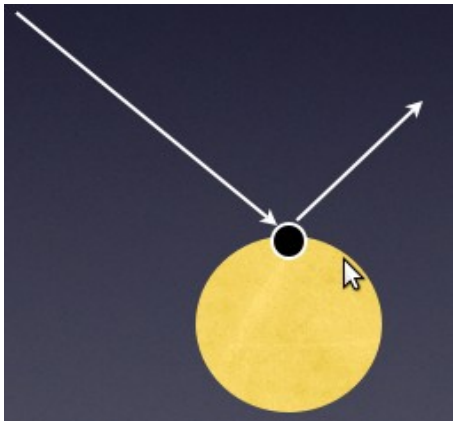
Raytracing



Optimized by SIMD







Hard to be optimized by SIMD

Computing dependency

Dynamic fast-paths required

Specialize Technique

color space convertion takes lots of time.
BGRA 444R → RGBA 8888

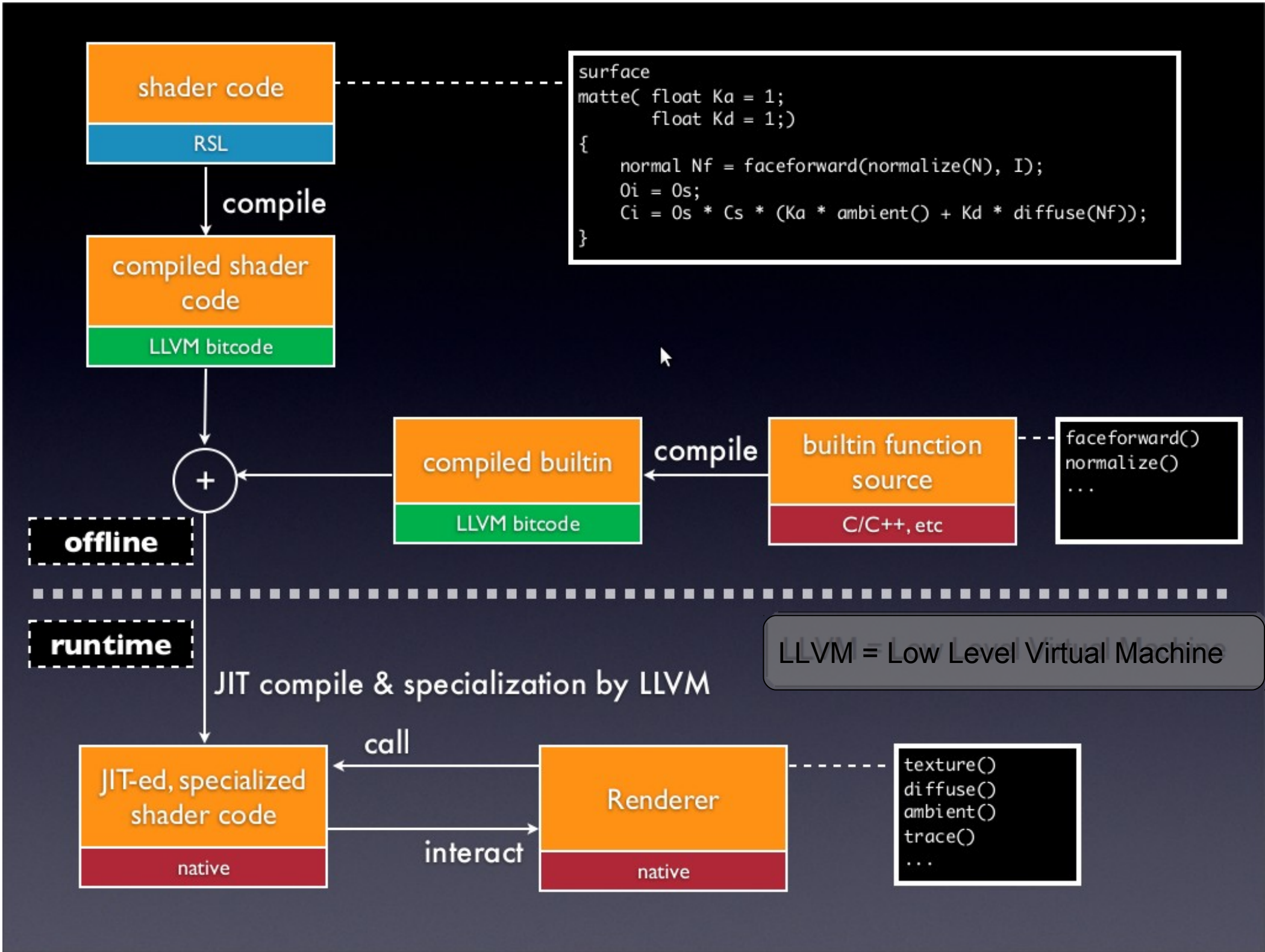
```
for each pixel {
  switch (infmt) {
  case RGBA 5551:
    R = (*in >> 11) & C
    G = (*in >> 6) & C
    B = (*in >> 1) & C
  ... }
  switch (outfmt) {
  case RGB888:
    *outptr = R << 16 |
              G << 8 ...
  }
}
```

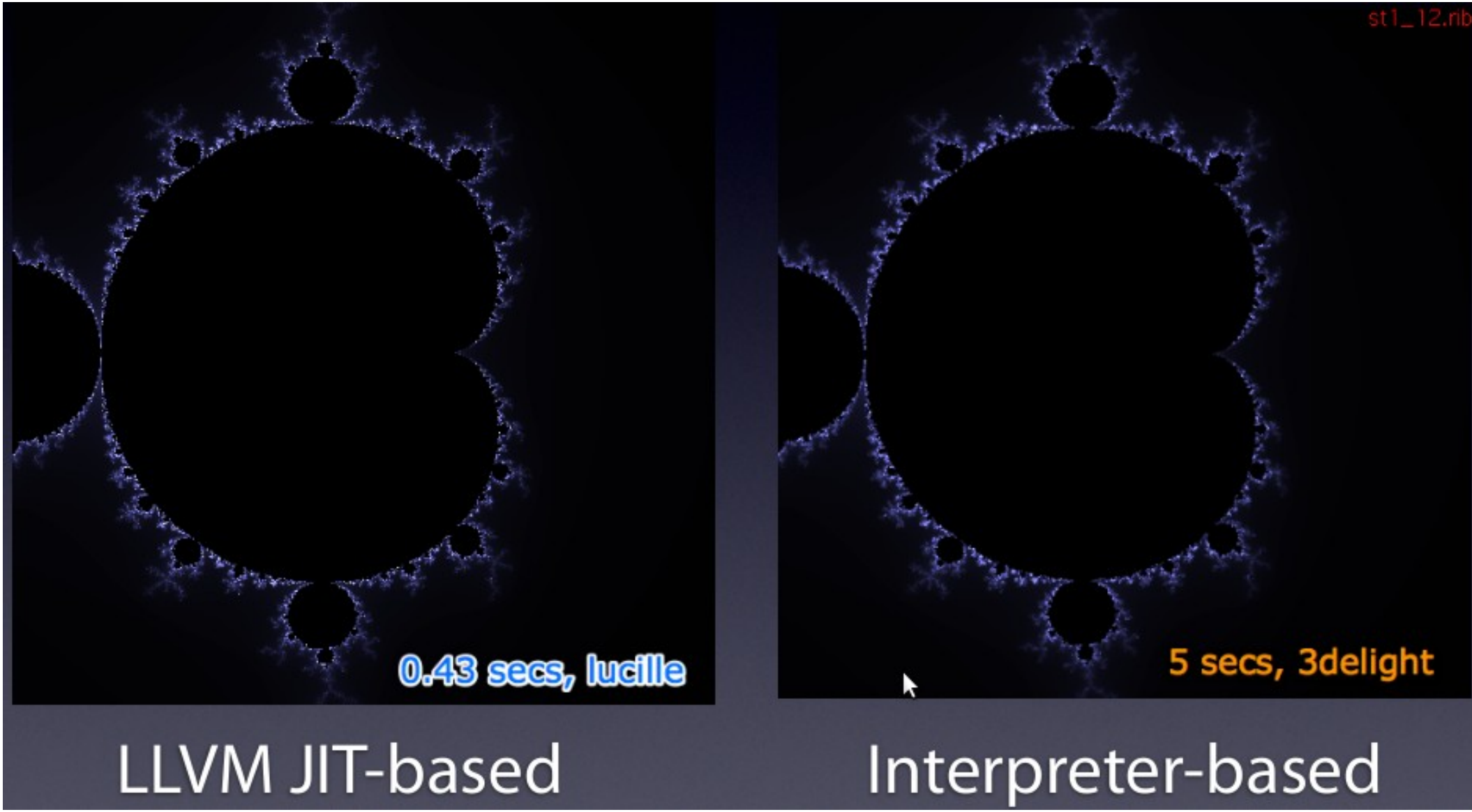
Run-time specialize

```
for each pixel {
  R = (*in >> 11) & C;
  G = (*in >> 6) & C;
  B = (*in >> 1) & C;
  *outptr = R << 16 |
            G << 8 ...
}
```

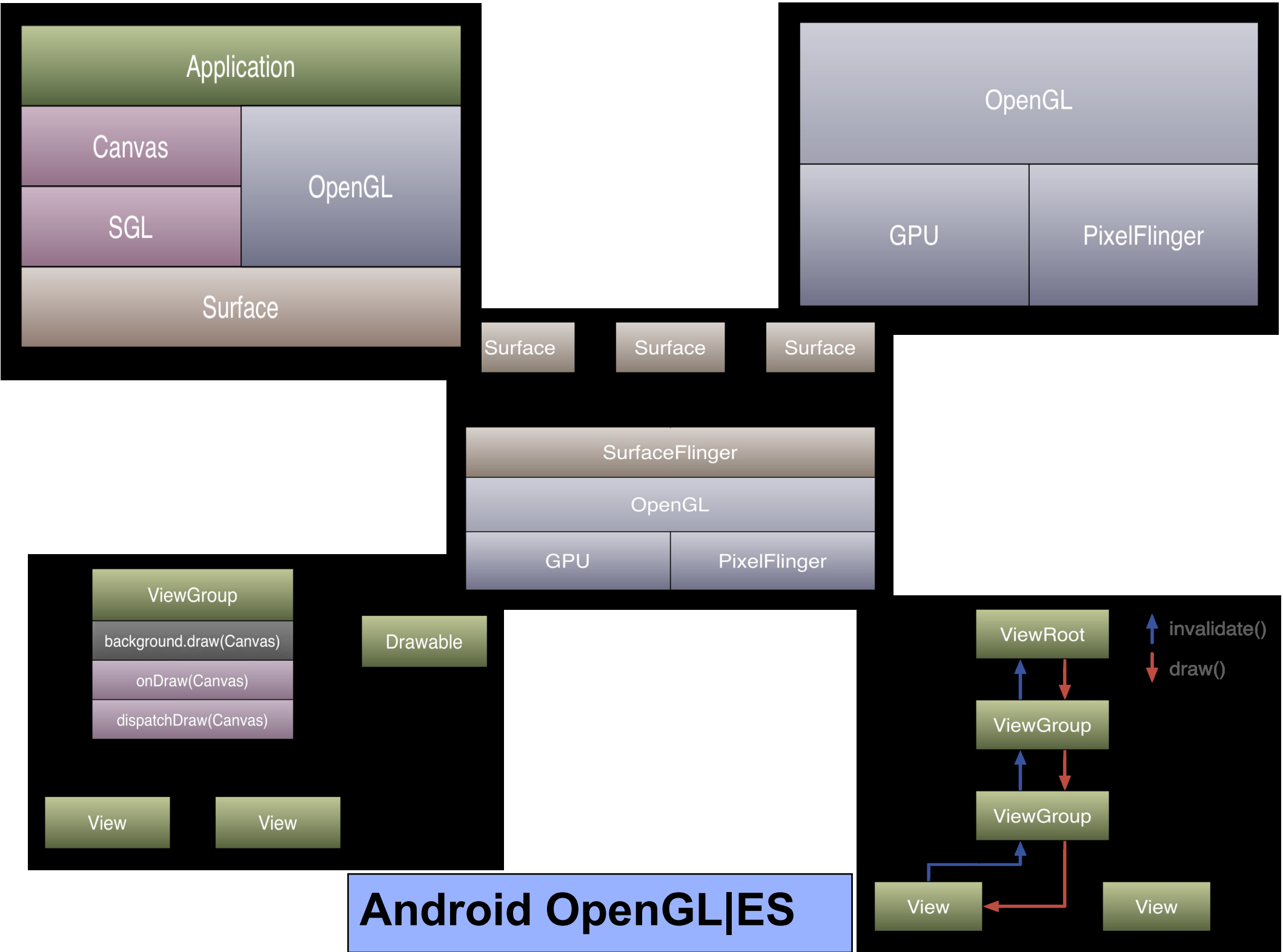
Compiler optimizes shifts and masking

Speedup depends on src/dest format:
– 5.4x speedup on average, 19.3x max speedup: (13.3MB/s to 257.7MB/s)


shader code
RSL
compile
compiled shader code
LLVM bitcode
surface
matte(float Ka = 1;
float Kd = 1;)
{
normal Nf = faceforward(normalize(N), I);
Oi = Os;
Ci = Os * Cs * (Ka * ambient() + Kd * diffuse(Nf));
}
+
compiled builtin
LLVM bitcode
compile
builtin function source
C/C++, etc
faceforward()
normalize()
...
offline
runtime
JIT compile & specialization by LLVM
LLVM = Low Level Virtual Machine
call
JIT-ed, specialized shader code
native
interact
Renderer
native
texture()
diffuse()
ambient()
trace()
...

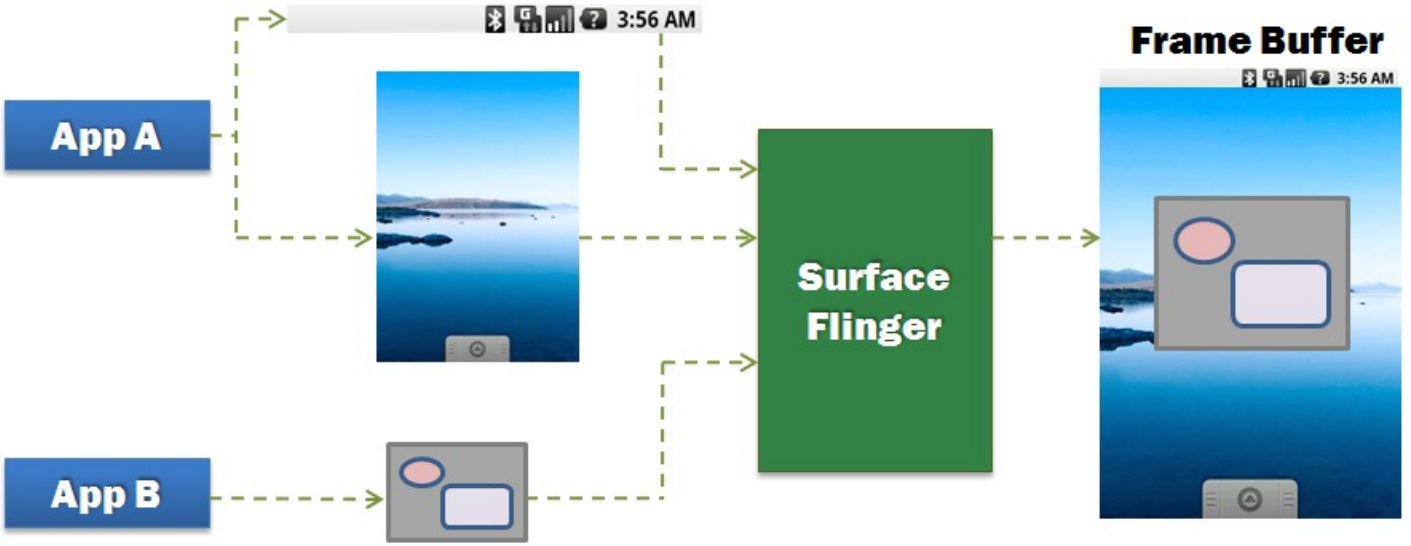


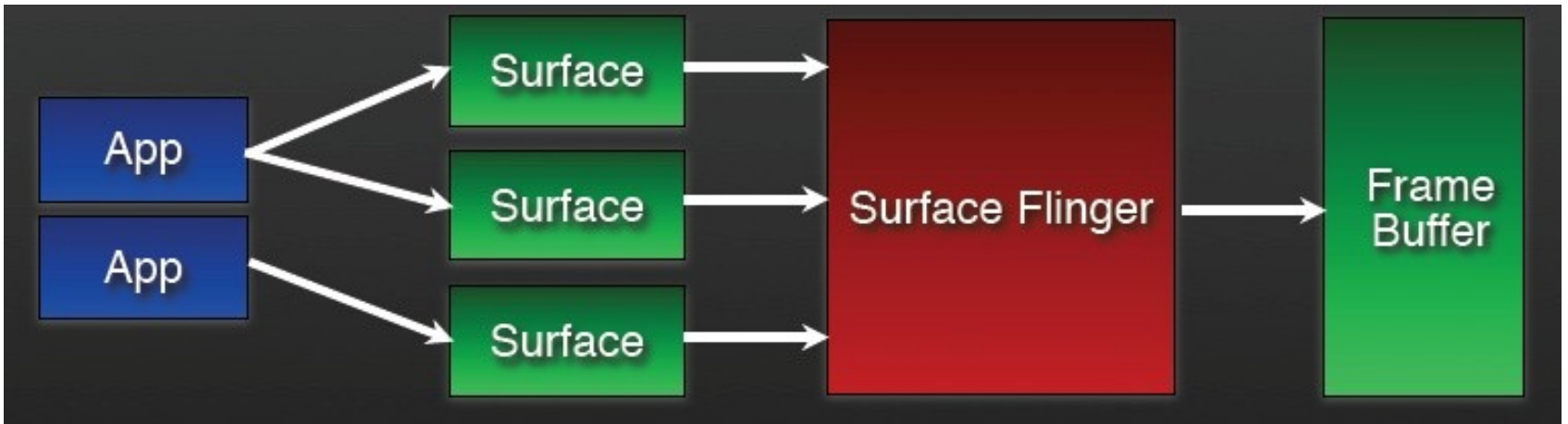
Mandelbort is optimized through LLVM JIT up to 11x.

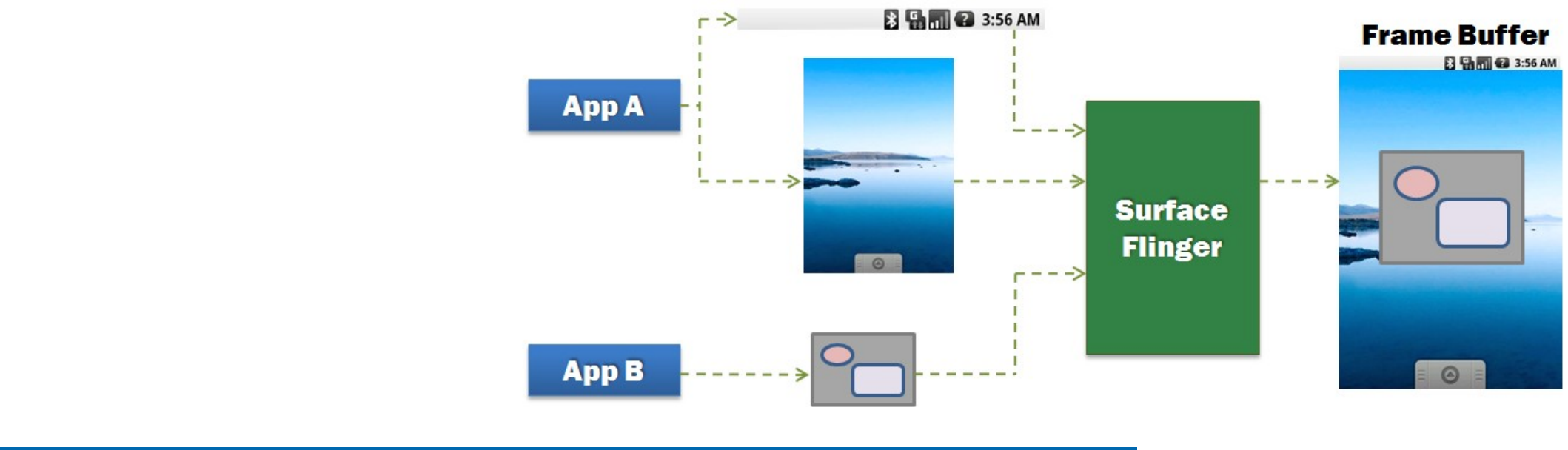

Application
Canvas
SGL
OpenGL
Surface
OpenGL
GPU
PixelFlinger
Surface
Surface
Surface
SurfaceFlinger
OpenGL
GPU
PixelFlinger
ViewGroup
background.draw(Canvas)
onDraw(Canvas)
dispatchDraw(Canvas)
Drawable
View
View
ViewRoot
invalidate()
draw()
ViewGroup
ViewGroup
View
View
Android OpenGL|ES

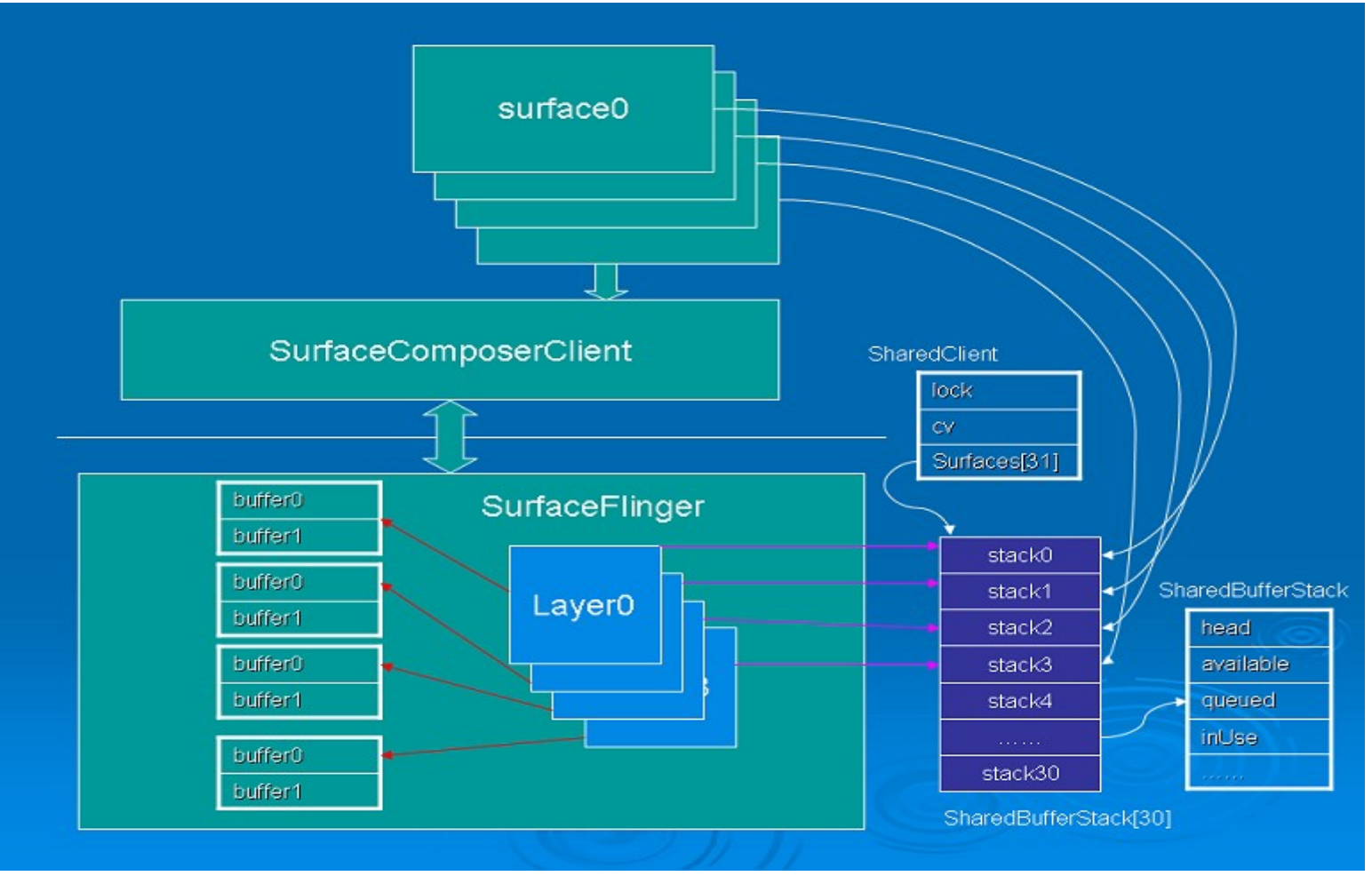
Android SurfaceFlinger

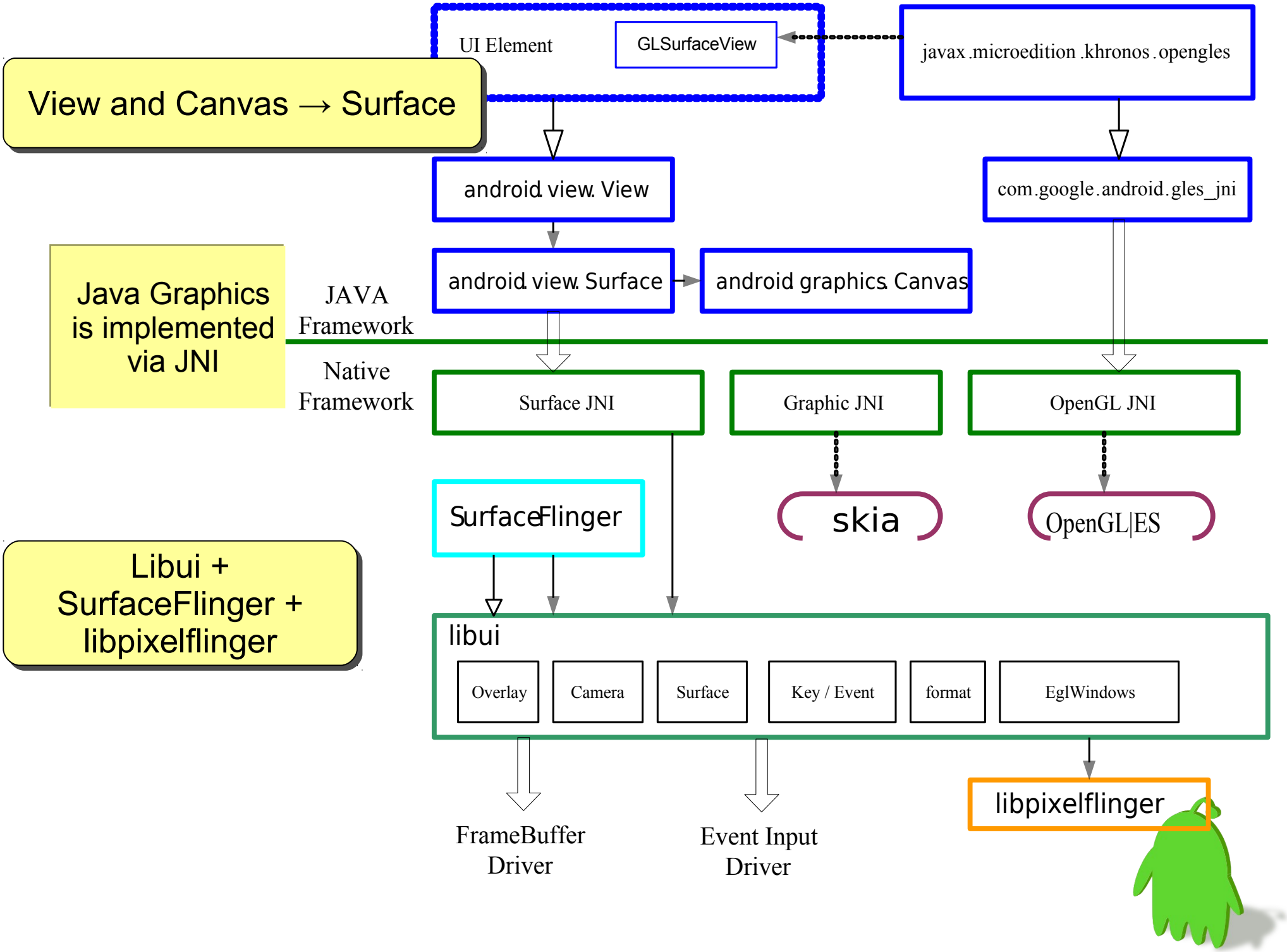
- Properties
  - Can combine 2D/3D surfaces and surfaces from multiple applications
  - Surfaces passed as buffers via Binder IPC calls
  - Can use OpenGL ES and 2D hardware accelerator for its compositions
    - Double-buffering using page-flip

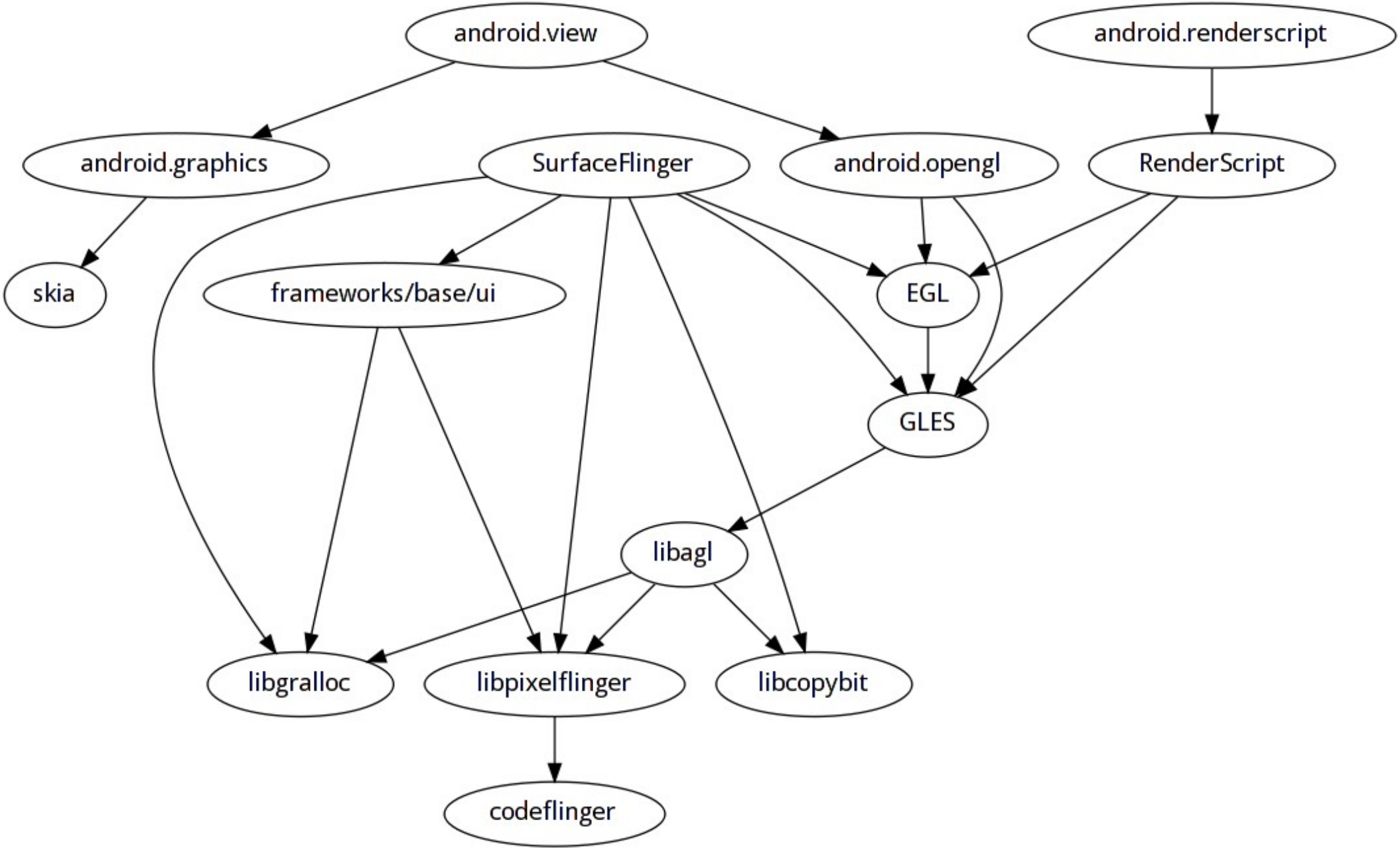





3:56 AM
App A
App B
Surface Flinger
Frame Buffer
3:56 AM


surface0
SurfaceComposerClient
SharedClient
lock
cv
Surfaces[31]
SurfaceFlinger
buffer0
buffer1
buffer0
buffer1
buffer0
buffer1
buffer0
buffer1
Layer0
stack0
stack1
stack2
stack3
stack4
.....
stack30
SharedBufferStack
head
available
queued
inUse
.....
SharedBufferStack[30]

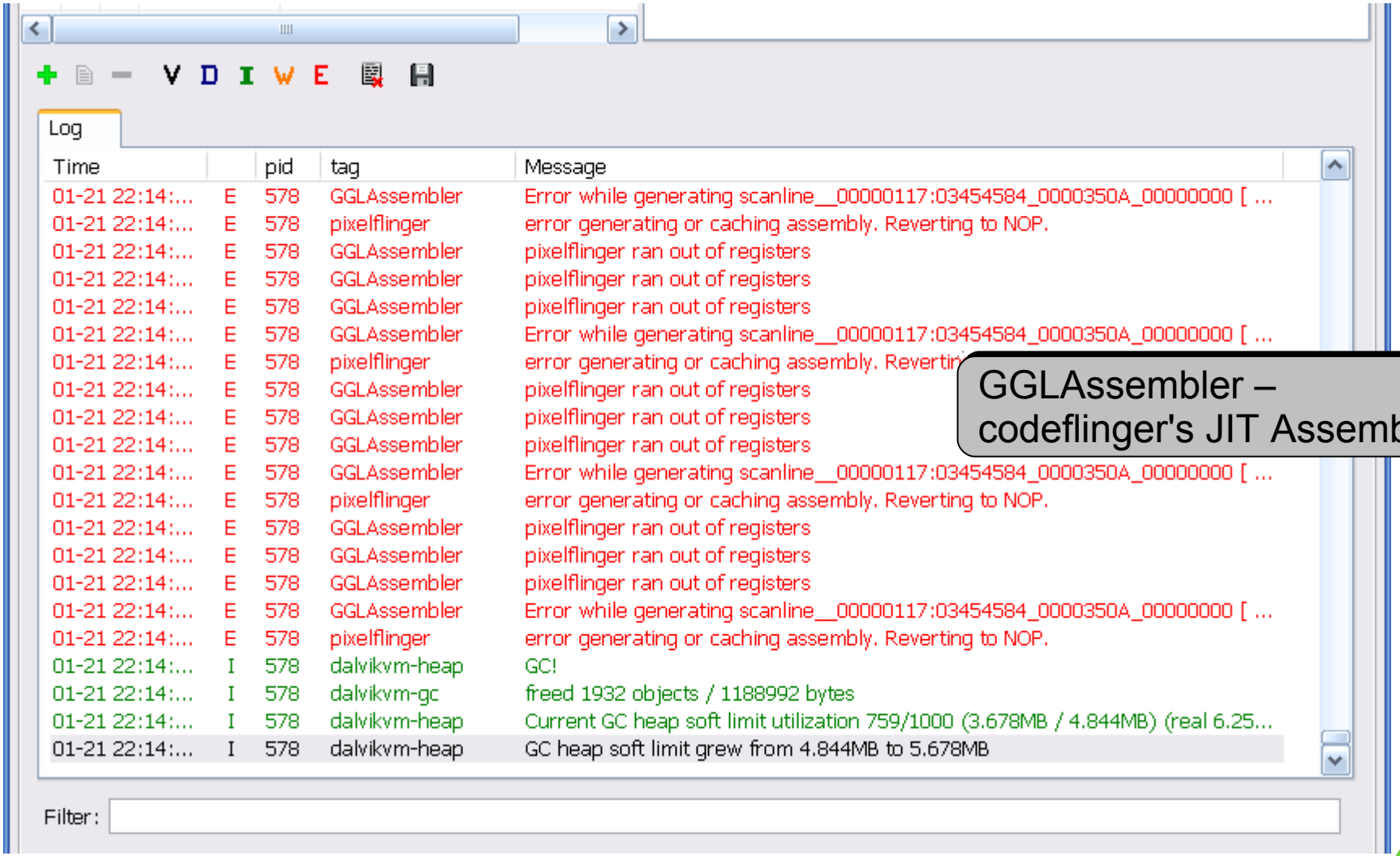

View and Canvas → Surface
UI Element
GLSurfaceView
javax.microedition.khronos.opengles
android.view.View
com.google.android.gles_jni
Java Graphics is implemented via JNI
android.view.Surface
android.graphics.Canvas
JAVA Framework
Native Framework
Surface JNI
Graphic JNI
OpenGL JNI
SurfaceFlinger
skia
OpenGL|ES
Libui + SurfaceFlinger + libpixelflinger
libui
Overlay
Camera
Surface
Key / Event
format
EglWindows
libpixelflinger
FrameBuffer Driver
Event Input Driver


android.view
android.renderscript
android.graphics
SurfaceFlinger
android.opengl
RenderScript
skia
frameworks/base/ui
EGL
GLES
libagl
libgralloc
libpixelflinger
libcopybit
codeflinger

PixelFlinger : software renderer

- Render functions: pointx, linex, recti, trianglex
- Texture and color buffer: activeTexture, bindTexture, colorBuffer, readBuffer, depthBuffer, BindTextureLod
- …
- Device framebuffer functions: copyPixels, rasterPos2x, rasterPos2i
- Optimizer: codeflinger (JIT assembler)

```
I/SurfaceFlinger( 1931): OpenGL informations:
I/SurfaceFlinger( 1931): vendor    : Android
I/SurfaceFlinger( 1931): renderer : Android PixelFlinger 1.2
I/SurfaceFlinger( 1931): version   : OpenGL ES-CM 1.0
```

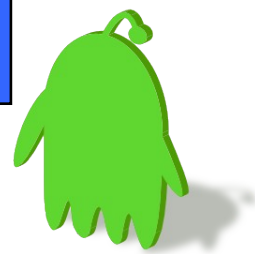


GGLAssembler – codeflinger's JIT Assembler

```
commit 77cadd2ffada95bb3279552e1a29f4bcf4012228
Author: Jim Huang <jserv@0xlab.org>
Date:   Wed Jan 13 01:01:18 2010 +0800

    [libpixelflinger] Adds UXTB16 support to Pixelflinger
    ...
    Uses UXTB16 to extract channels for SIMD operations, rather than creating and
    ANDing with masks. Saves a register and is faster on A8, as UXTB16 result can
    feed into first stage of multiply, unlike AND.

    Also, used SMULWB rather than SMULBB, which allows removal of MOVs used to
    rescale results.

    Code has been scheduled for A8 pipeline, specifically aiming to allow
    multiplies to issue in pipeline 0, for efficient dual issue operation.

    Testing on SpriteMethodTest (http://code.google.com/p/apps-for-android/) gives
    8% improvement (12.7 vs. 13.7 fps.)
```

```
libpixelflinger/codeflinger/ARMAssembler.cpp
@@ -424,5 +424,15 @@ void ARMAssembler::SMLAW(int cc, int y,
     *mPC++ = (cc<<28) | 0x1200080 | (Rd<<16) | (Rn<<12) | (Rs<<8) | (y<<4) | Rm;
 }

+#if 0
+#pragma mark -
+#pragma mark Byte/half word extract and extend (ARMv6+ only)...
+#endif
+
+void ARMAssembler::UXTB16(int cc, int Rd, int Rm, int rotate)
+{
+    *mPC++ = (cc<<28) | 0x6CF0070 | (Rd<<12) | ((rotate >> 3) << 10) | Rm;
+}
+
 }; // namespace android
```

opcode



```
libpixelflinger/codeflinger/ARMAssemblerProxy.cpp
@@ -195,6 +195,10 @@ void ARMAssemblerProxy::SMLAW(int cc, int y, int Rd, int Rm, int Rs, int Rn) {
     mTarget->SMLAW(cc, y, Rd, Rm, Rs, Rn);
 }

+void ARMAssemblerProxy::UXTB16(int cc, int Rd, int Rm, int rotate) {
+    mTarget->UXTB16(cc, Rd, Rm, rotate);
+}
+

 }; // namespace android
```

```
libpixelflinger/codeflinger/texturing.cpp

@@ -868,6 +869,106 @@ void GGLAssembler::filter24(
     load(txPtr, texel, 0);
 }

+#if __ARM_ARCH__ >= 6
+// ARMv6 version, using UXTB16, and scheduled for Cortex-A8 pipeline
+void GGLAssembler::filter32(
+        const fragment_parts_t& parts,
+        pixel_t& texel, const texture_unit_t& tmu,
+        int U, int V, pointer_t& txPtr,
+        int FRAC_BITS)
+{
...

+    UXTB16(AL, temp, pixel, 0);
+    if (round) {
+        ADD(AL, 0, u, u, imm(1<<(adjust-1)));
+        MOV(AL, 0, u, reg_imm(u, LSR, adjust));
+    }
+    LDR(AL, pixellb, txPtr.reg, reg_scale_pre(offsetlb));
+    MUL(AL, 0, dh, temp, u);
+    UXTB16(AL, temp, pixel, 8);
+    MUL(AL, 0, dl, temp, u);
+    RSB(AL, 0, k, u, imm(0x100));
...
+#else
 void GGLAssembler::filter32(
         const fragment_parts_t& parts,
         pixel_t& texel, const texture_unit_t& tmu,
```

Reimplement filter32()

In optimized ASM

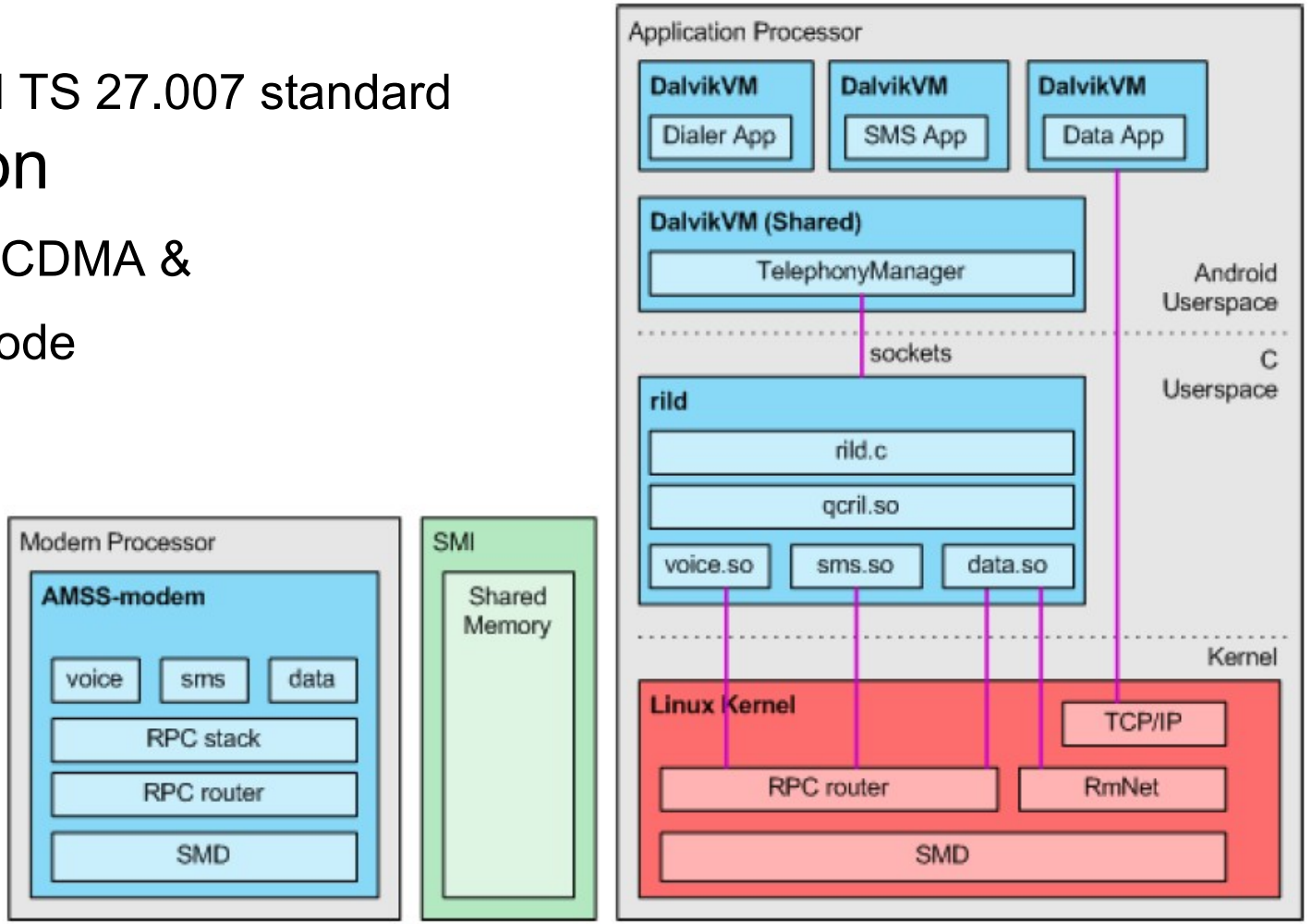
Operating System

Applied in Telephony

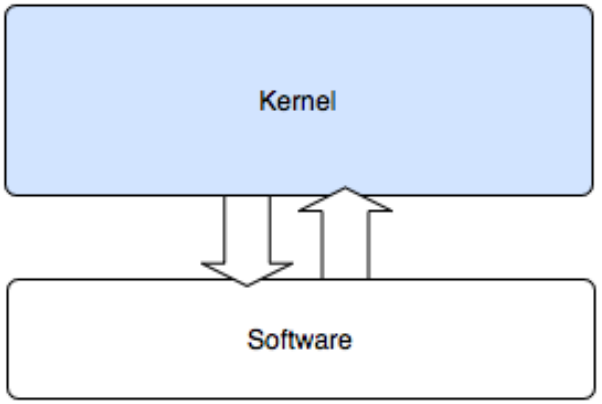
RIL → Telephony

- Radio Interface Layer(RIL)
  - HAL → Android TelephonyManager → baseband modem
  - Voice, Data, SMS, SIM, SIMToolkit
  - Android RIL → AT command
- RIL API
  - Android follows GSM TS 27.007 standard
- RIL implementation
  - Qualcomm supports CDMA &

    CDMA-GSM Multi-mode

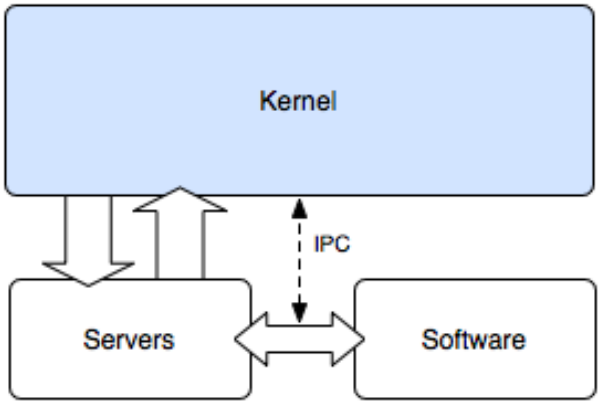


OS Kernels



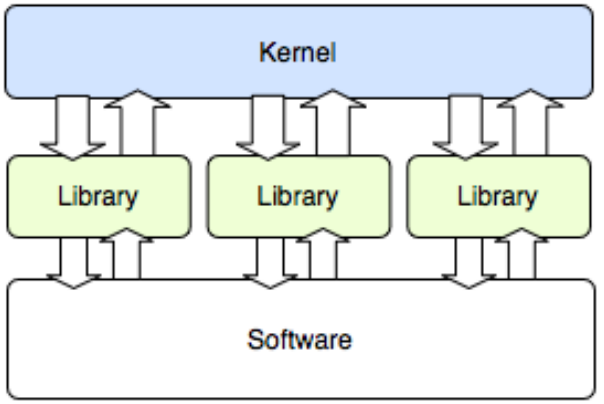
Monolithic Kernel

Traditional UNIX, Linux, ...



Micro Kernel

Mach, L4, Symbian OS



Exokernel

L4 Microkernel

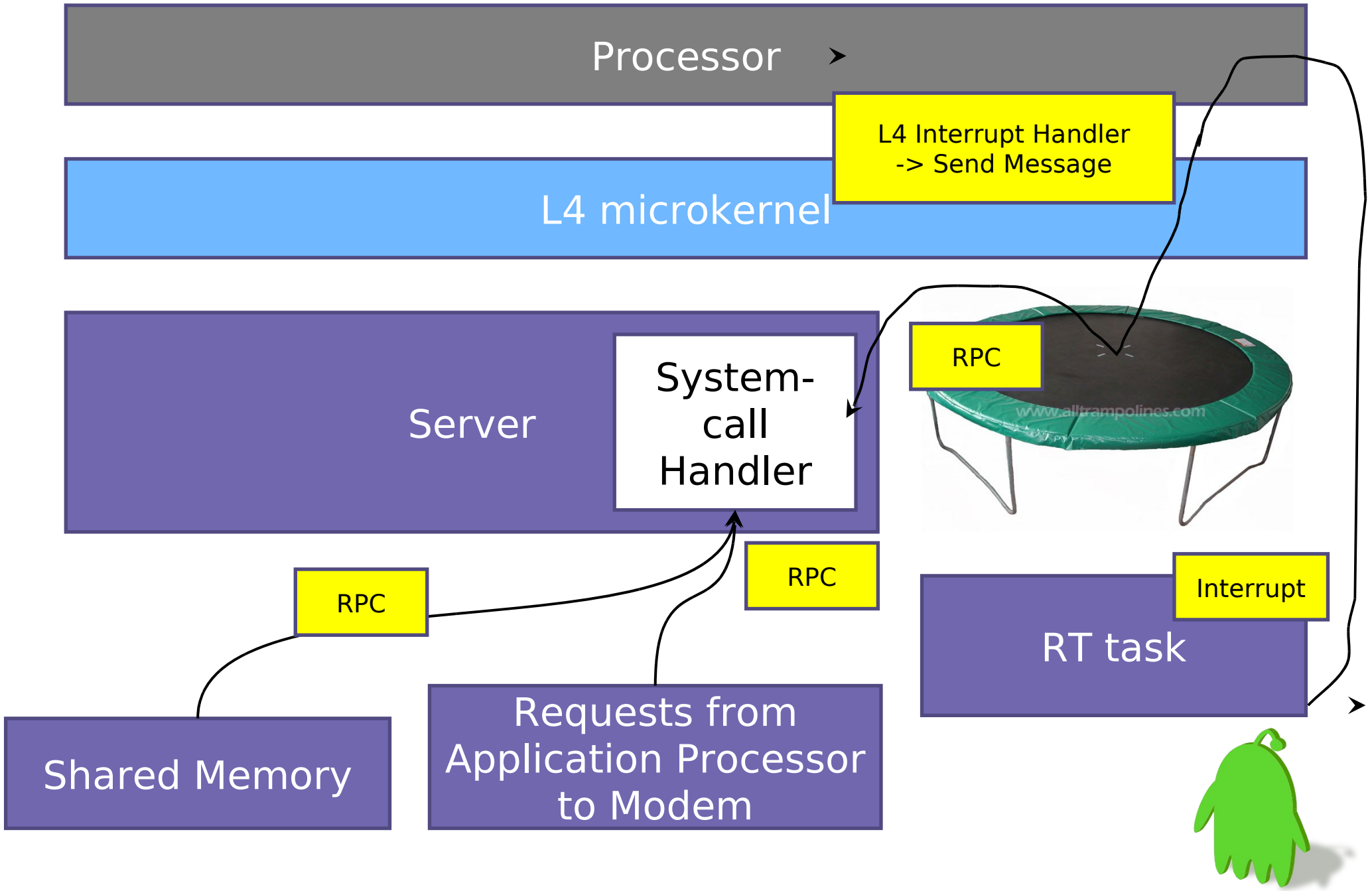
- Developed by Jochen Liedtke in 1995.
  - German National Research Center for IT
- Developed from scratch
- 2nd generation microkernel
- Small and Fast Kernel
  - 7 system calls
  - 12KB
- In November 2005, NICTA announced that Qualcomm was deploying NICTA's L4 version on their Mobile Station Modem chipsets.

Pipes and RPC

System	Latency	Bandwidth
(1) Linux pipe	29 μs	41 MB/s
(1a) L^{4}Linux pipe	46 μs	40 MB/s
(1b) L^{4}Linux (trampoline) pipe	56 μs	38 MB/s
(1c) MkLinux (user) pipe	722 μs	10 MB/s
(1d) MkLinux (in-kernel) pipe	316 μs	13 MB/s
(2) L4 pipe	22 μs	48–70 MB/s
(3) synchronous L4 RPC	5 μs	65–105 MB/s
(4) synchronous mapping RPC	12 μs	2470–2900 MB/s

Table 4: *Pipe and RPC performance.* (133 MHz Pentium.) Only communication costs are measured, not the costs to generate or consume data.

Communications



Virtual Machine

Applied in Database

sqlite3_prepare()

```
/*
** Compile the UTF-8 encoded SQL statement zSql into a statement handle.
*/
int sqlite3_prepare(
  sqlite3 *db,            /* Database handle. */
  const char *zSql,       /* UTF-8 encoded SQL statement. */
  int nBytes,             /* Length of zSql in bytes. */
  sqlite3_stmt **ppStmt,  /* OUT: A pointer to the prepared
       statement */
  const char** pzTail     /* OUT: End of parsed string */
)
{
...
      sqlite3VdbeSetNumCols(sParse.pVdbe, 5);
      sqlite3VdbeSetColName(sParse.pVdbe, 0, COLNAME_NAME, "addr", P3_STATIC);
      sqlite3VdbeSetColName(sParse.pVdbe, 1, COLNAME_NAME, "opcode", P3_STATIC);
      sqlite3VdbeSetColName(sParse.pVdbe, 2, COLNAME_NAME, "p1", P3_STATIC);
      sqlite3VdbeSetColName(sParse.pVdbe, 3, COLNAME_NAME, "p2", P3_STATIC);
      sqlite3VdbeSetColName(sParse.pVdbe, 4, COLNAME_NAME, "p3", P3_STATIC);
…
```

Opcode!

SQLite Virtual Machine (VDBE)

...And what about other platforms?

- The concept can be applied to all VM platforms
- Application virtual machines (short list)
  - .NET (CLR)
  - Java Virtual Machine (JVM)
  - Dalvik virtual machine (Google Android)
  - PHP (Zend Engine)
  - Flash Player / AIR - ActionScript Virtual Machine (AVM)
  - SQLite virtual machine (VDBE; Virtual DataBase Engine)
  - Perl virtual machine

Inspiration


抗

Inspiration

- Computer science concepts are applied everywhere in Android software stack.
- Essential system software brings the further optimizations and feasibility of new technologies.
- Advanced compiler techniques are already applied in every domain
- Best Practice in Android

http://0xlab.org