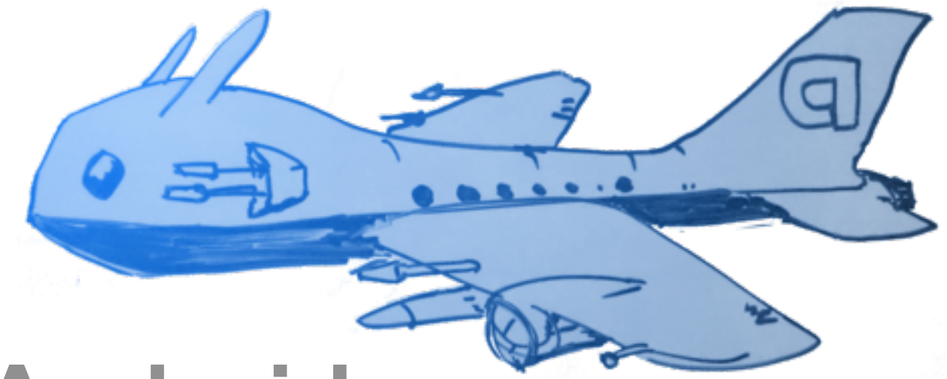

Google™
11
I/O



Bringing C and C++ Games to Android

Ian Ni-Lewis
Dan Galpin
Game Developer Advocates
May, 11 2011

Feedback: http://goo.gl/NudVs
#Android

Google I/O 11

About This Talk

Audience

- C/C++ Developers
- Android background
- Game development experience
- No prior NDK knowledge needed

Agenda

- Programming Android in C/C++?
- Using the NDK
- Bringing your game to Android
- Troubleshooting, tips, and tricks
- Best practices

Programming Android in C/C++?

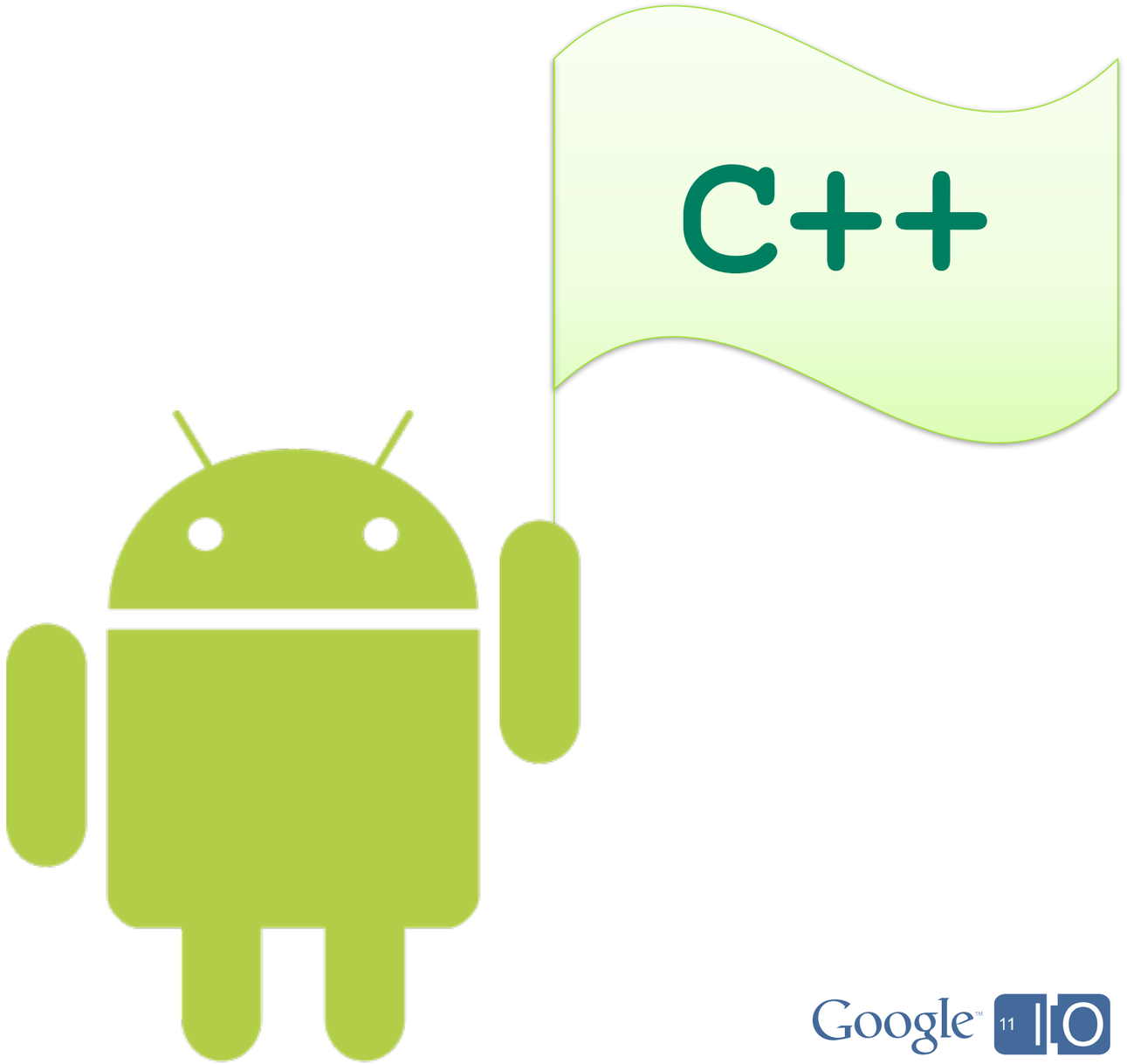
Native development on phones, tablets, and beyond

The NDK Provides Support for C/C++ Development

NDK = Native Development Kit

Scenarios:

- Reuse existing libraries
- Accelerate key subroutines
  - Access VFP and NEON
- Port entire games



What's in the Box?

Toolchain

– Standard gcc cross-compilers

Really fancy build system

– Primarily written in GNU make

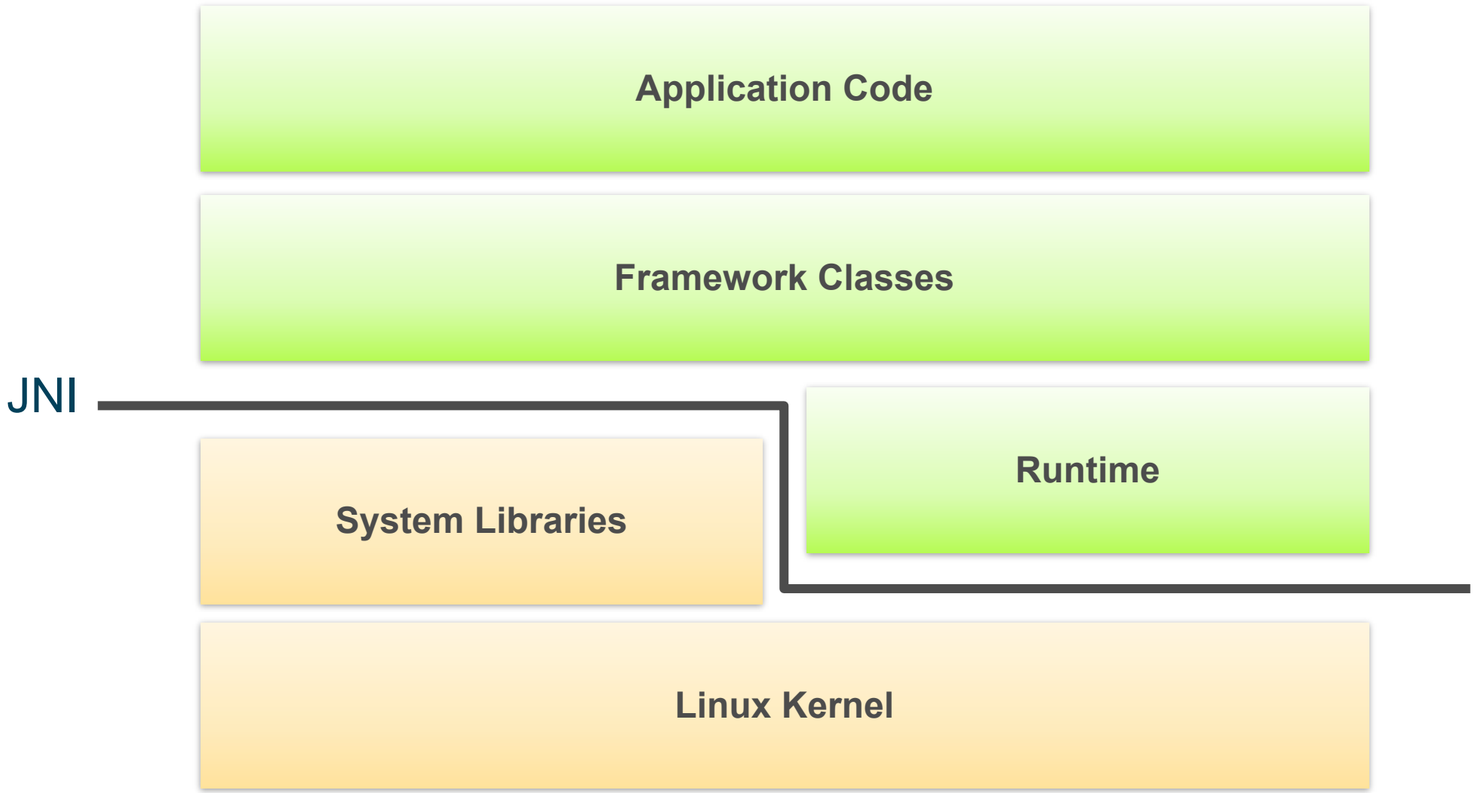
– Based on Android platform build system

Headers and libraries

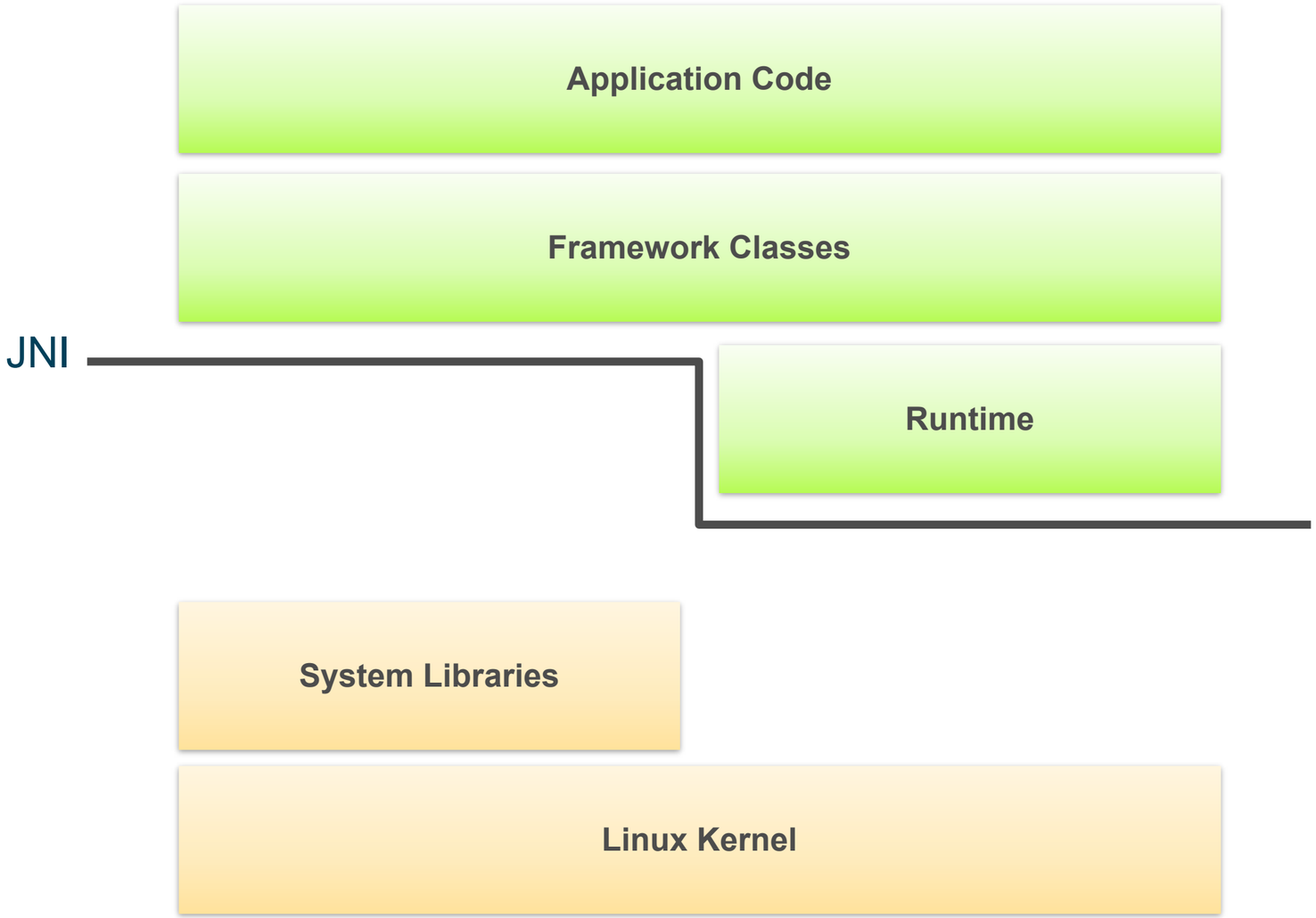
– C and C++ runtimes

– Stable system interface

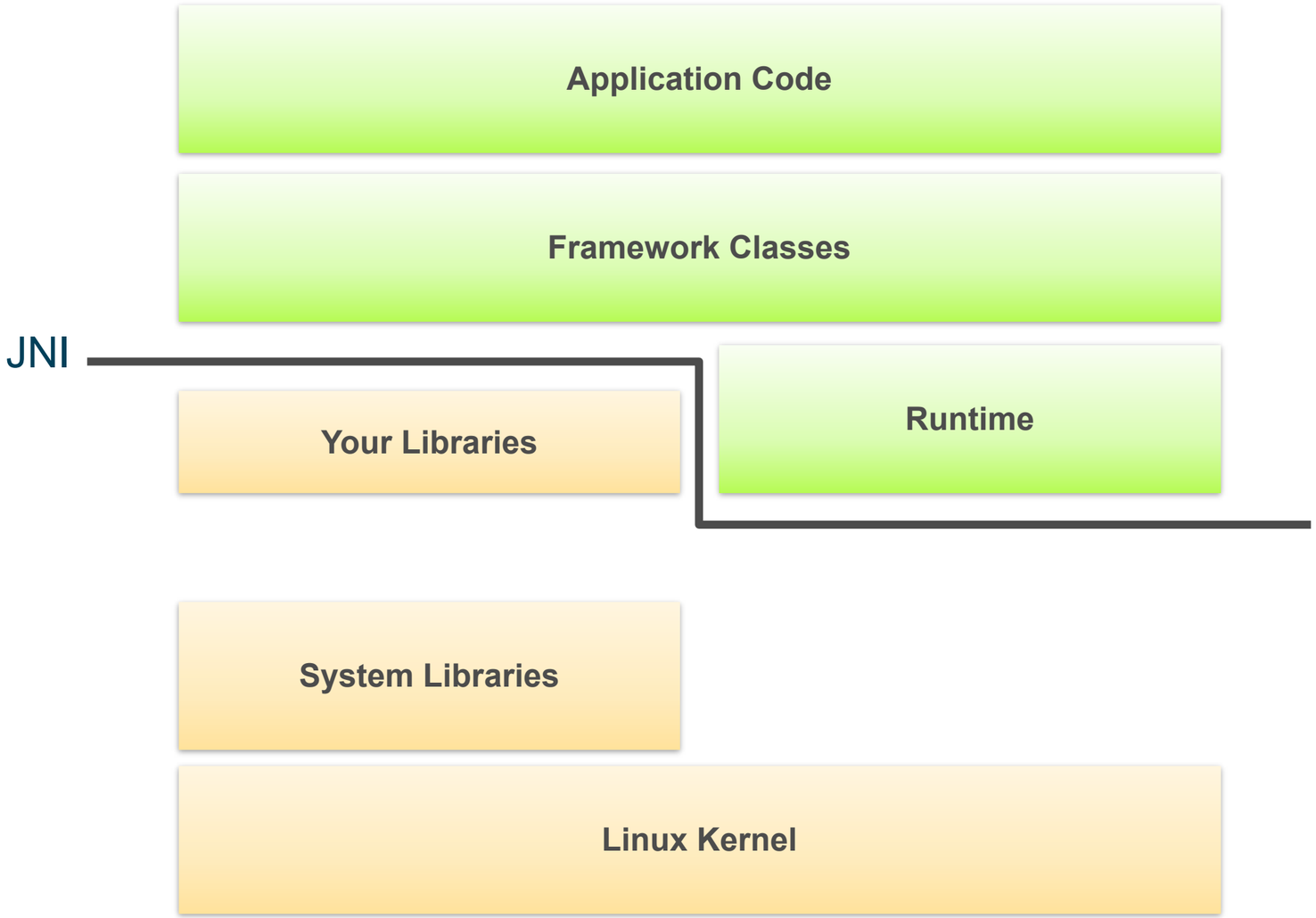
Why are Headers and Libraries So Important?



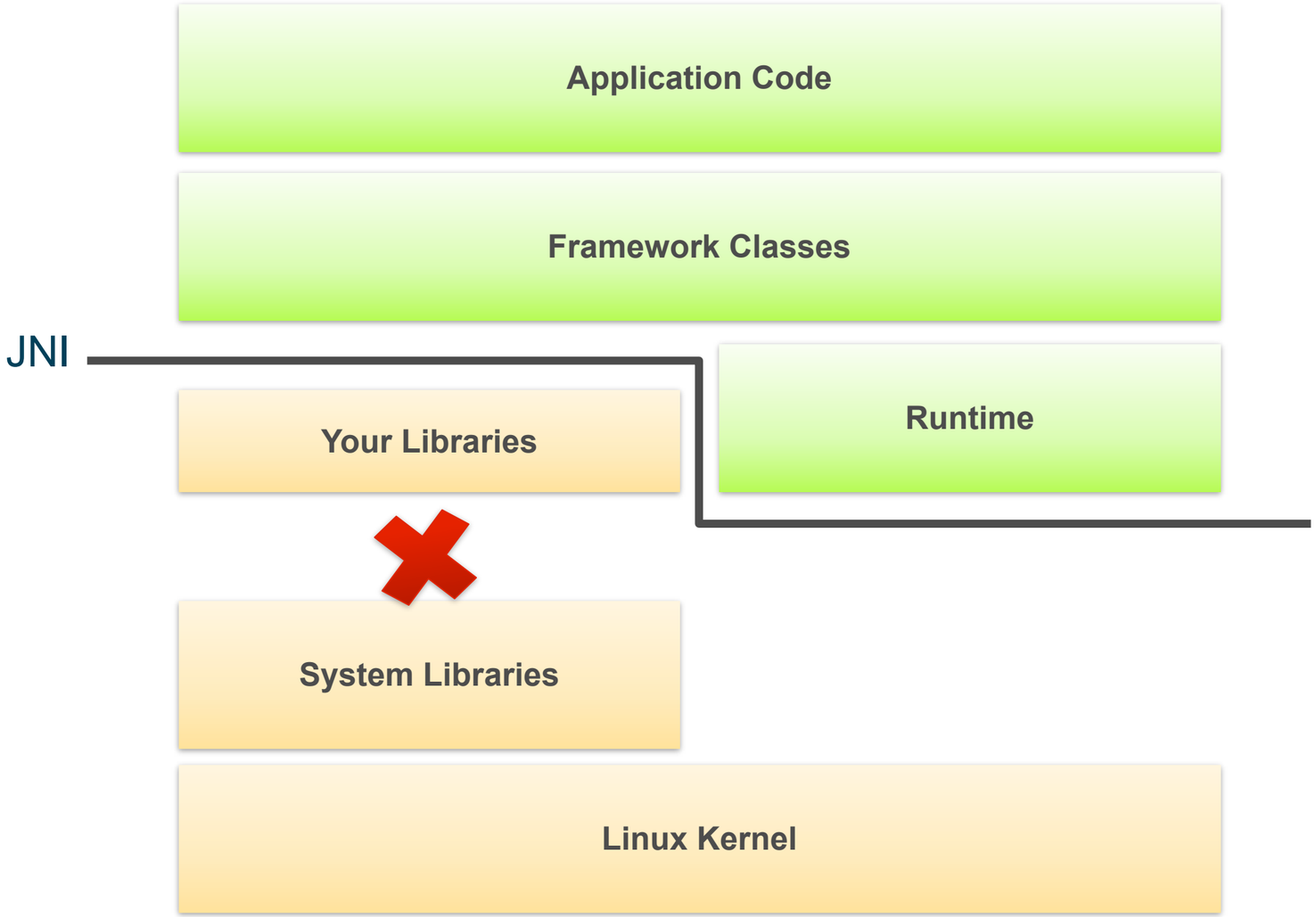
Why are Headers and Libraries So Important?



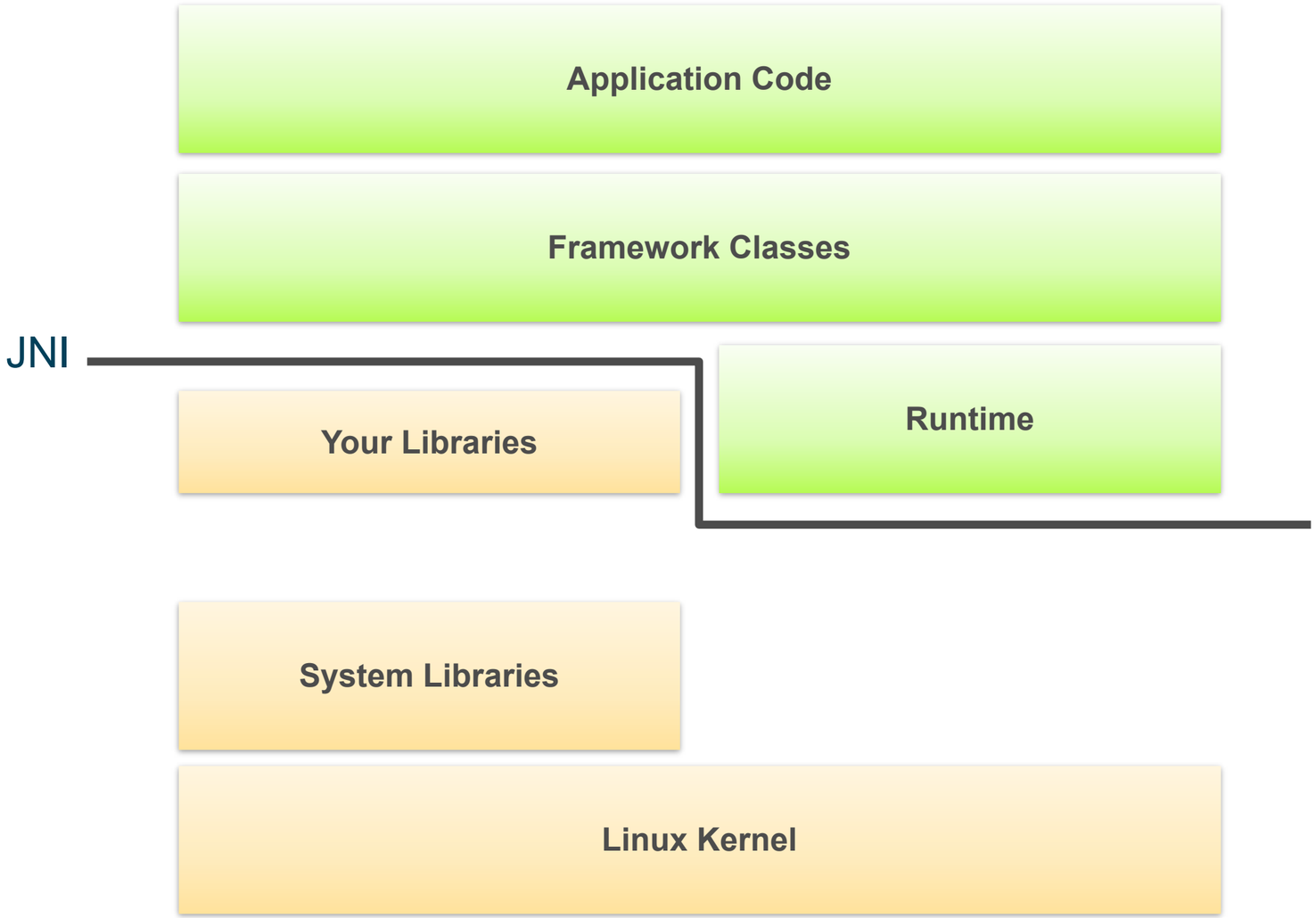
Why are Headers and Libraries So Important?



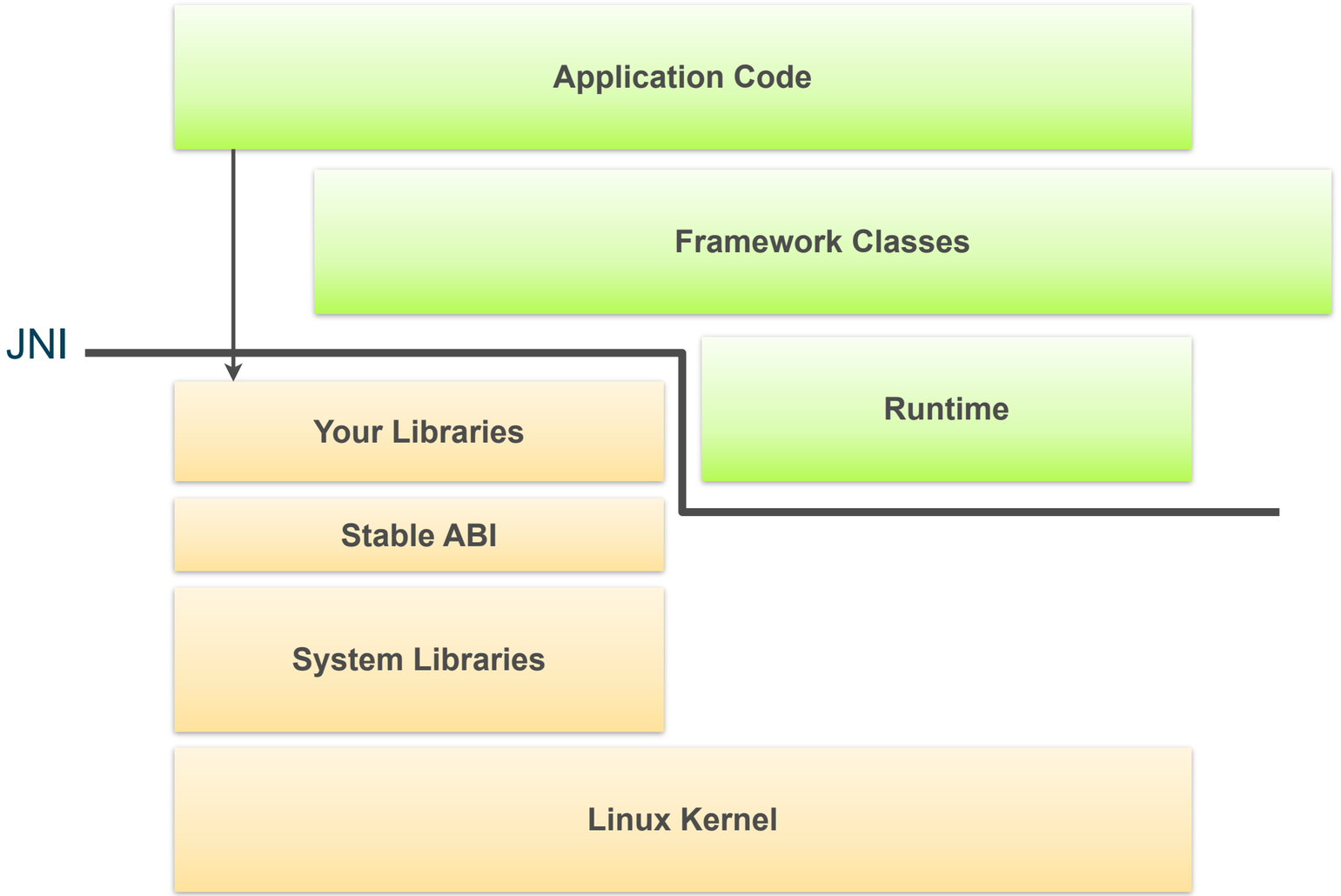
Why are Headers and Libraries So Important?



Why are Headers and Libraries So Important?



Why are Headers and Libraries So Important?



Feature	Library
C runtime	libc
C++ runtime	libstdc++
Math	libm
Dynamic Linking	libdl
Logging	liblog
Zlib compression	libz
OpenGL ES 1.1	libGLESv1_CM
OpenGL ES 2.0	libGLESv2

Feature	Library
C runtime	libc
C++ runtime	libstdc++
Math	libm
Dynamic Linking	libdl
Logging	liblog
Zlib compression	libz
OpenGL ES 1.1	libGLESv1_CM
OpenGL ES 2.0	libGLESv2
JNI Graphics	libjnigraphics

Feature	Library
C runtime	libc
C++ runtime	libstdc++
Math	libm
Dynamic Linking	libdl
Logging	liblog
Zlib compression	libz
OpenGL ES 1.1	libGLESv1_CM
OpenGL ES 2.0	libGLESv2
JNI Graphics	libjnigraphics
EGL	libEGL
OpenSL ES	libOpenSLES
Native framework / NativeActivity)	libandroid

What the NDK is Not

- Not the end of Dalvik
- Not always higher performance
- Not necessarily the right choice for every game

Dalvik vs. C++

Easily Extensible

Flexible

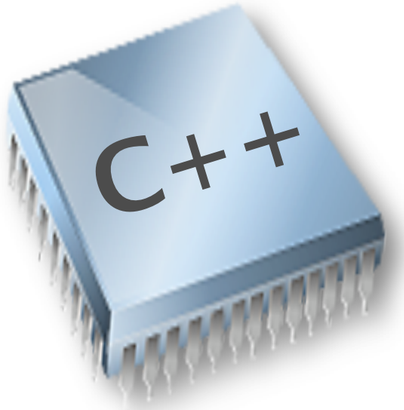
Automatic
Memory
Management

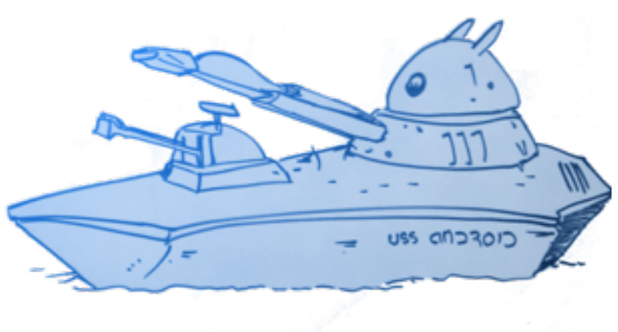
Android API
Access

Fast!

Low-Level

Common on
Other Platforms





How to Use the NDK

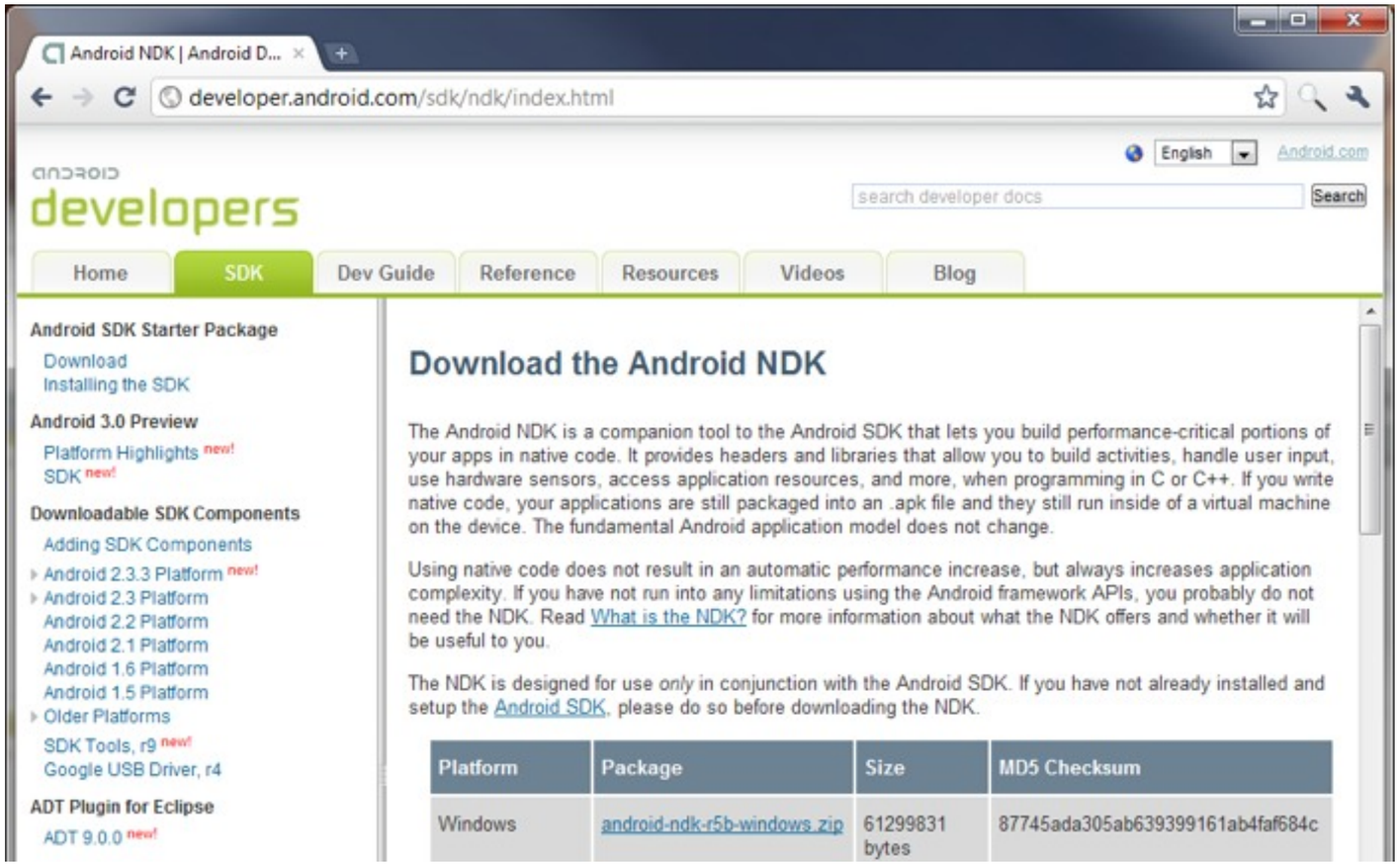
Installing the NDK

Download NDK from developer.android.com/sdk/ndk

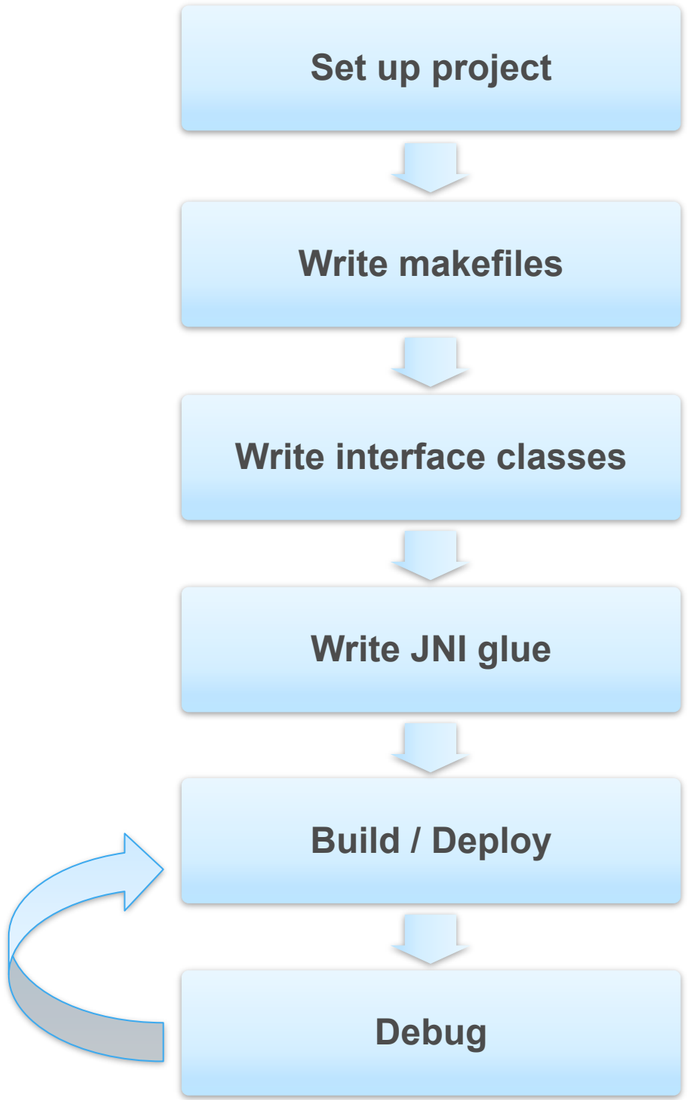
- Add NDK folder to your path
- Install Java JDK
- Install Ant

Windows:

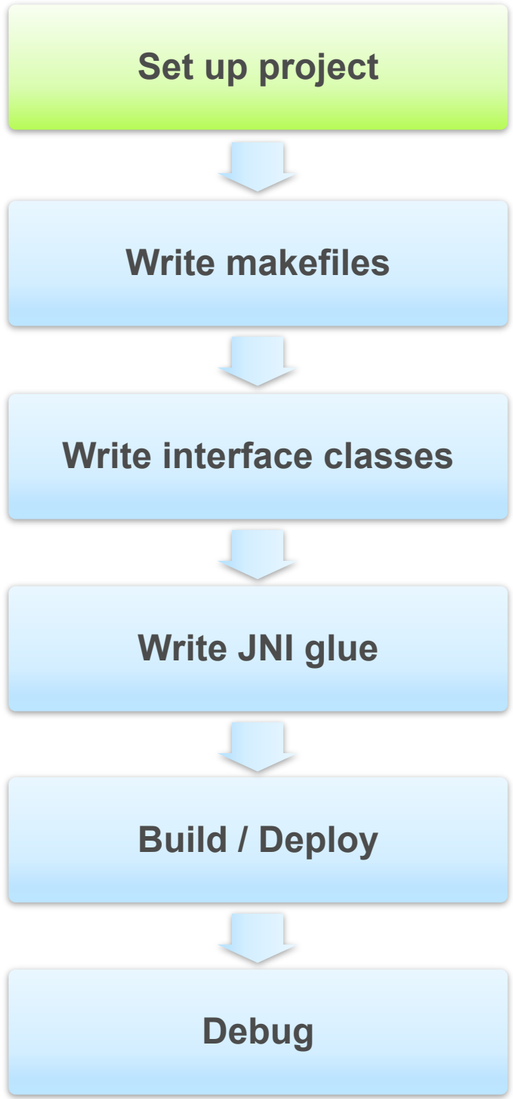
- Install Cygwin
  - Required for debugger
  - Recommended for build scripts



Development Flow



Development Flow: Project Setup

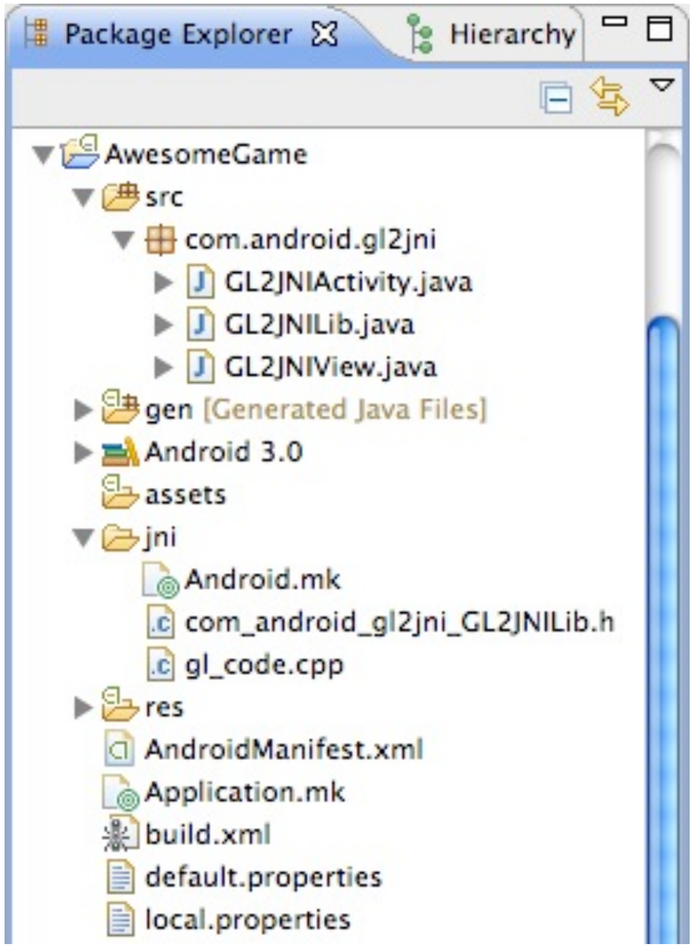


Run `android create project`

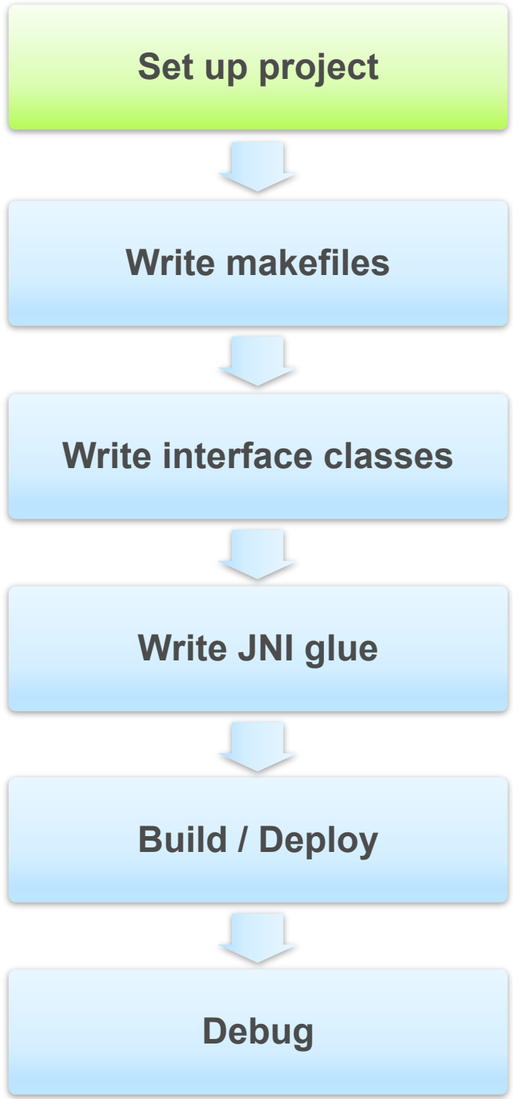
- Ant script
- Common directories
- Resources
- Per-machine settings

OR `android update project`

- For existing code



Development Flow: Project Setup



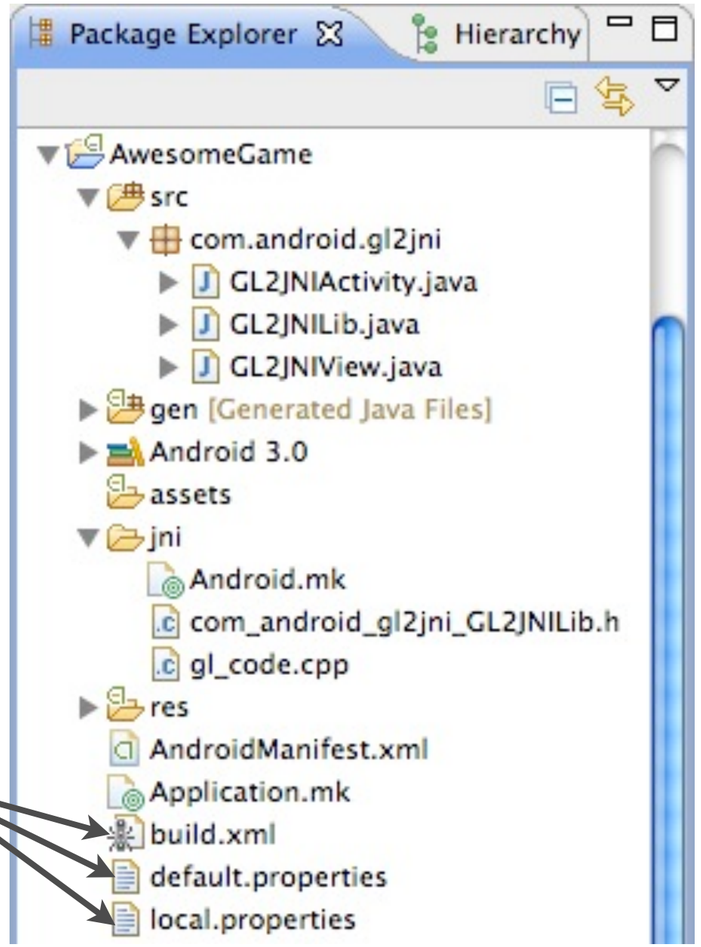
Run `android create project`

- Ant script
- Common directories
- Resources
- Per-machine settings

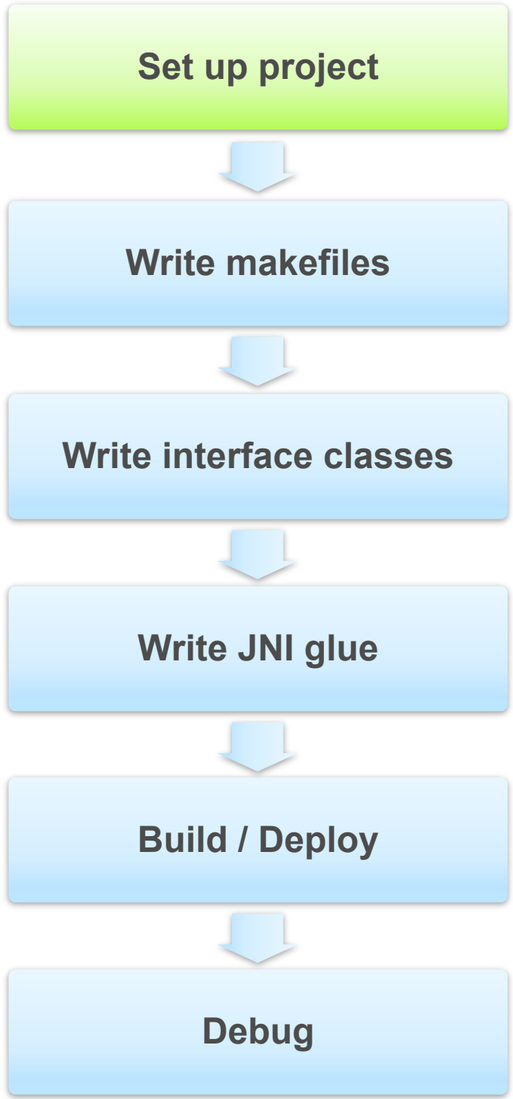
OR `android update project`

- For existing code

Created by `android create project`



Development Flow: Project Setup

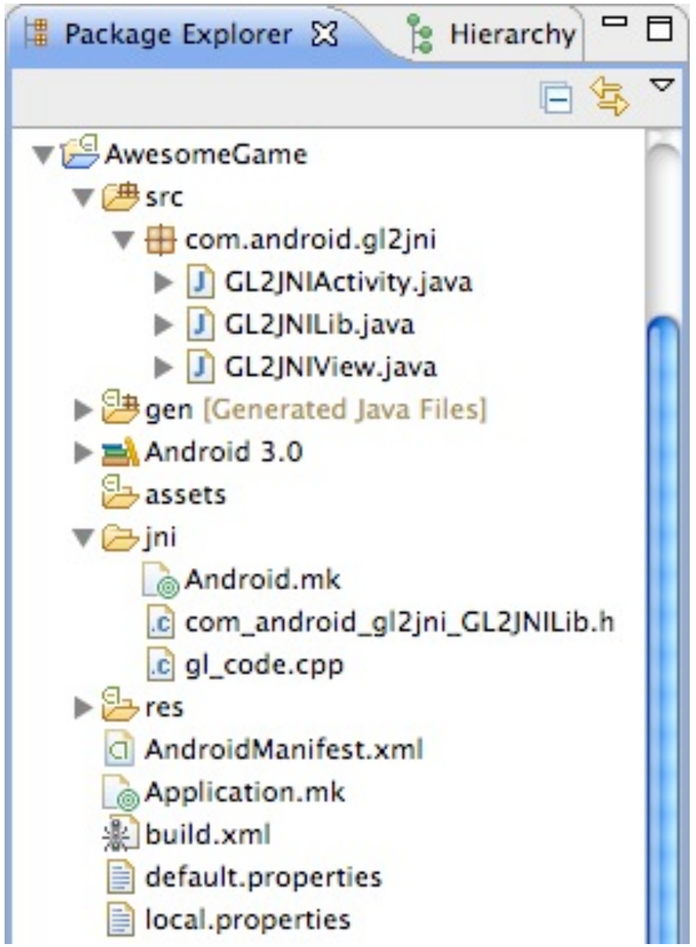


Run `android create project`

- Ant script
- Common directories
- Resources
- Per-machine settings

OR `android update project`

- For existing code



Development Flow: Project Setup

Set up project

Write makefiles

Write interface classes

Write JNI glue

Build / Deploy

Debug

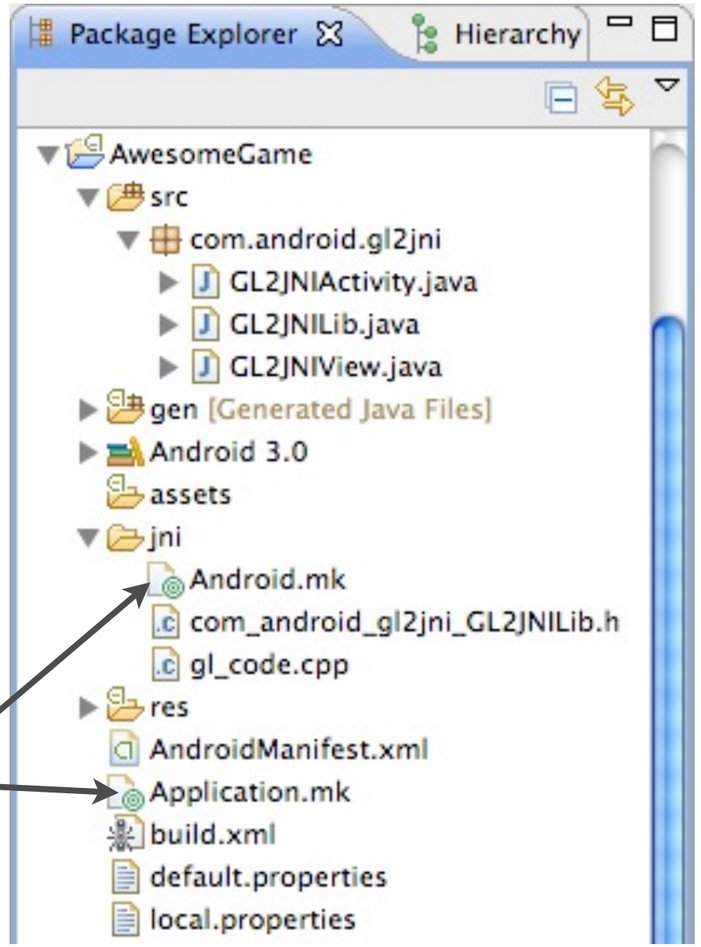
Run `android create project`

- Ant script
- Common directories
- Resources
- Per-machine settings

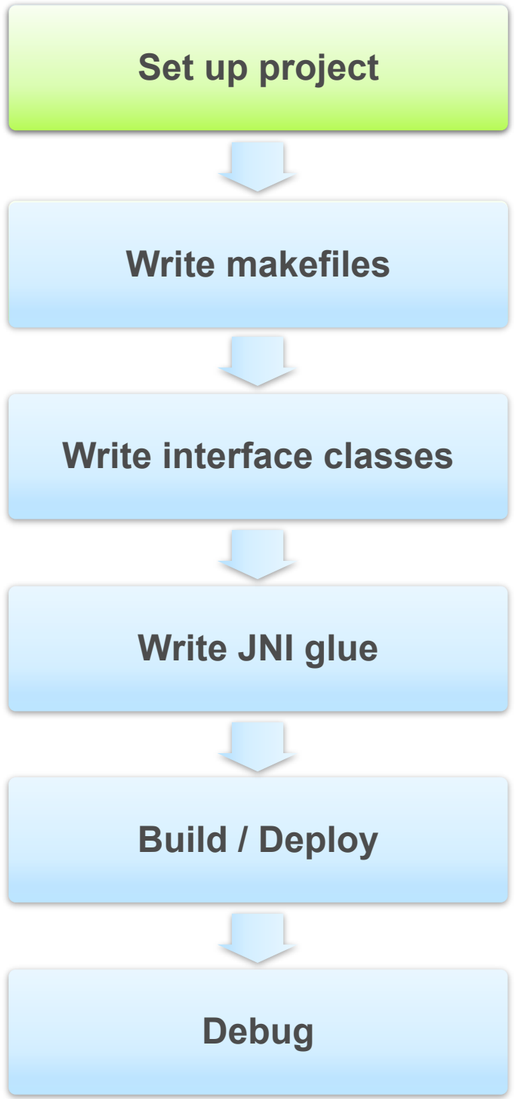
OR `android update project`

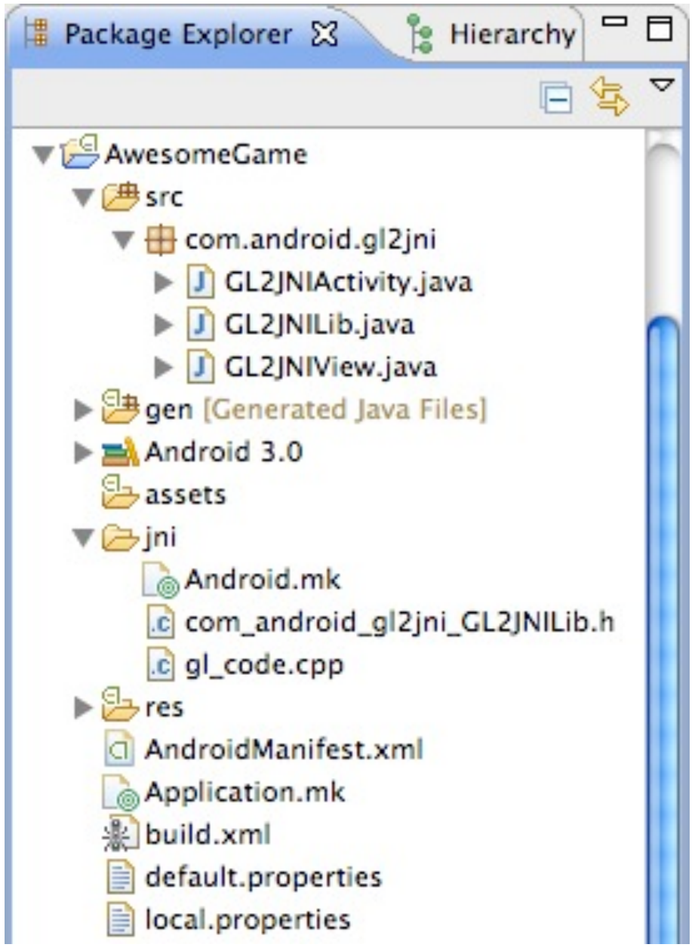
- For existing code

Created Manually



Development Flow: Makefiles

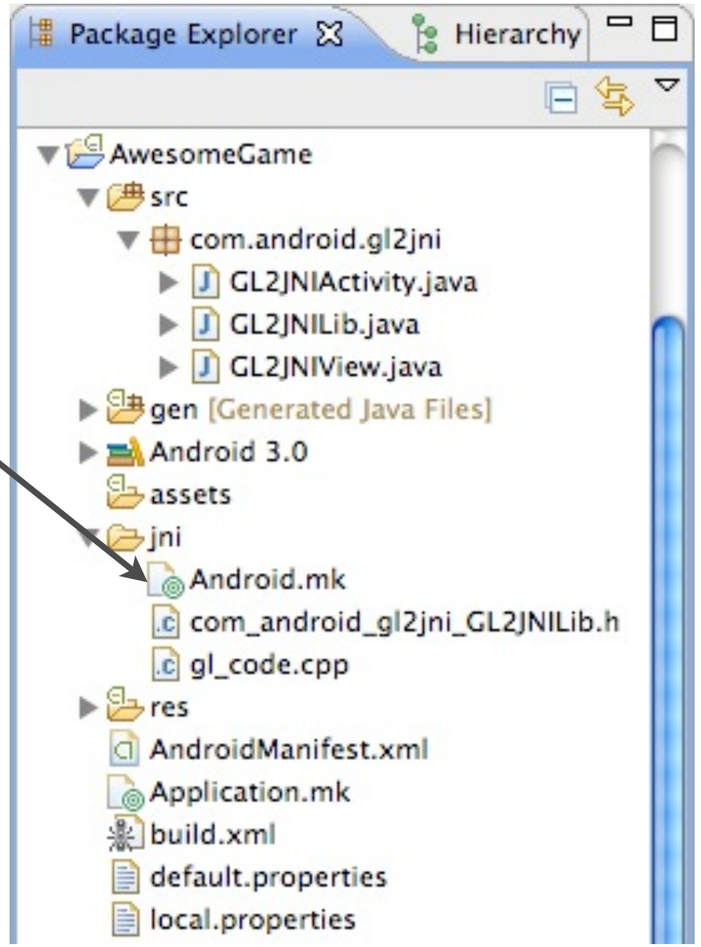




Development Flow: Makefiles

- Set up project
- Write makefiles
- Write interface classes
- Write JNI glue
- Build / Deploy
- Debug

```
LOCAL_PATH:= $(call my-dir)

include $(CLEAR_VARS)

LOCAL_MODULE    := libgl2jni
LOCAL_CFLAGS    := -Werror
LOCAL_SRC_FILES := gl_code.cpp
LOCAL_LDLIBS    := -llog -lGLESv2

include $(BUILD_SHARED_LIBRARY)
```

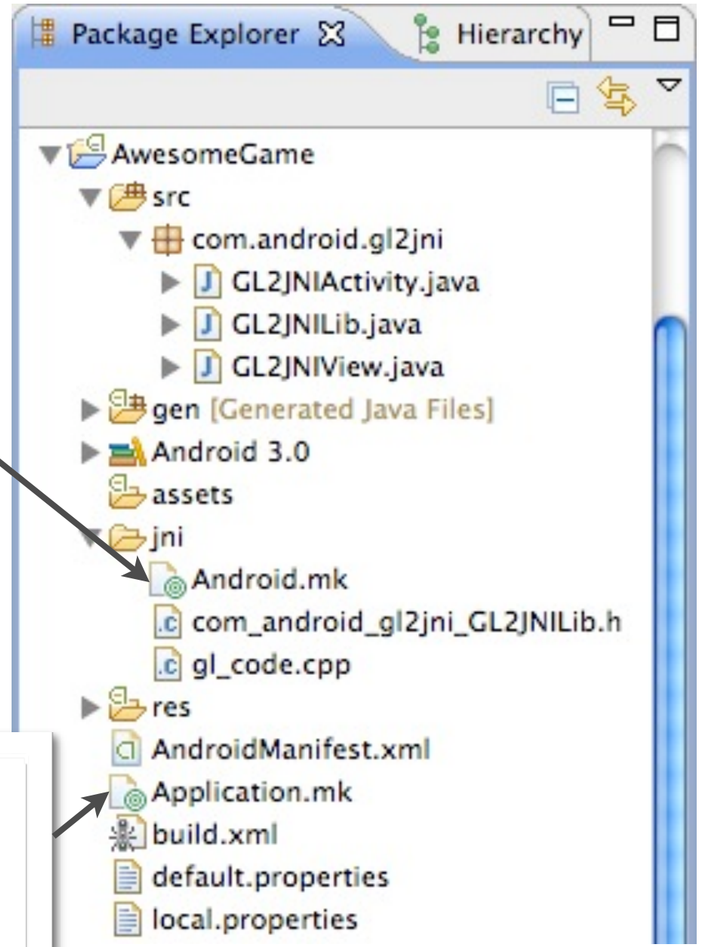


Development Flow: Makefiles

- Set up project
- Write makefiles
- Write interface classes
- Write JNI glue
- Build / Deploy
- Debug

```
LOCAL_PATH:= $(call my-dir)

include $(CLEAR_VARS)

LOCAL_MODULE    := libgl2jni
LOCAL_CFLAGS    := -Werror
LOCAL_SRC_FILES := gl_code.cpp
LOCAL_LDLIBS    := -llog -lGLESv2

include $(BUILD_SHARED_LIBRARY)
```

```
APP_PROJECT_PATH := $(call my-dir)/project
APP_MODULES      := libgl2jni
APP_OPTIM        := debug
APP_ABI          := armeabi armeabi-v7a
APP_STL          := gnustl_static
```

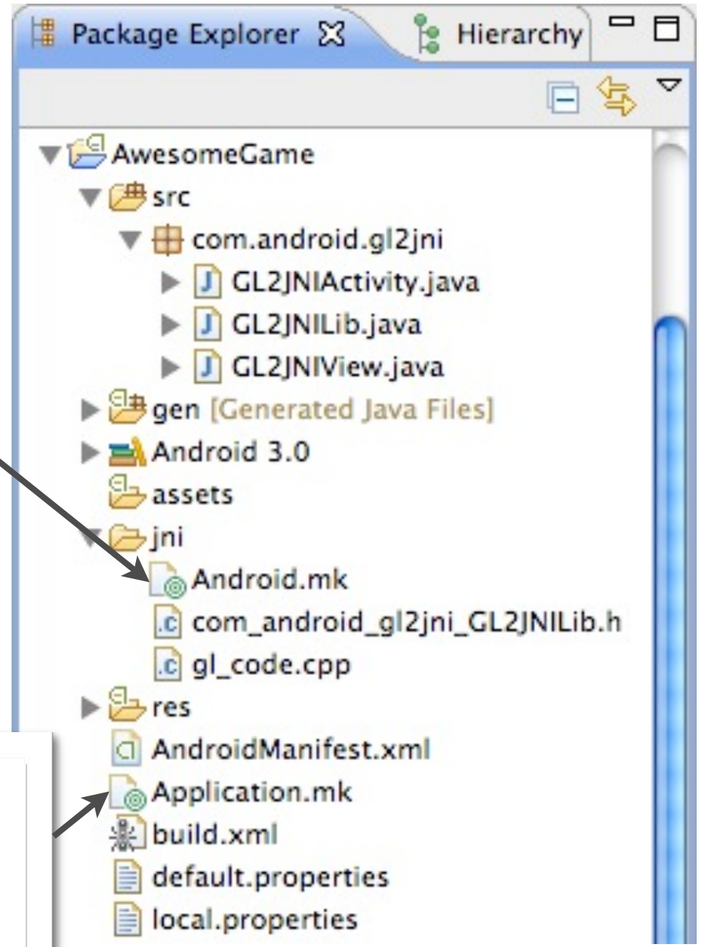


Development Flow: Makefiles

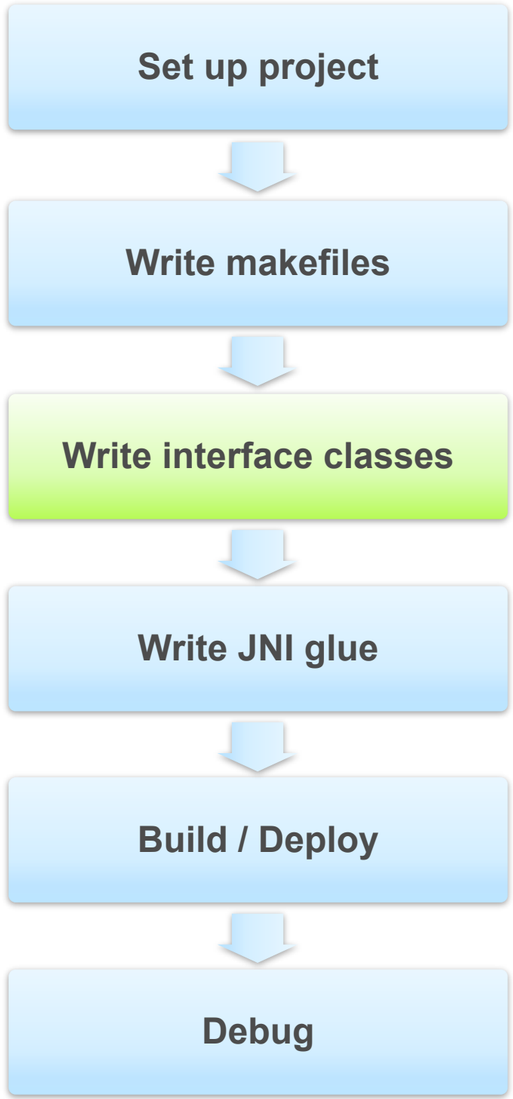
- Set up project
- Write makefiles
- Write interface classes
- Write JNI glue
- Build / Deploy
- Debug

```
LOCAL_PATH:= $(call my-dir)

include $(CLEAR_VARS)

LOCAL_MODULE    := libgl2jni
LOCAL_CFLAGS    := -Werror
LOCAL_SRC_FILES := gl_code.cpp
LOCAL_LDLIBS    := -llog -lGLESv2

include $(BUILD_SHARED_LIBRARY)
```

```
APP_PROJECT_PATH := $(call my-dir)/project
APP_MODULES      := libgl2jni
APP_OPTIM        := debug
APP_ABI          := armeabi armeabi-v7a
APP_STL          := gnustl_static
```



Development Flow: Interface Classes

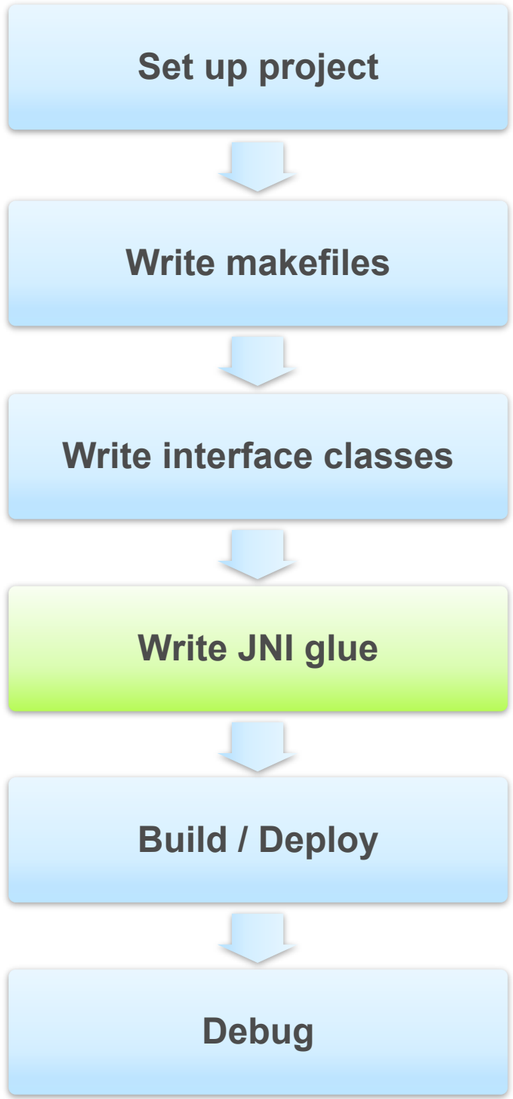


Write interface in Java

– Use **native** keyword

```
package com.android.gl2jni;

// Wrapper for native library

public class GL2JNILib extends Activity {

  static {
    System.loadLibrary("gl2jni");
  }

  /**
   * @param width the current view width
   * @param height the current view height
   */
   public native void init(int width, int height);
   public static native void step();
}
```

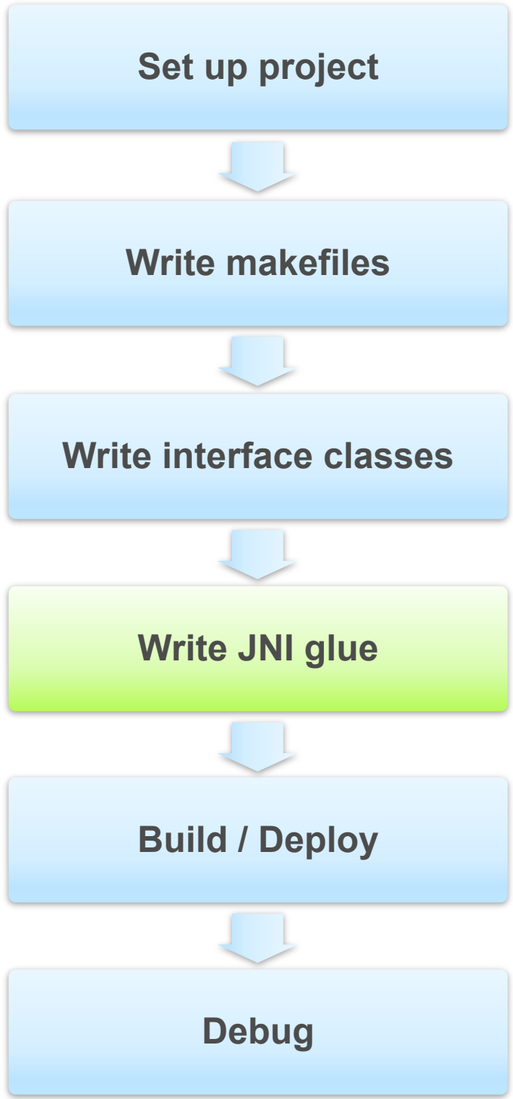
Development Flow: JNI Glue



Generate with javah tool

- `javah com.android.gl2jni.GL2JNILib`
- Eclipse: Source|Generate C files from Java class

Development Flow: JNI Glue

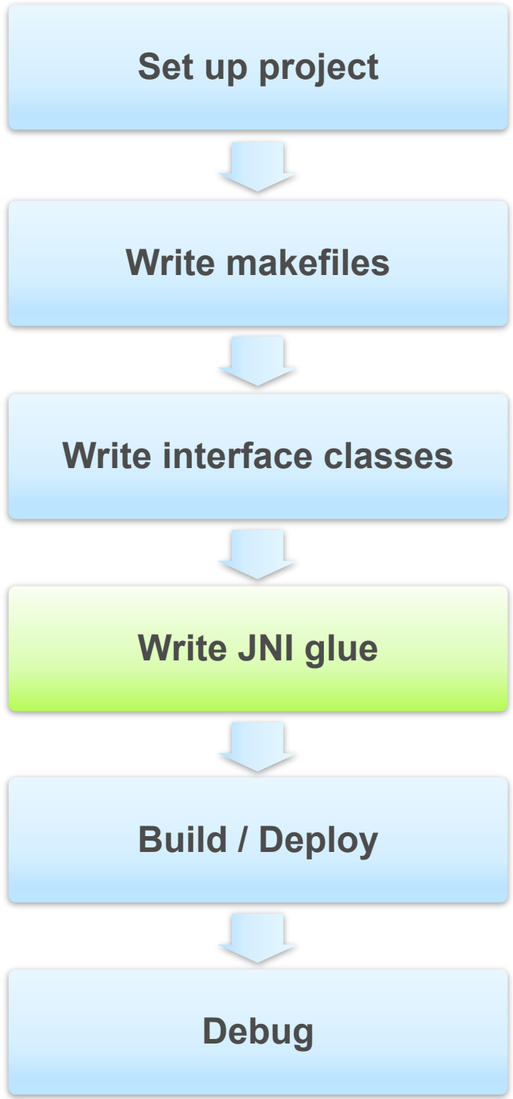


Generate with javah tool

- `javah com.android.gl2jni.GL2JNILib`
- Eclipse: Source|Generate C files from Java class

```
public native void init(int width, int height);
public static native void step();
```

Development Flow: JNI Glue



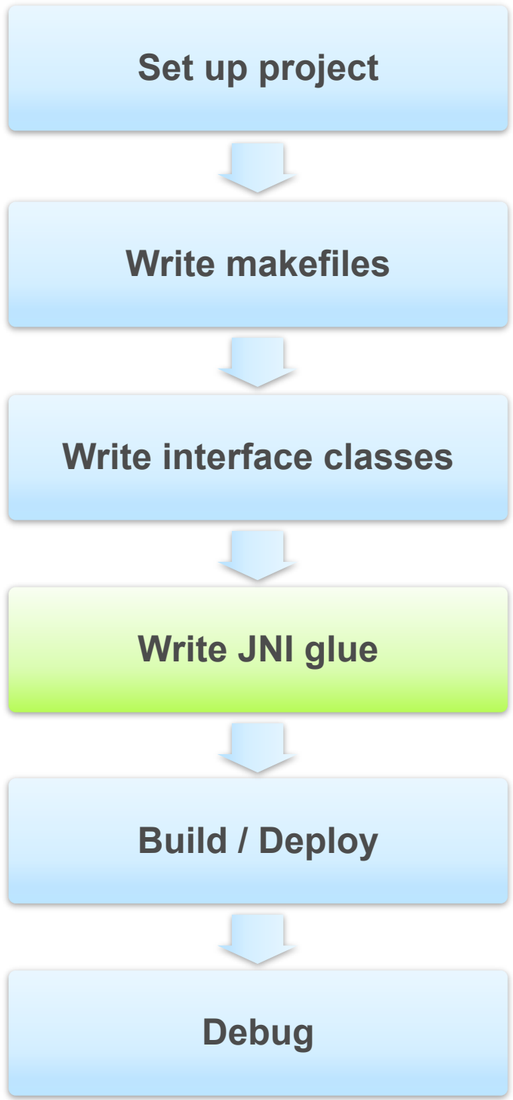
Generate with javah tool

- `javah com.android.gl2jni.GL2JNILib`
- Eclipse: Source|Generate C files from Java class

```
public native void init(int width, int height);
public static native void step();
```

```
#include <jni.h>
/*
 * Class:     com_android_gl2jni_GL2JNILib
 * Method:    init    Signature: (II)V
 */
JNIEXPORT void JNICALL Java_com_android_gl2jni_GL2JNILib_init
  (JNIEnv *, jobject, jint, jint);
/*
 * Class:     com_android_gl2jni_GL2JNILib
 * Method:    step    Signature: ()V
 */
JNIEXPORT void JNICALL Java_com_android_gl2jni_GL2JNILib_step
  (JNIEnv *, jclass);
```

Development Flow: JNI Glue



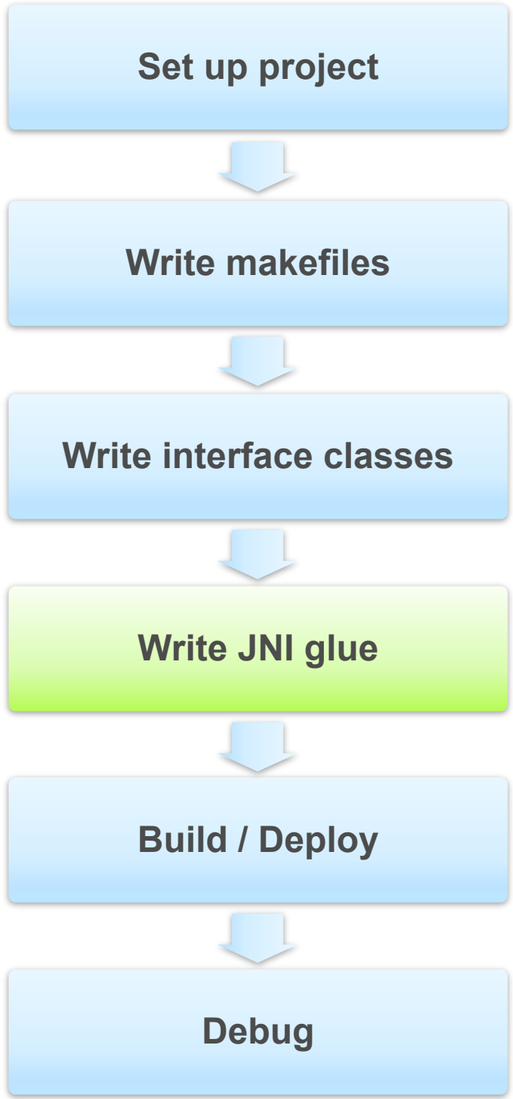
Generate with javah tool

- `javah com.android.gl2jni.GL2JNILib`
- Eclipse: Source|Generate C files from Java class

```
public native void init(int width, int height);
public static native void step();
```

```
#include <jni.h>
/*
 * Class:     com_android_gl2jni_GL2JNILib
 * Method:    init    Signature: (II)V
 */
JNIEXPORT void JNICALL Java_com_android_gl2jni_GL2JNILib_init
  (JNIEnv *, jobject, jint, jint);
/*
 * Class:     com_android_gl2jni_GL2JNILib
 * Method:    step    Signature: ()V
 */
JNIEXPORT void JNICALL Java_com_android_gl2jni_GL2JNILib_step
  (JNIEnv *, jclass);
```

Development Flow: JNI Glue



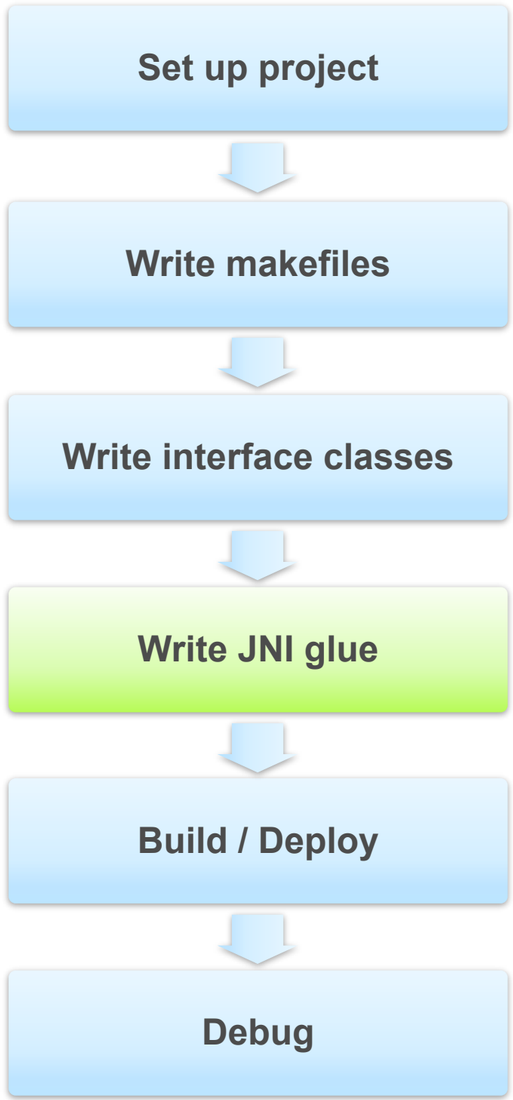
Generate with javah tool

- `javah com.android.gl2jni.GL2JNILib`
- Eclipse: Source|Generate C files from Java class

```
public native void init(int width, int height);
public static native void step();
```

```
#include <jni.h>
/*
 * Class:     com_android_gl2jni_GL2JNILib
 * Method:    init    Signature: (II)V
 */
JNIEXPORT void JNICALL Java_com_android_gl2jni_GL2JNILib_init
  (JNIEnv *, jobject, jint, jint);
/*
 * Class:     com_android_gl2jni_GL2JNILib
 * Method:    step    Signature: ()V
 */
JNIEXPORT void JNICALL Java_com_android_gl2jni_GL2JNILib_step
  (JNIEnv *, jclass);
```

Development Flow: JNI Glue



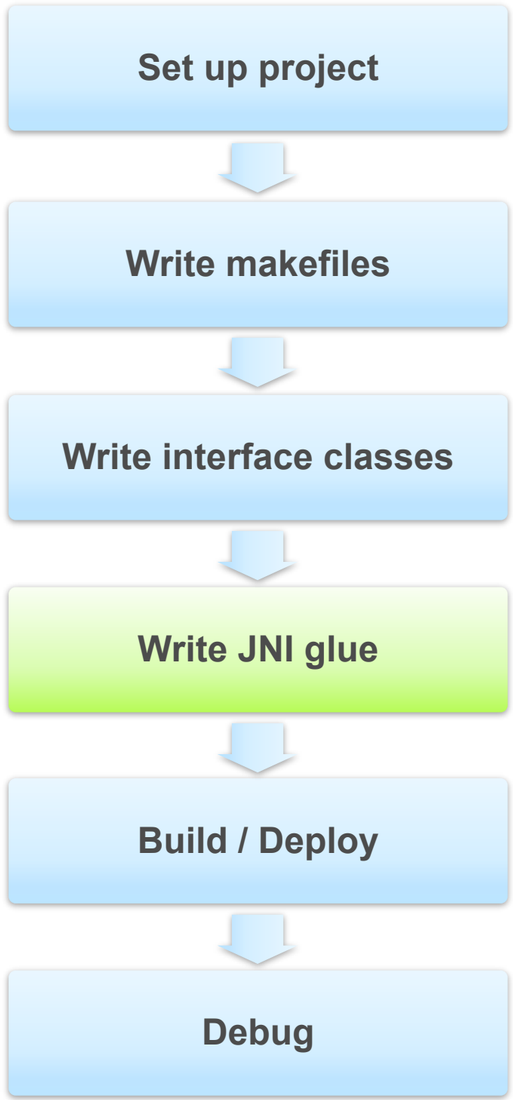
Generate with javah tool

- `javah com.android.gl2jni.GL2JNILib`
- Eclipse: Source|Generate C files from Java class

```
public native void init(int width, int height);
public static native void step();
```

```
#include <jni.h>
/*
 * Class:     com_android_gl2jni_GL2JNILib
 * Method:    init    Signature: (II)V
 */
JNIEXPORT void JNICALL Java_com_android_gl2jni_GL2JNILib_init
  (JNIEnv *, jobject, jint, jint);
/*
 * Class:     com_android_gl2jni_GL2JNILib
 * Method:    step    Signature: ()V
 */
JNIEXPORT void JNICALL Java_com_android_gl2jni_GL2JNILib_step
  (JNIEnv *, jclass);
```

Development Flow: JNI Glue



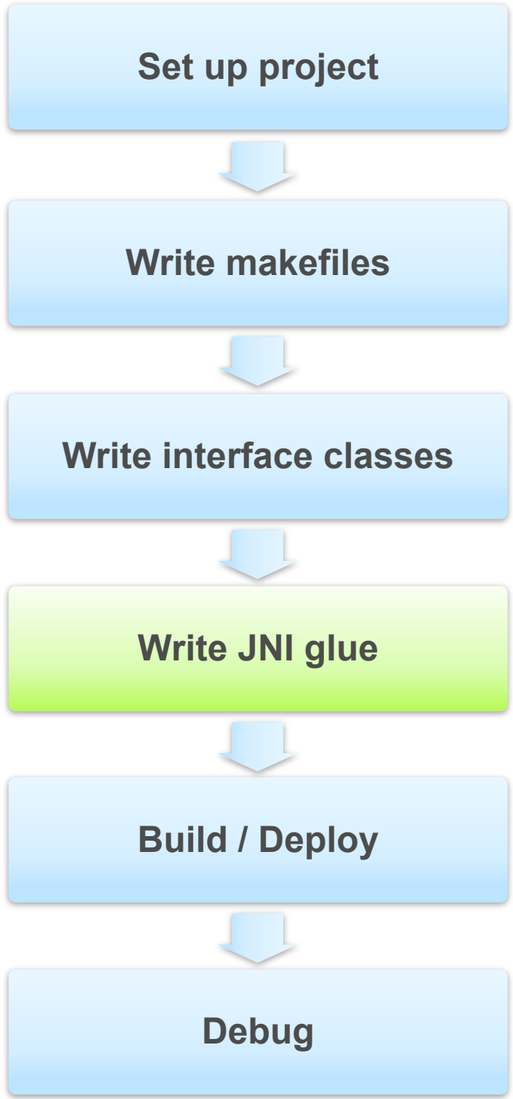
Generate with javah tool

- `javah com.android.gl2jni.GL2JNILib`
- Eclipse: Source|Generate C files from Java class

```
public native void init(int width, int height);
public static native void step();
```

```
#include <jni.h>
/*
 * Class:     com_android_gl2jni_GL2JNILib
 * Method:    init    Signature: (II)V
 */
JNIEXPORT void JNICALL Java_com_android_gl2jni_GL2JNILib_init
  (JNIEnv *, jobject, jint, jint);
/*
 * Class:     com_android_gl2jni_GL2JNILib
 * Method:    step    Signature: ()V
 */
JNIEXPORT void JNICALL Java_com_android_gl2jni_GL2JNILib_step
  (JNIEnv *, jclass);
```

Development Flow: JNI Glue



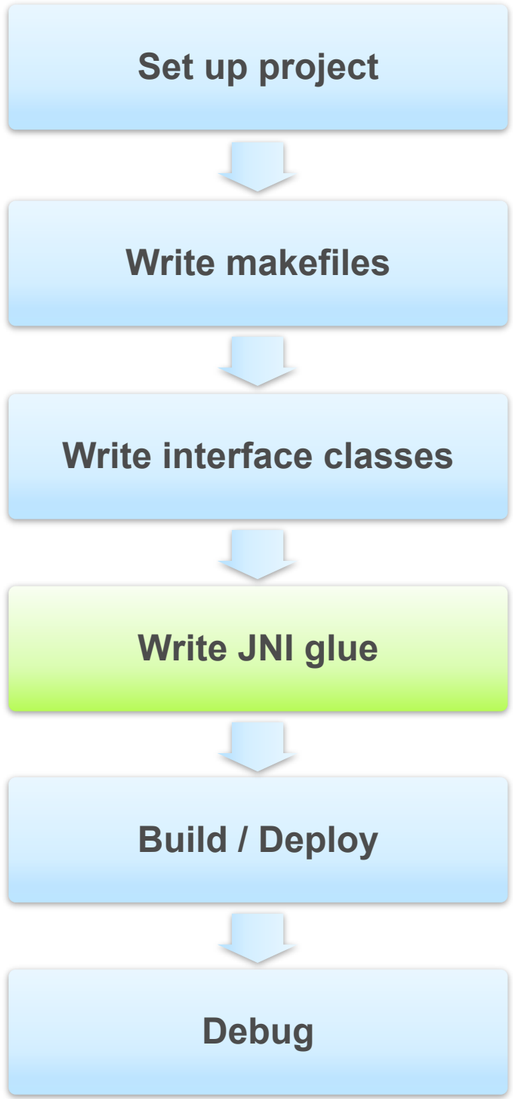
Generate with javah tool

- `javah com.android.gl2jni.GL2JNILib`
- Eclipse: Source|Generate C files from Java class

```
public native void init(int width, int height);
public static native void step();
```

```
#include <jni.h>
/*
 * Class:     com_android_gl2jni_GL2JNILib
 * Method:    init    Signature: (II)V
 */
JNIEXPORT void JNICALL Java_com_android_gl2jni_GL2JNILib_init
  (JNIEnv *, jobject, jint, jint);
/*
 * Class:     com_android_gl2jni_GL2JNILib
 * Method:    step    Signature: ()V
 */
JNIEXPORT void JNICALL Java_com_android_gl2jni_GL2JNILib_step
  (JNIEnv *, jclass);
```

Development Flow: JNI Glue



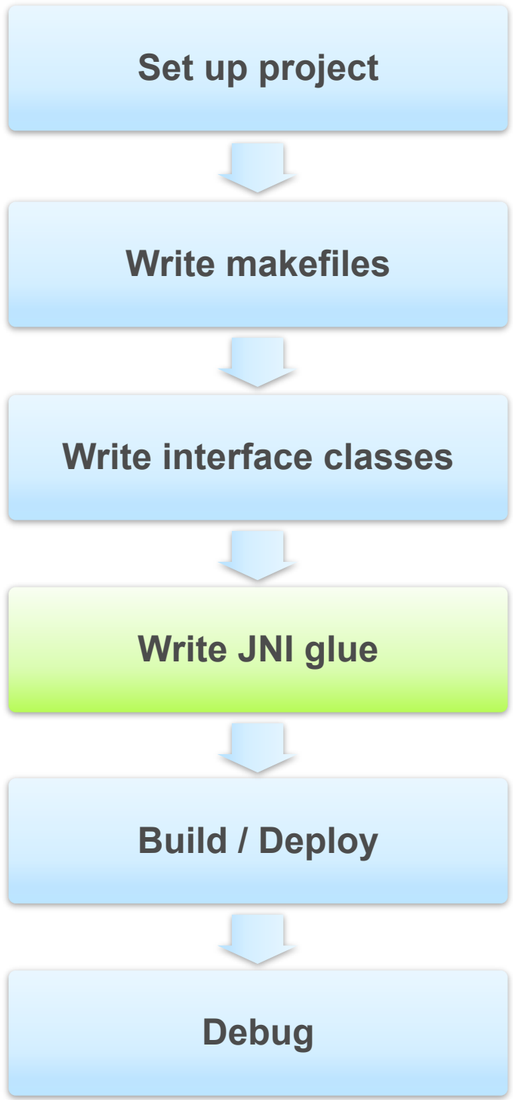
Generate with javah tool

- `javah com.android.gl2jni.GL2JNILib`
- Eclipse: Source|Generate C files from Java class

```
public native void init(int width, int height);
public static native void step();
```

```
#include <jni.h>
/*
 * Class:     com_android_gl2jni_GL2JNILib
 * Method:    init    Signature: (II)V
 */
JNIEXPORT void JNICALL Java_com_android_gl2jni_GL2JNILib_init
  (JNIEnv *, jobject, jint, jint);
/*
 * Class:     com_android_gl2jni_GL2JNILib
 * Method:    step    Signature: ()V
 */
JNIEXPORT void JNICALL Java_com_android_gl2jni_GL2JNILib_step
  (JNIEnv *, jclass);
```

Development Flow: JNI Glue



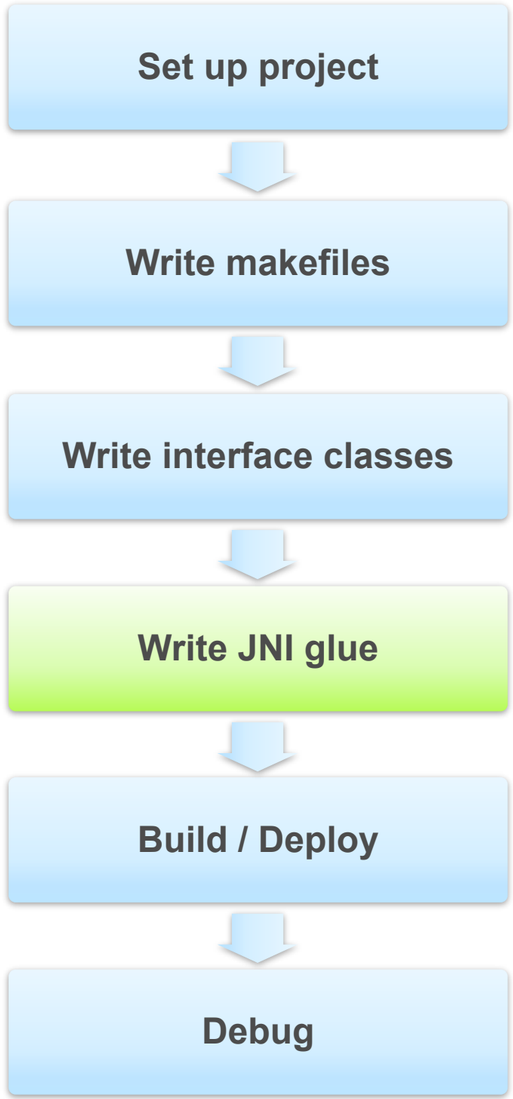
Generate with javah tool

- `javah com.android.gl2jni.GL2JNILib`
- Eclipse: Source|Generate C files from Java class

```
public native void init(int width, int height);
public static native void step();
```

```
#include <jni.h>
/*
 * Class:     com_android_gl2jni_GL2JNILib
 * Method:    init    Signature: (II)V
 */
JNIEXPORT void JNICALL Java_com_android_gl2jni_GL2JNILib_init
  (JNIEnv *, jobject, jint, jint);
/*
 * Class:     com_android_gl2jni_GL2JNILib
 * Method:    step    Signature: ()V
 */
JNIEXPORT void JNICALL Java_com_android_gl2jni_GL2JNILib_step
  (JNIEnv *, jclass);
```

Development Flow: JNI Glue with Manual Registration

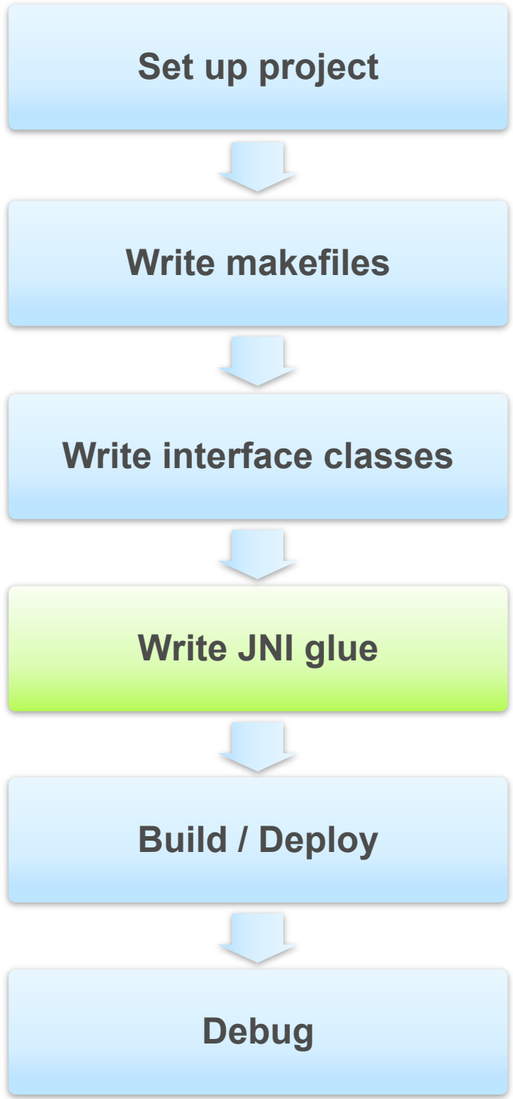



```
static JNINativeMethod gMethods[] = {
  { "init", "(II)V", (void*) nativeGameInit },
  { "step", "()V", (void*) nativeStep }
};
```

Get native signatures from class file with javap -s

- `javap -s -private com.android.gl2jni.GL2JNILib`

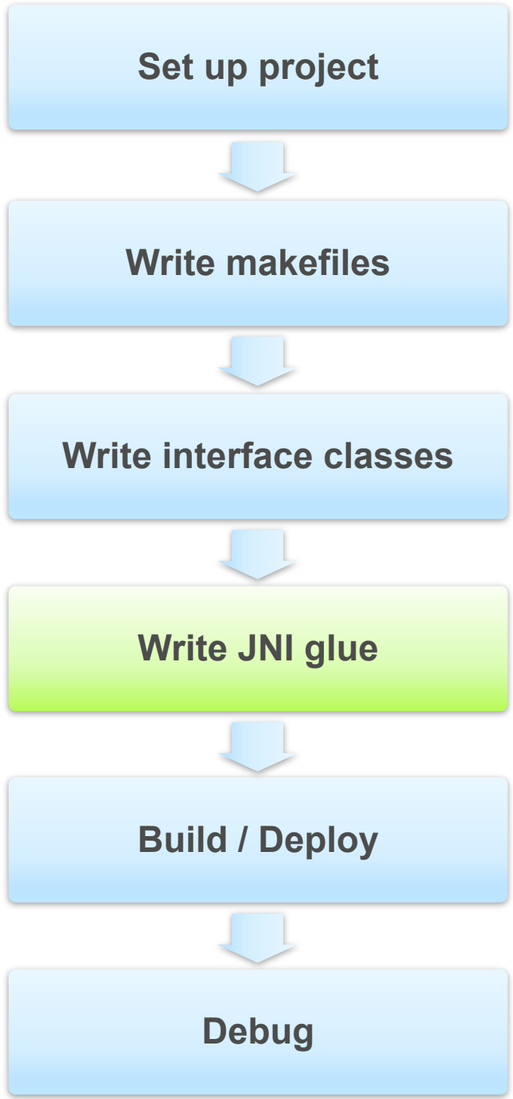
Development Flow: JNI Glue with Manual Registration




```
static JNINativeMethod gMethods[] = {
  { "init", "(II)V", (void*) nativeGameInit },
  { "step", "()V", (void*) nativeStep }
};
```

```
  char* className = "com/android/gl2jni/GL2JNILib";
  jclass clazz = env->FindClass(className);
  if (clazz == NULL) {
    __android_log_print(ANDROID_LOG_ERROR, "AwesomeGame",
      "Native registration unable to find class '%s'\n", className);
  return JNI_FALSE;
  }
  if (env->RegisterNatives(clazz, gMethods, numMethods) < 0) {
    __android_log_print(ANDROID_LOG_ERROR, "AwesomeGame",
      "RegisterNatives failed for '%s'\n", className);
  return JNI_FALSE;
  }
```

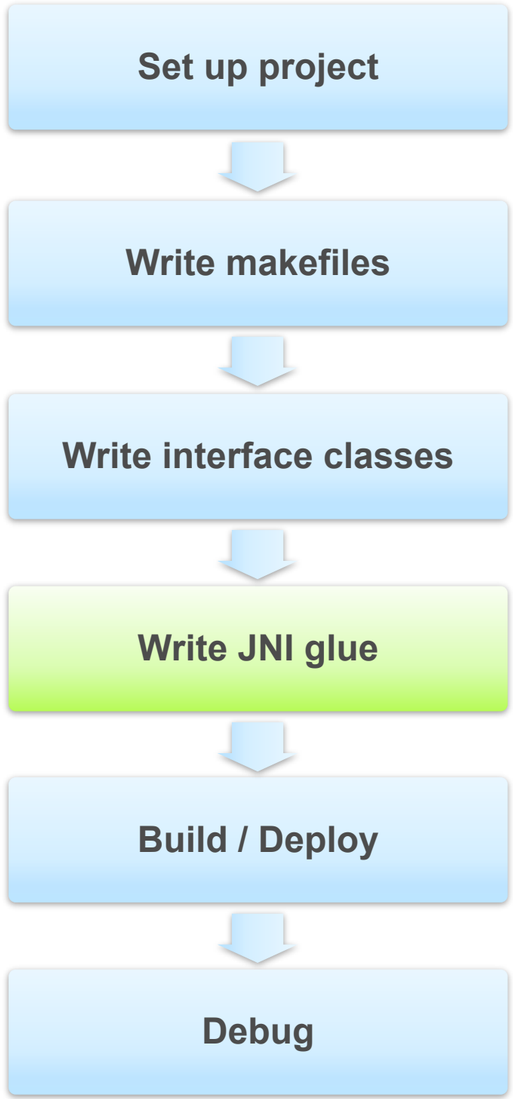
Development Flow: JNI Glue with Manual Registration




```
static JNINativeMethod gMethods[] = {
  { "init", "(II)V", (void*) nativeGameInit },
  { "step", "()V", (void*) nativeStep }
};
```

```
  char* className = "com/android/gl2jni/GL2JNILib";
  jclass clazz = env->FindClass(className);
  if (clazz == NULL) {
    __android_log_print(ANDROID_LOG_ERROR, "AwesomeGame",
      "Native registration unable to find class '%s'\n", className);
  return JNI_FALSE;
  }
  if (env->RegisterNatives(clazz, gMethods, numMethods) < 0) {
    __android_log_print(ANDROID_LOG_ERROR, "AwesomeGame",
      "RegisterNatives failed for '%s'\n", className);
  return JNI_FALSE;
  }
```

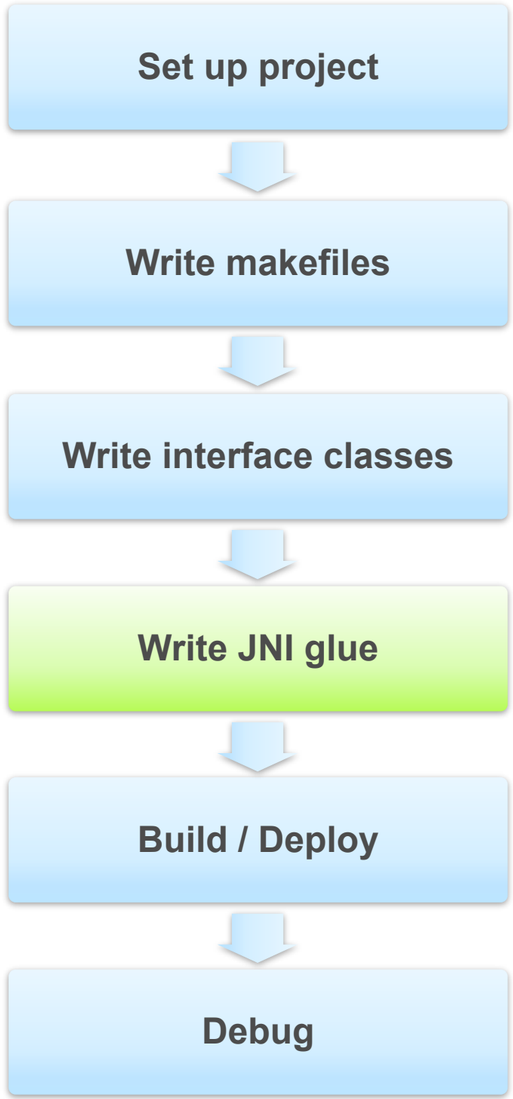
Development Flow: JNI Glue with Manual Registration




```
static JNINativeMethod gMethods[] = {
  { "init", "(II)V", (void*) nativeGameInit },
  { "step", "()V", (void*) nativeStep }
};
```

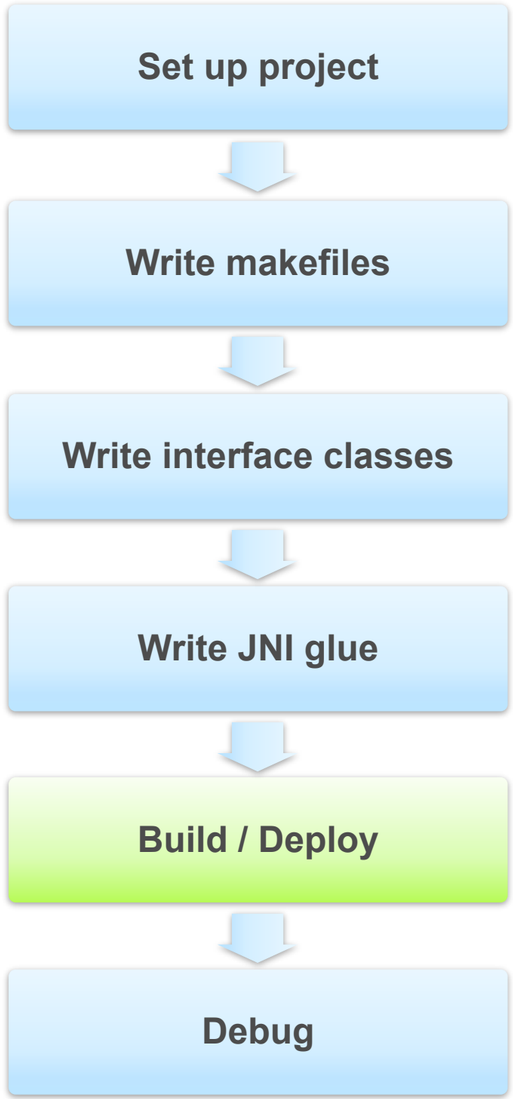
```
  char* className = "com/android/gl2jni/GL2JNILib";
  jclass clazz = env->FindClass(className);
  if (clazz == NULL) {
    __android_log_print(ANDROID_LOG_ERROR, "AwesomeGame",
      "Native registration unable to find class '%s'\n", className);
  return JNI_FALSE;
  }
  if (env->RegisterNatives(clazz, gMethods, numMethods) < 0) {
     __android_log_print(ANDROID_LOG_ERROR, "AwesomeGame",
       "RegisterNatives failed for '%s'\n", className);
  return JNI_FALSE;
  }
```

Development Flow: Calling Dalvik Code Within C/C++



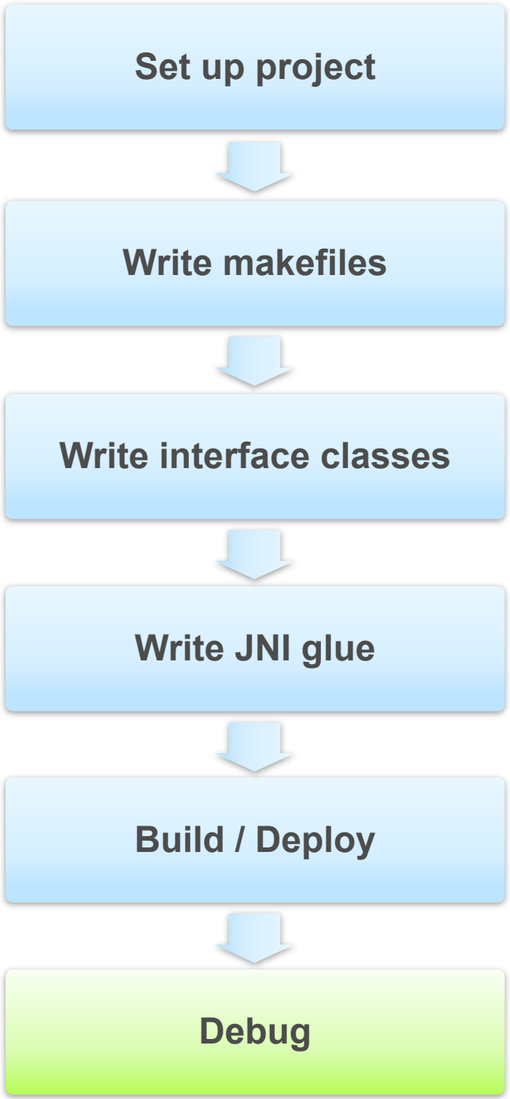

```
JNIEXPORT void JNICALL Java_com_android_gl2jni_GL2JNILib_init
  (JNIEnv *pJNIEnv, jobject obj, jint a, jint b) {

  jclass cls = (*pJNIEnv)->GetObjectClass(pJNIEnv, obj);

  jmethodID id = (*pJNIEnv)->GetMethodID(
                                pJNIEnv, cls, "takeKeyEvents", "(Z)V");

  (*pJNIEnv)->CallVoidMethod(pJNIEnv, obj, id, JNI_TRUE);
}
```

Development Flow: Build and Deploy

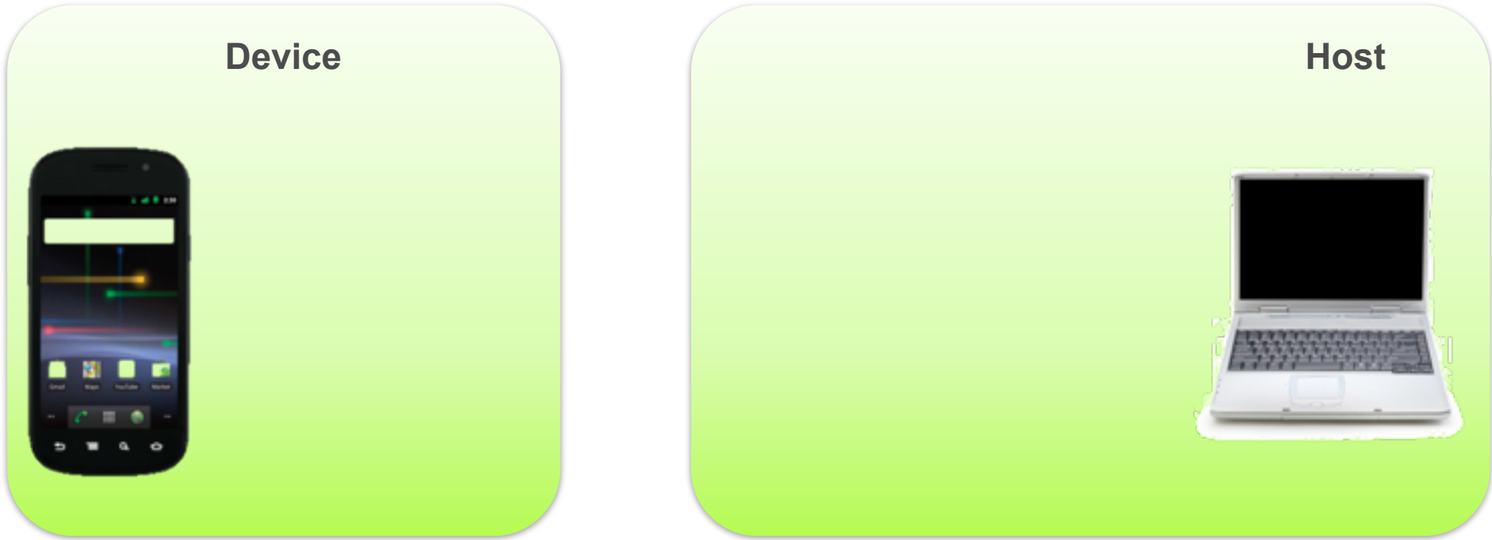


- `> [bash] ndk-build [NDK_DEBUG=1]`
  - Builds C libraries and emits .so files
- `> ant debug`
  - (or ant release)
  - Builds Java classes, apk package
- `> ant install`
  - Deploys to emulator or device

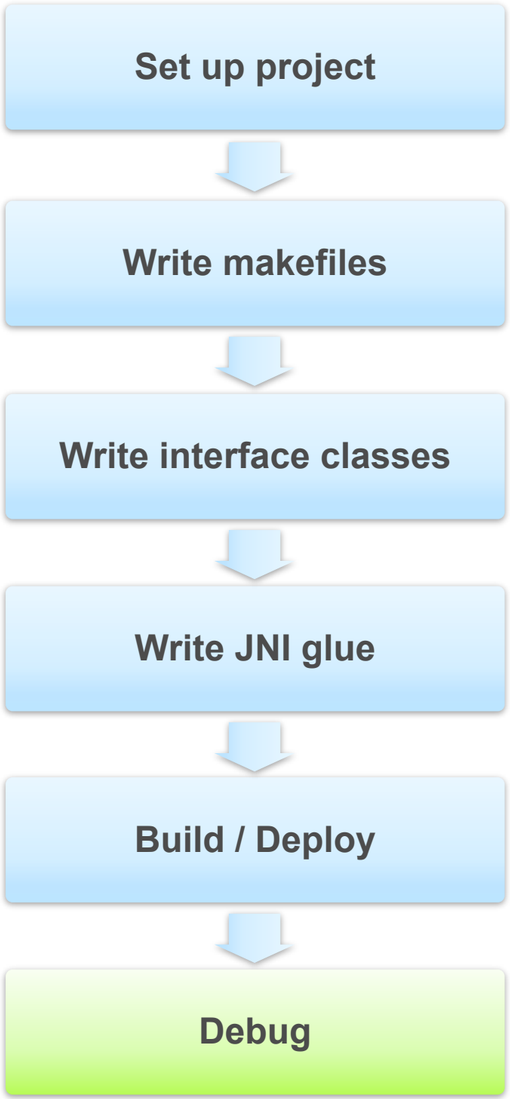
Development Flow: Debug



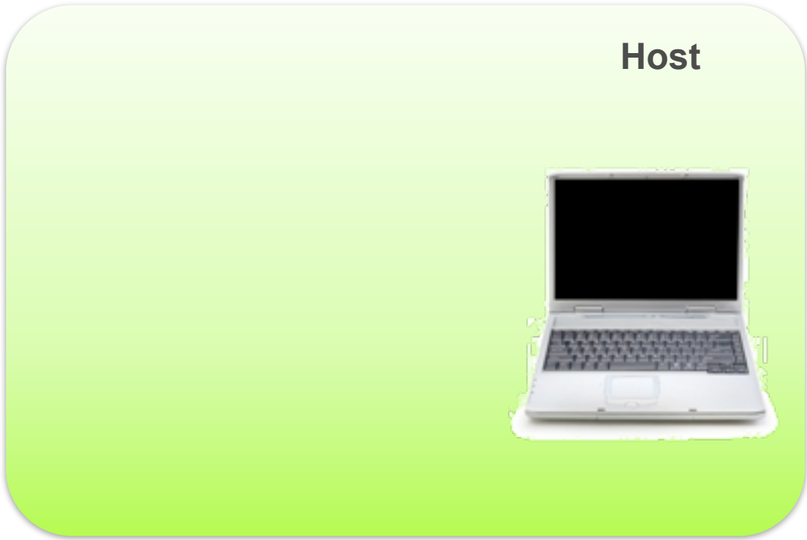
- `[bash] ndk-gdb`
  - Sanity checks your setup
  - Starts gdbserver on device, opens named pipe
  - Forwards pipe to socket on host
  - Starts gdb client on host, connects to socket



Development Flow: Debug



- `[bash] ndk-gdb`
  - Sanity checks your setup
  - Starts gdbserver on device, opens named pipe
  - Forwards pipe to socket on host
  - Starts gdb client on host, connects to socket



Development Flow: Debug

Set up project

Write makefiles

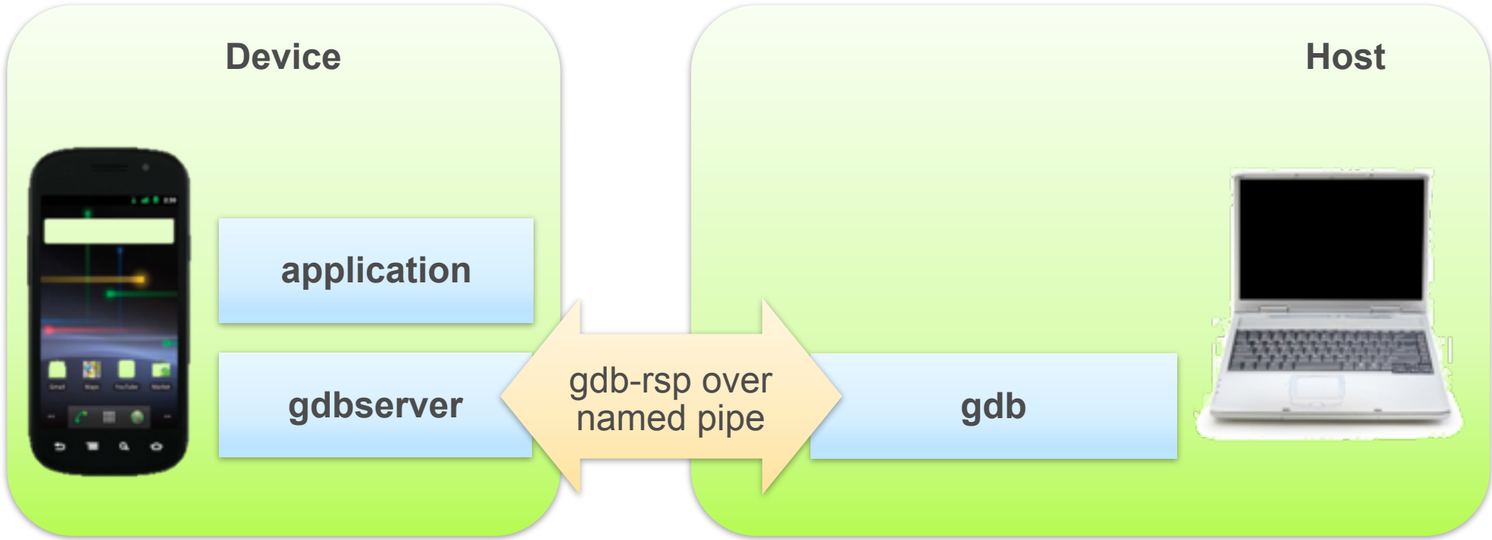
Write interface classes

Write JNI glue

Build / Deploy

Debug

- `[bash] ndk-gdb`
  - Sanity checks your setup
  - Starts gdbserver on device, opens named pipe
  - Forwards pipe to socket on host
  - Starts gdb client on host, connects to socket



Development Flow: Debug

Set up project

Write makefiles

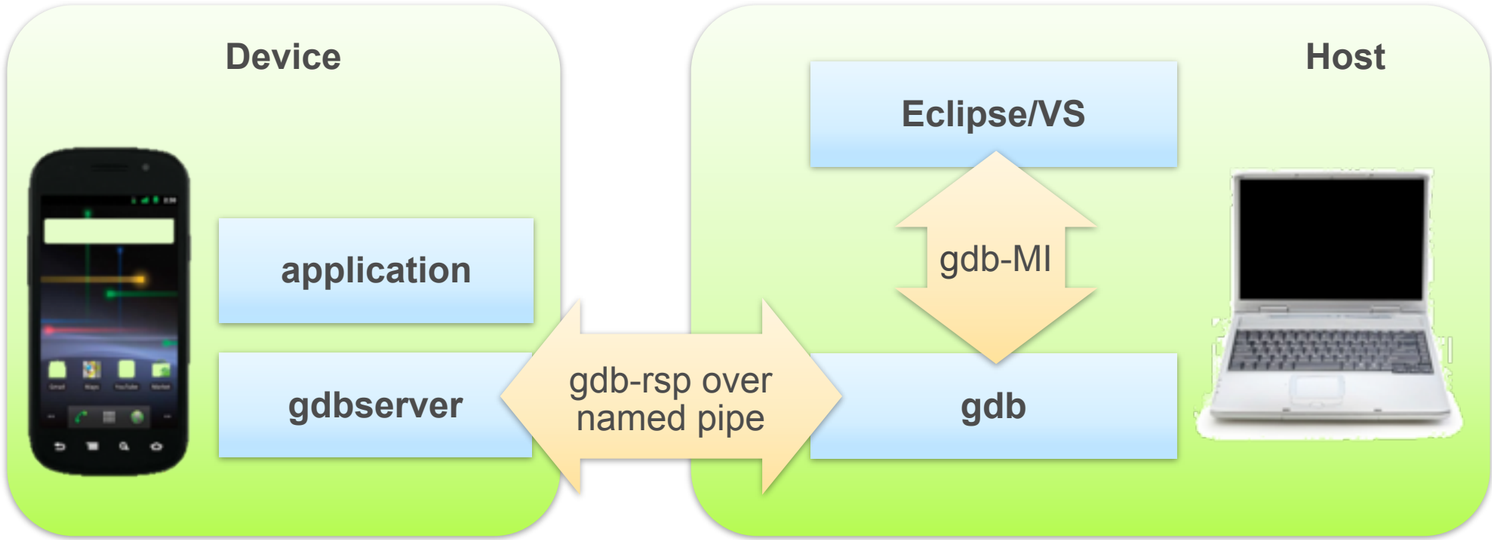
Write interface classes

Write JNI glue

Build / Deploy

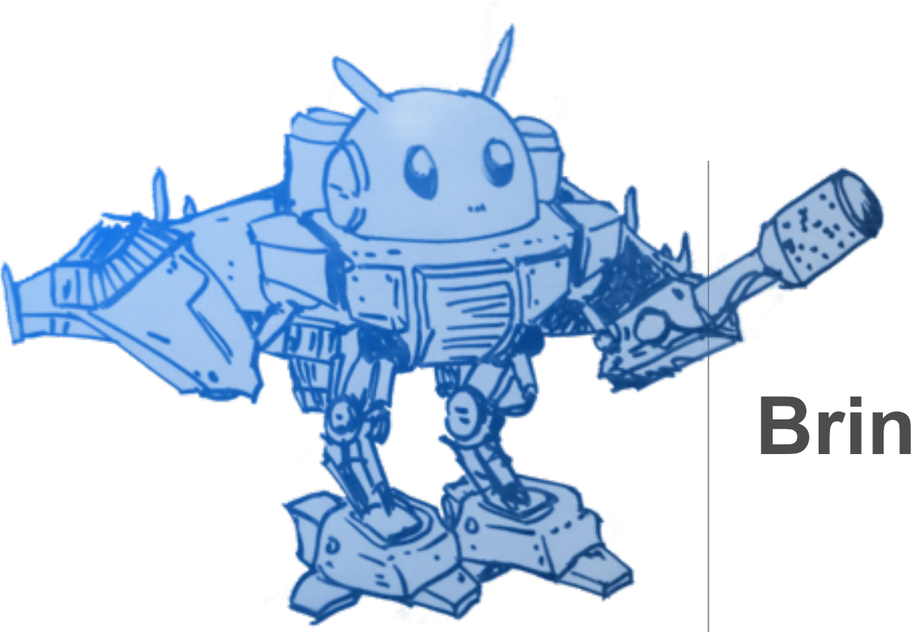
Debug

- `[bash] ndk-gdb`
  - Sanity checks your setup
  - Starts gdbserver on device, opens named pipe
  - Forwards pipe to socket on host
  - Starts gdb client on host, connects to socket



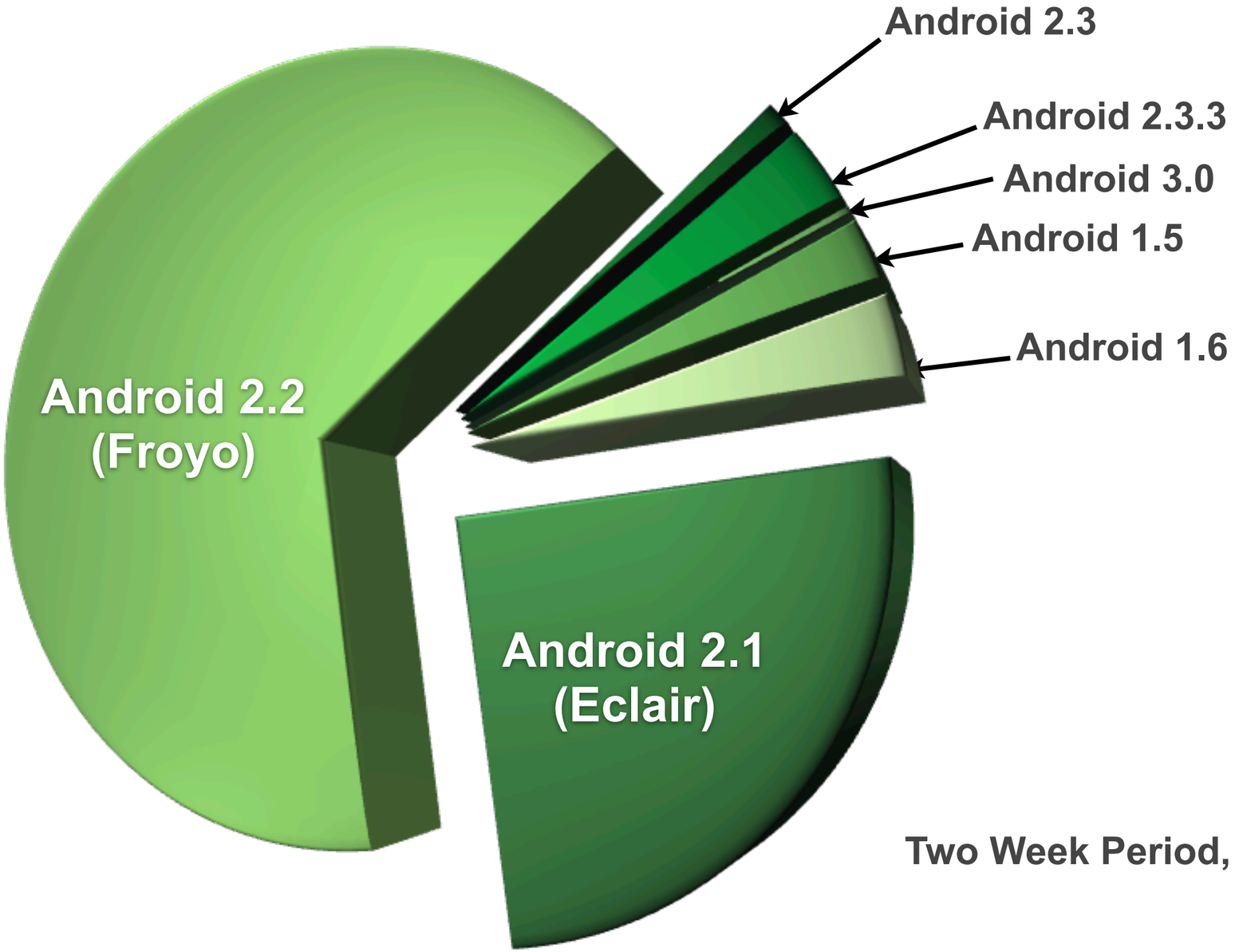
Important Tip:

Debug on Gingerbread +



Bringing Your Game to Android

Devices Accessing Android Market

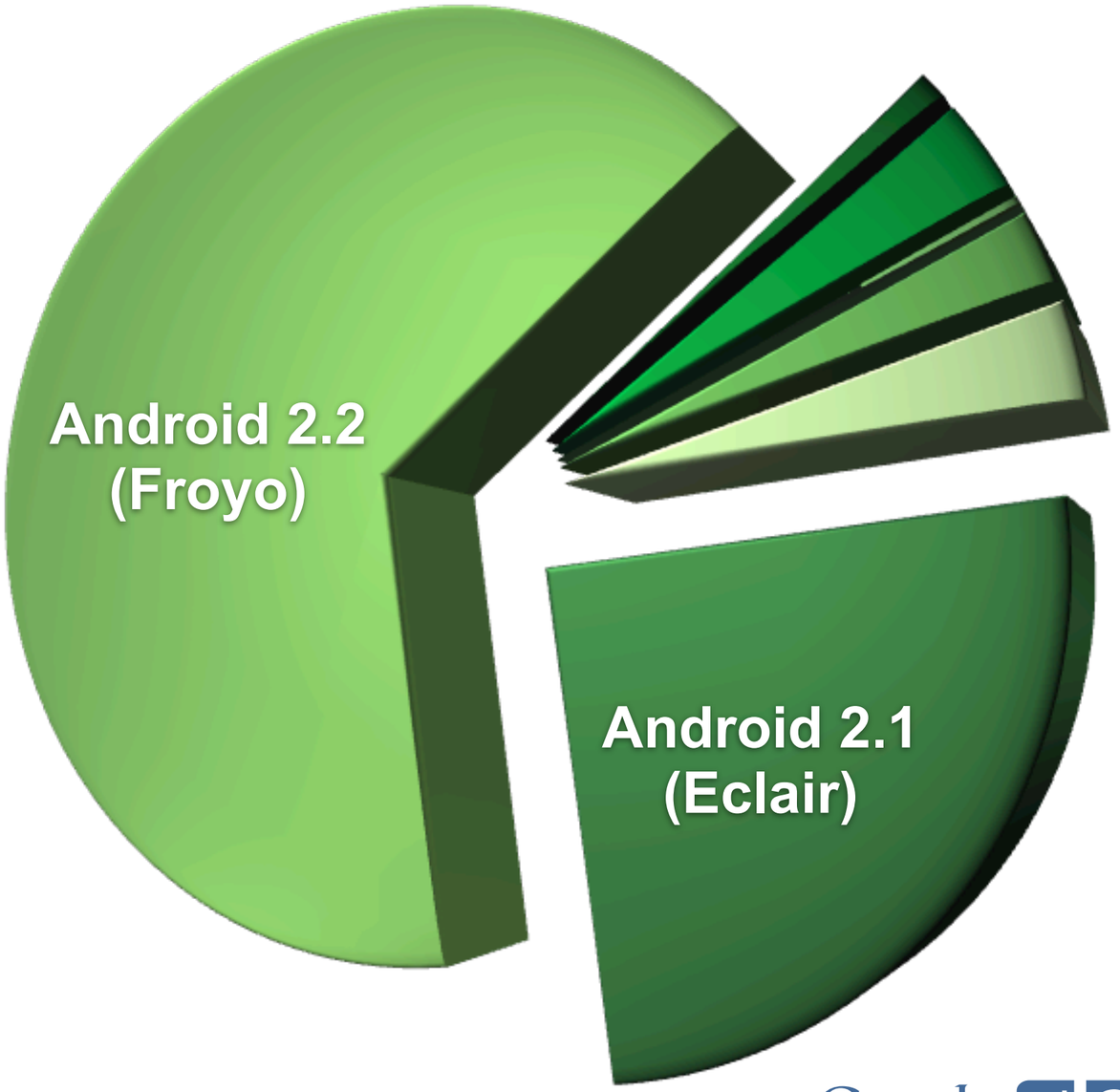


Two Week Period, Published May 2011

Devices Accessing Android Market

Great uptake for Android 2.2

- In ten months, 65.9% of devices checking in to Market running Froyo
- 94.7% of devices running Android 2.1 or higher



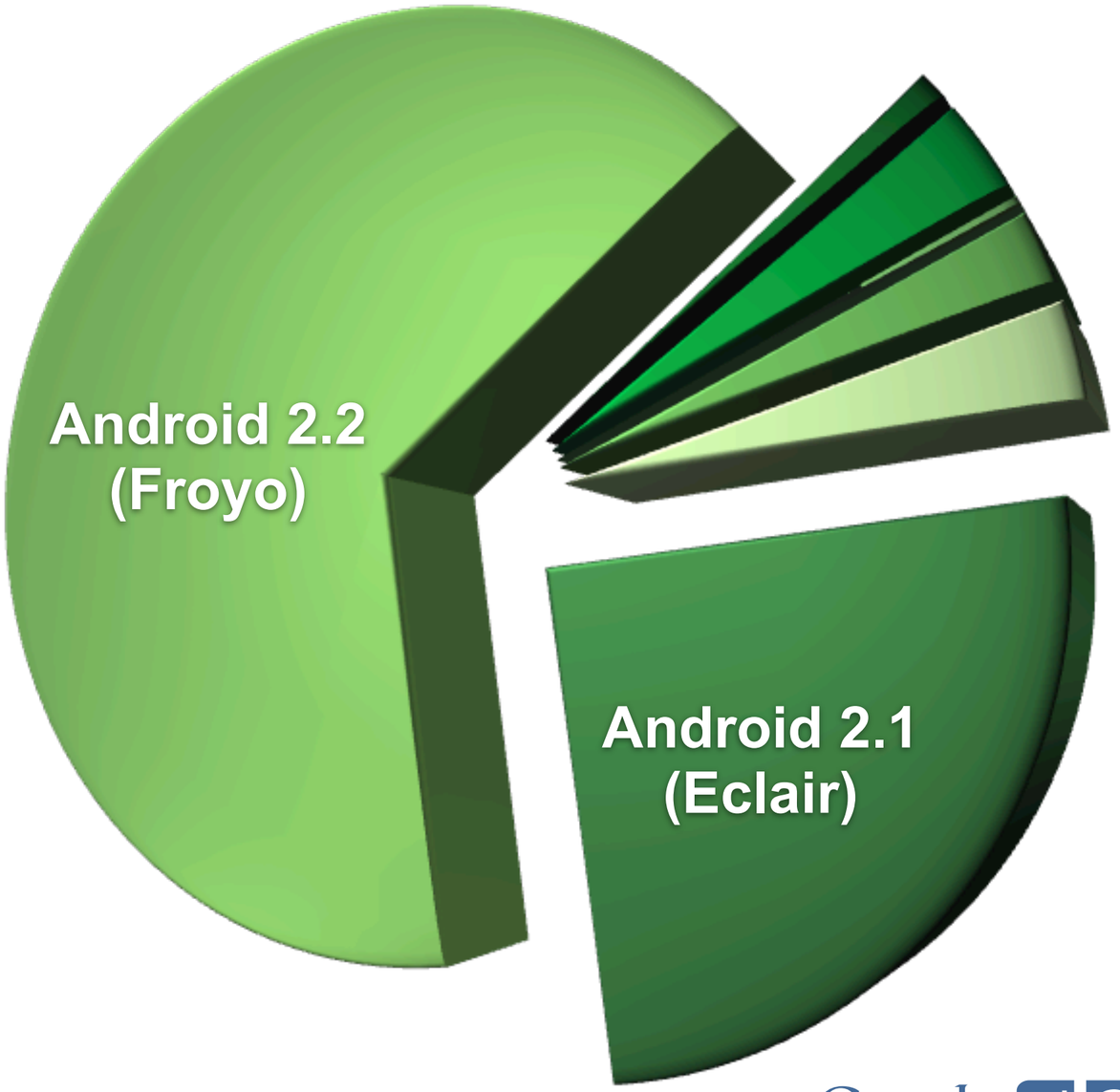
Devices Accessing Android Market

Great uptake for Android 2.2

- In ten months, 65.9% of devices checking in to Market running Froyo
- 94.7% of devices running Android 2.1 or higher

NativeActivity is just starting

- 4.3% of checking-in devices can utilize it
- For non-tablet-specific titles, we can take advantage of NativeActivity without becoming incompatible

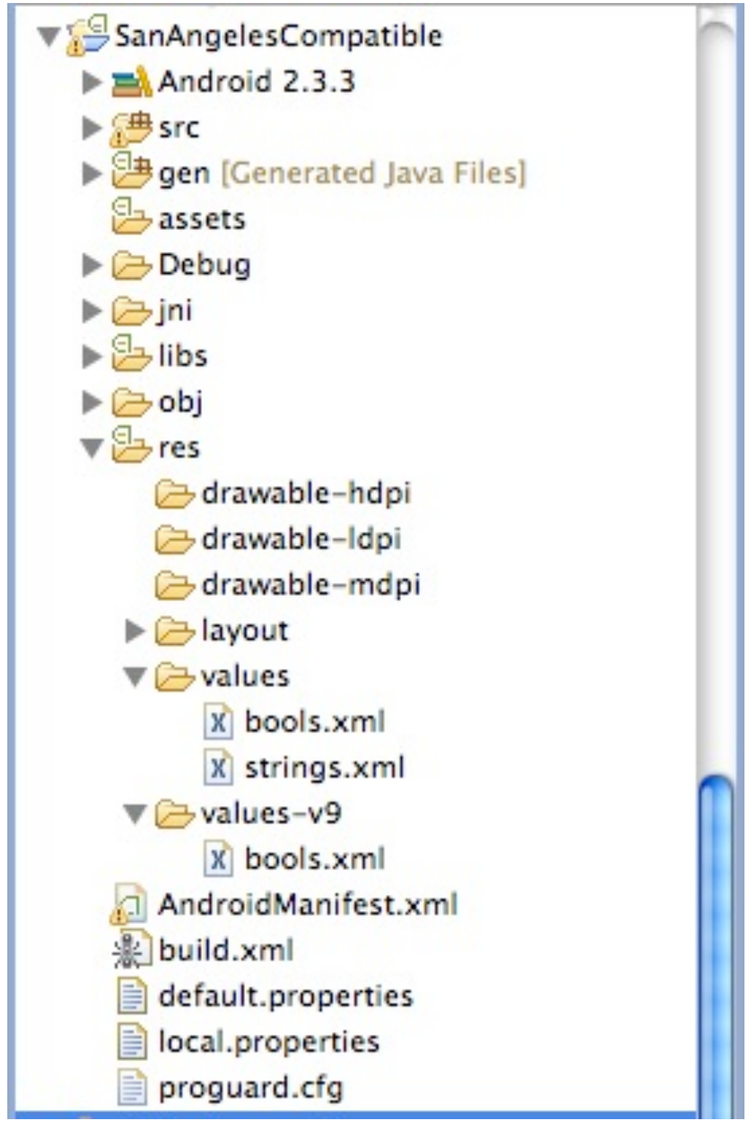


Using NativeActivity Compatibly

Platform-specific resources

– can be used to define boolean values

- activities can be enabled based upon these values



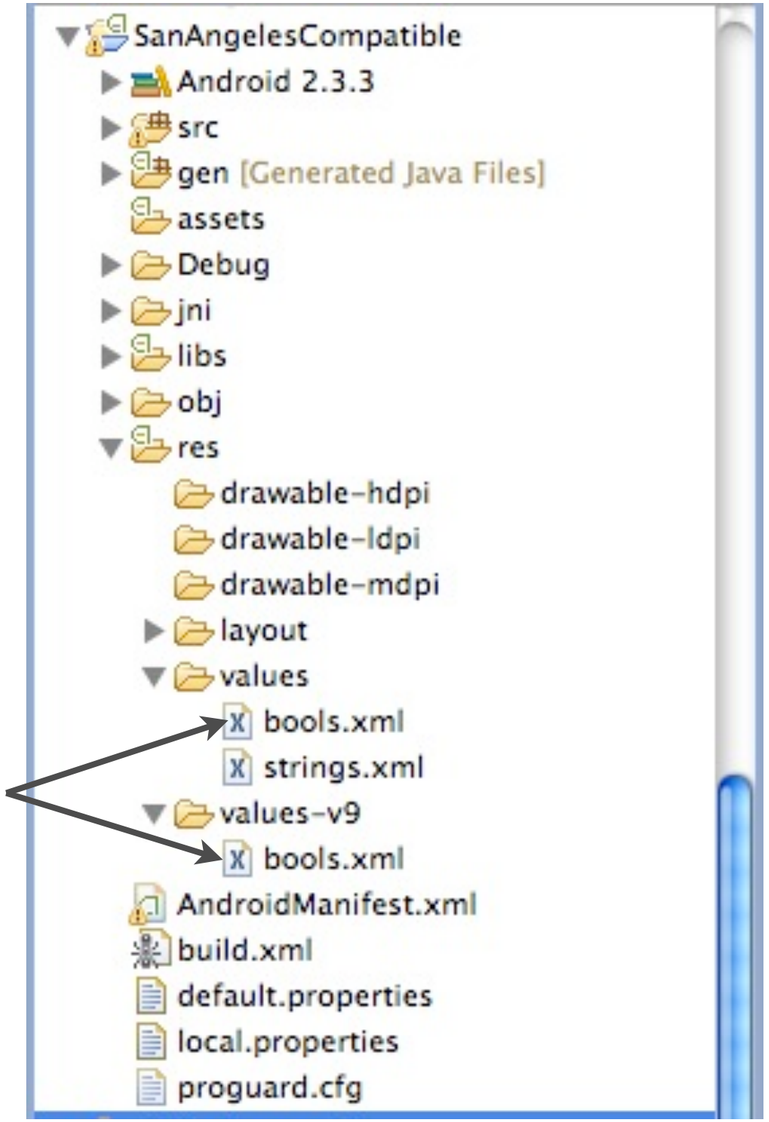
Using NativeActivity Compatibly

Platform-specific resources

– can be used to define boolean values

- activities can be enabled based upon these values

Platform-specific Resources



Using NativeActivity Compatibly

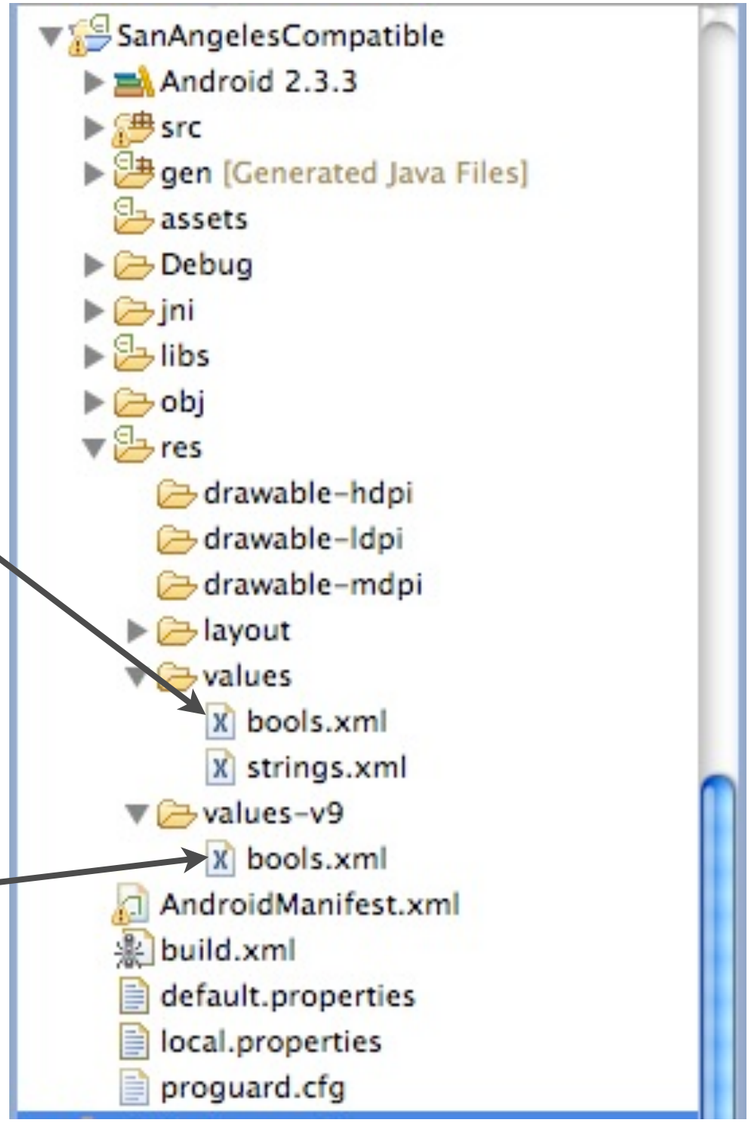
Platform-specific resources

– can be used to define boolean values

- activities can be enabled based upon these values

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <bool name="atLeastGingerbread">false</bool>
    <bool name="notGingerbread">true</bool>
</resources>
```

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <bool name="atLeastGingerbread">true</bool>
    <bool name="notGingerbread">false</bool>
</resources>
```



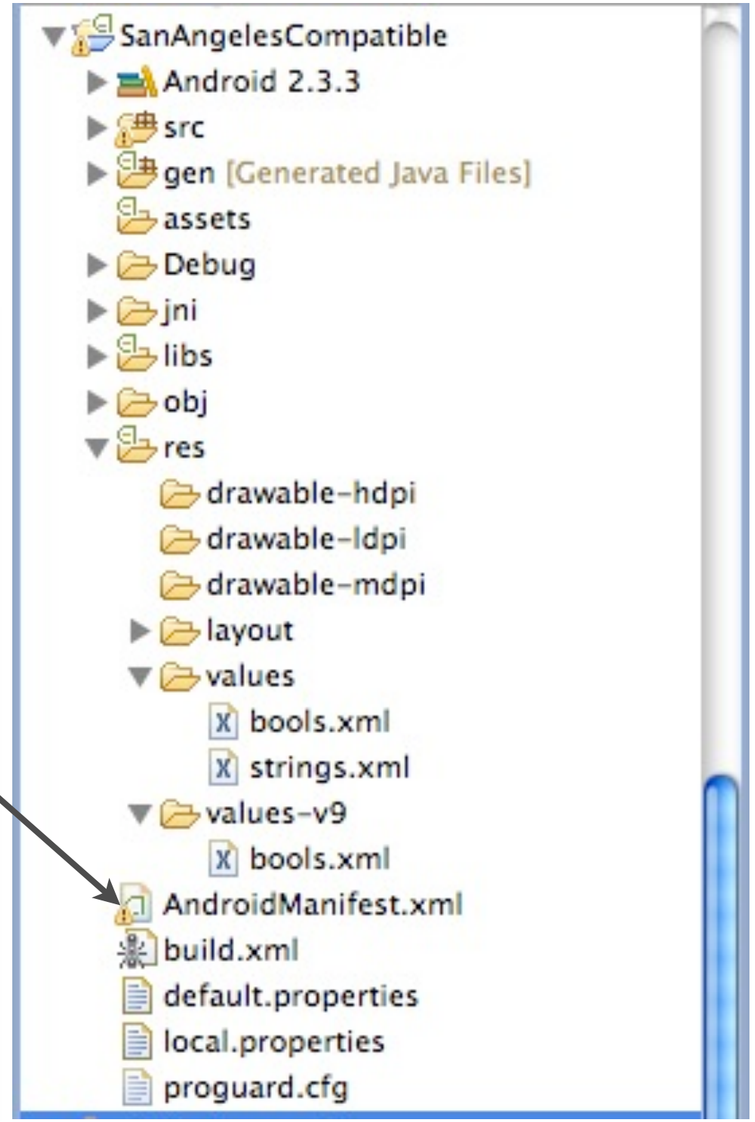
Using NativeActivity Compatibly

Platform-specific resources

– can be used to define boolean values

- activities can be enabled based upon these values

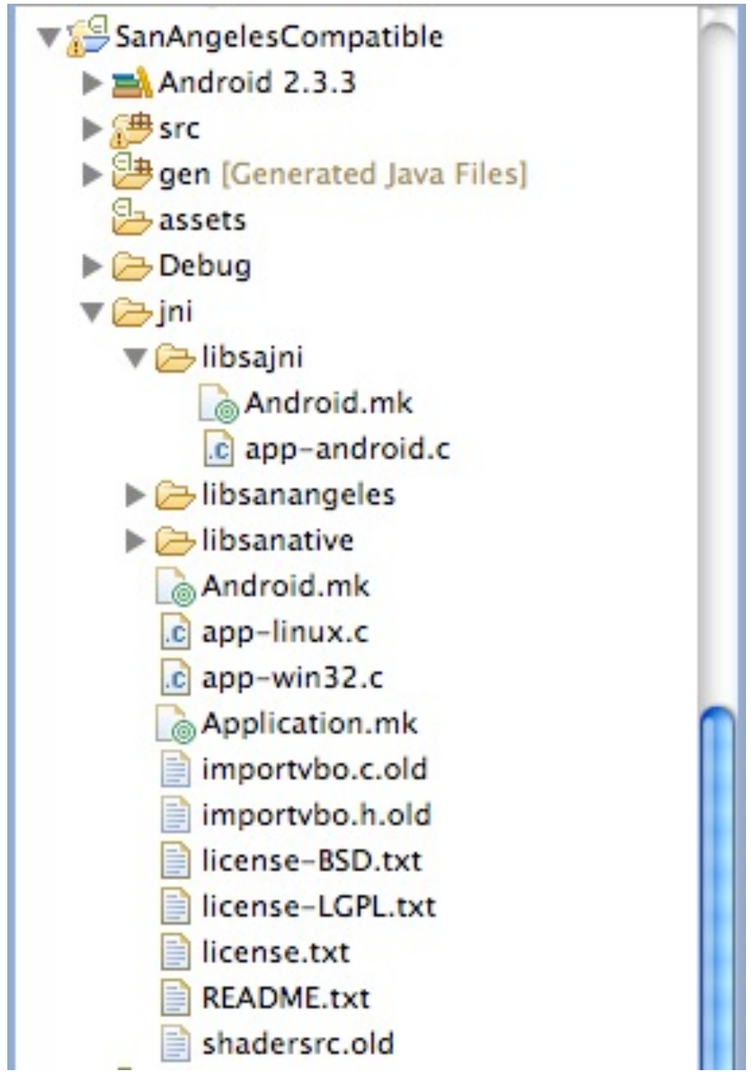
```
<activity android:name=".SANativeActivity"
   android:enabled="@bool/atLeastGingerbread">
  <meta-data android:name="android.app.lib_name"
     android:value="sanative" />
  <intent-filter>
    <action android:name="android.intent.action.MAIN"/>
    <category android:name="android.intent.category.LAUNCHER"/>
  </intent-filter>
</activity>
<activity android:name=".DemoActivity"
   android:enabled="@bool/notGingerbread">
  <intent-filter>
    <action android:name="android.intent.action.MAIN"/>
    <category android:name="android.intent.category.LAUNCHER"/>
  </intent-filter>
</activity>
```



Using NativeActivity Compatibly

Dynamic libraries

– create stub libraries as interfaces to your game engine

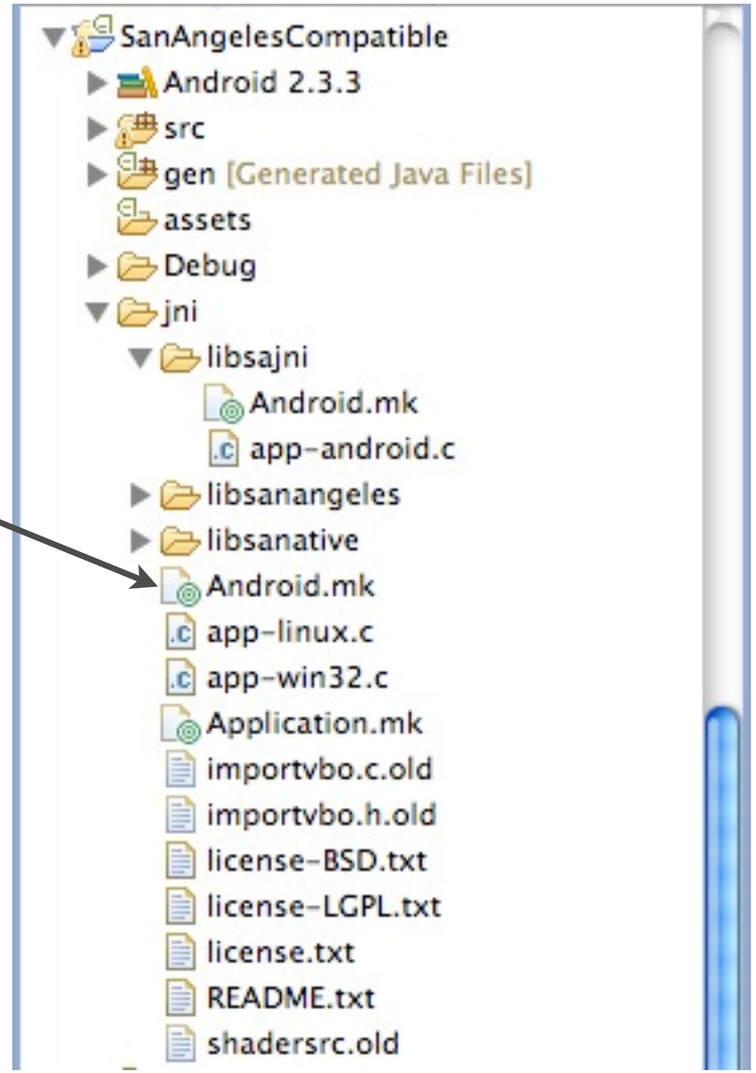


Using NativeActivity Compatibly

Dynamic libraries

– create stub libraries as interfaces to your game engine

```
JNI_ROOT_FOLDER   := $(call my-dir)
include $(JNI_ROOT_FOLDER)/libsanangeles/Android.mk
include $(JNI_ROOT_FOLDER)/libsanative/Android.mk
include $(JNI_ROOT_FOLDER)/libsajni/Android.mk
```



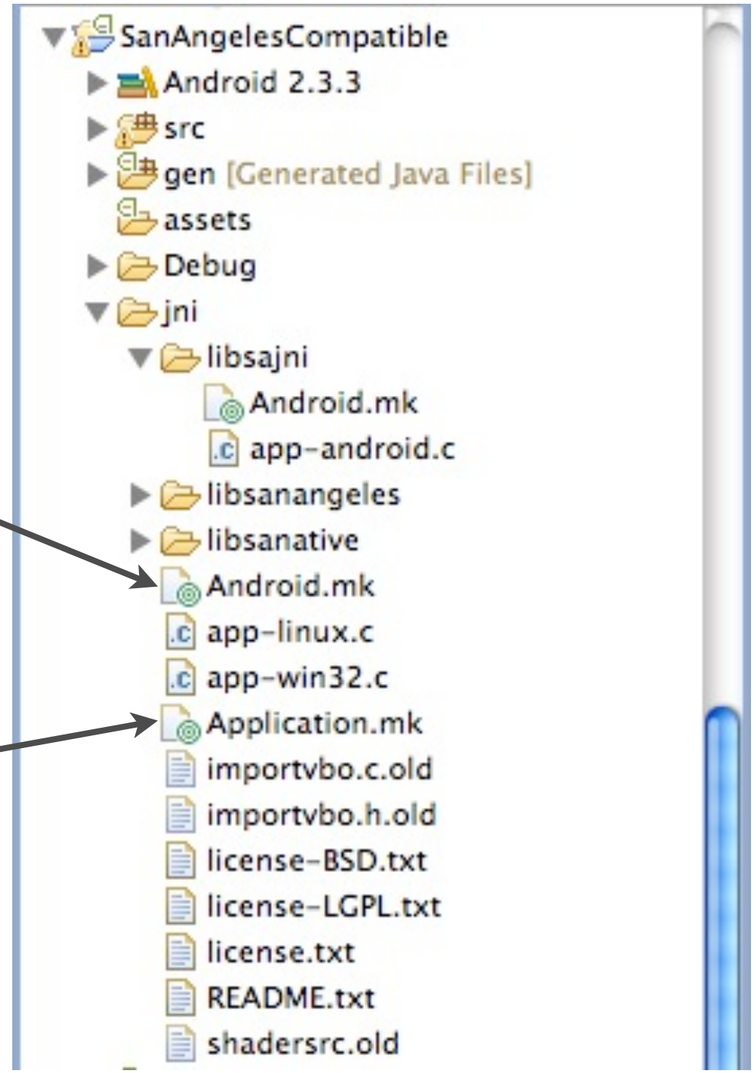
Using NativeActivity Compatibly

Dynamic libraries

– create stub libraries as interfaces to your game engine

```
JNI_ROOT_FOLDER   := $(call my-dir)
include $(JNI_ROOT_FOLDER)/libsanangeles/Android.mk
include $(JNI_ROOT_FOLDER)/libsanative/Android.mk
include $(JNI_ROOT_FOLDER)/libsajni/Android.mk
```

```
APP_PLATFORM := android-9
APP_MODULES := libsanangeles libsanative libsajni
APP_OPTIM := release
APP_ABI := armeabi armeabi-v7a
```



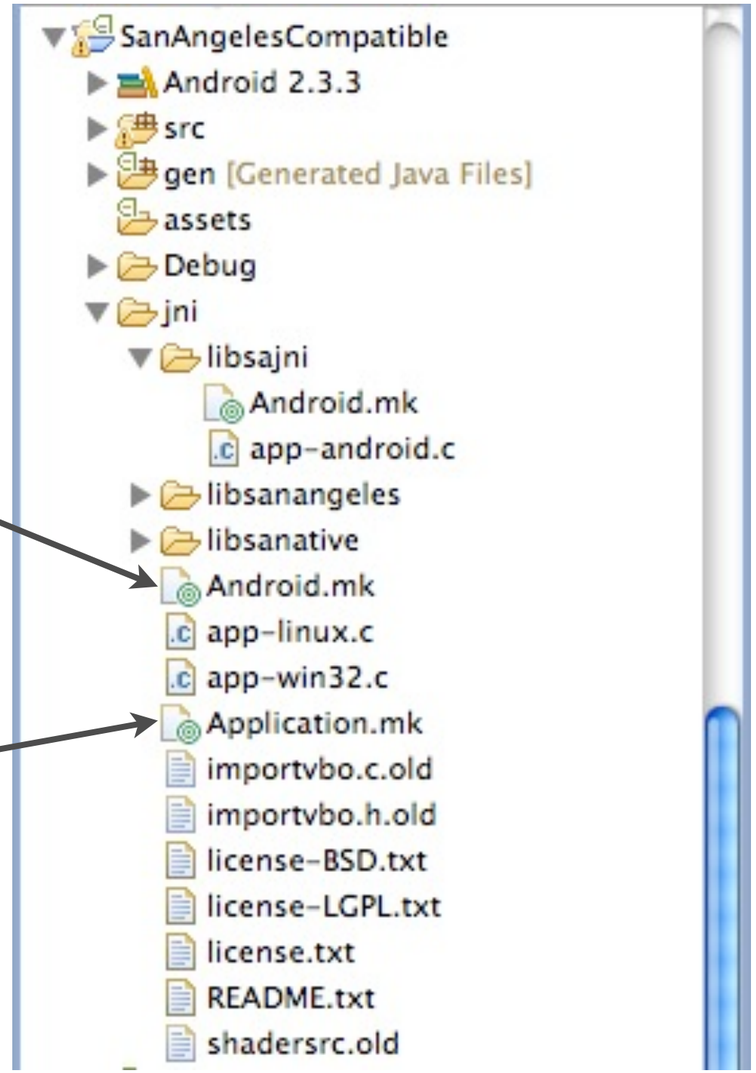
Using NativeActivity Compatibly

Dynamic libraries

– create stub libraries as interfaces to your game engine

```
JNI_ROOT_FOLDER   := $(call my-dir)
include $(JNI_ROOT_FOLDER)/libsanangeles/Android.mk
include $(JNI_ROOT_FOLDER)/libsanative/Android.mk
include $(JNI_ROOT_FOLDER)/libsajni/Android.mk
```

```
APP_PLATFORM := android-9
APP_MODULES := libsanangeles libsanative libsajni
APP_OPTIM := release
APP_ABI := armeabi armeabi-v7a
```



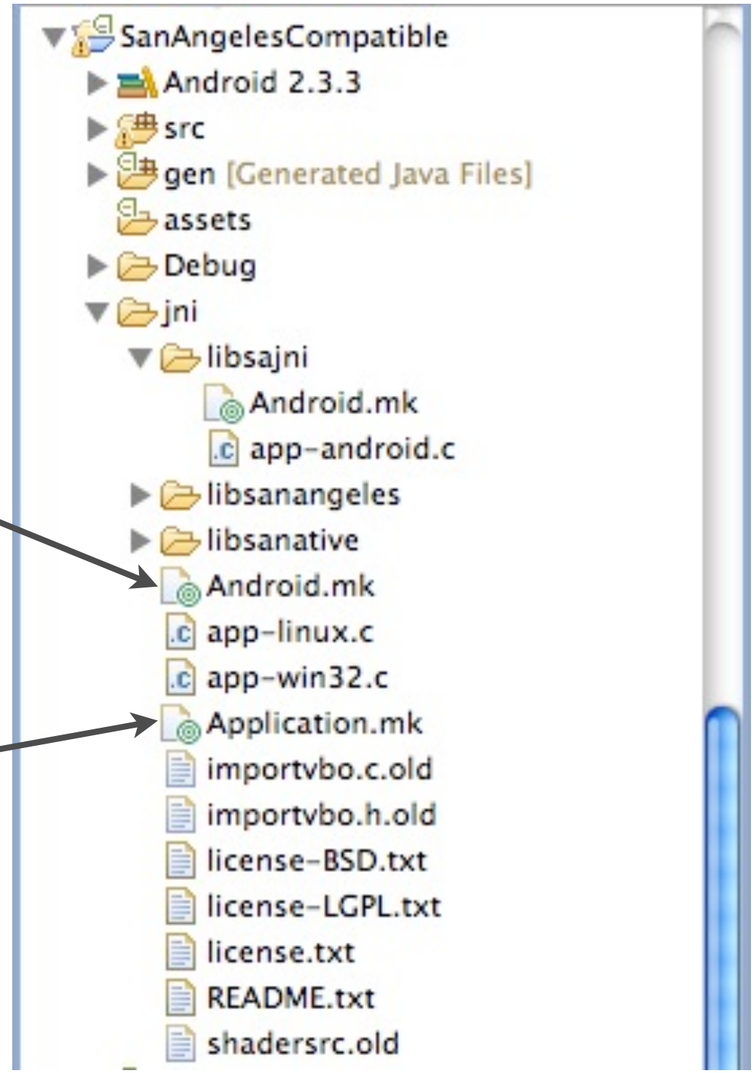
Using NativeActivity Compatibly

Dynamic libraries

– create stub libraries as interfaces to your game engine

```
JNI_ROOT_FOLDER   := $(call my-dir)
include $(JNI_ROOT_FOLDER)/libsanangeles/Android.mk
include $(JNI_ROOT_FOLDER)/libsanative/Android.mk
include $(JNI_ROOT_FOLDER)/libsajni/Android.mk
```

```
APP_PLATFORM := android-9
APP_MODULES := libsanangeles libsanative libsajni
APP_OPTIM := release
APP_ABI := armeabi armeabi-v7a
```



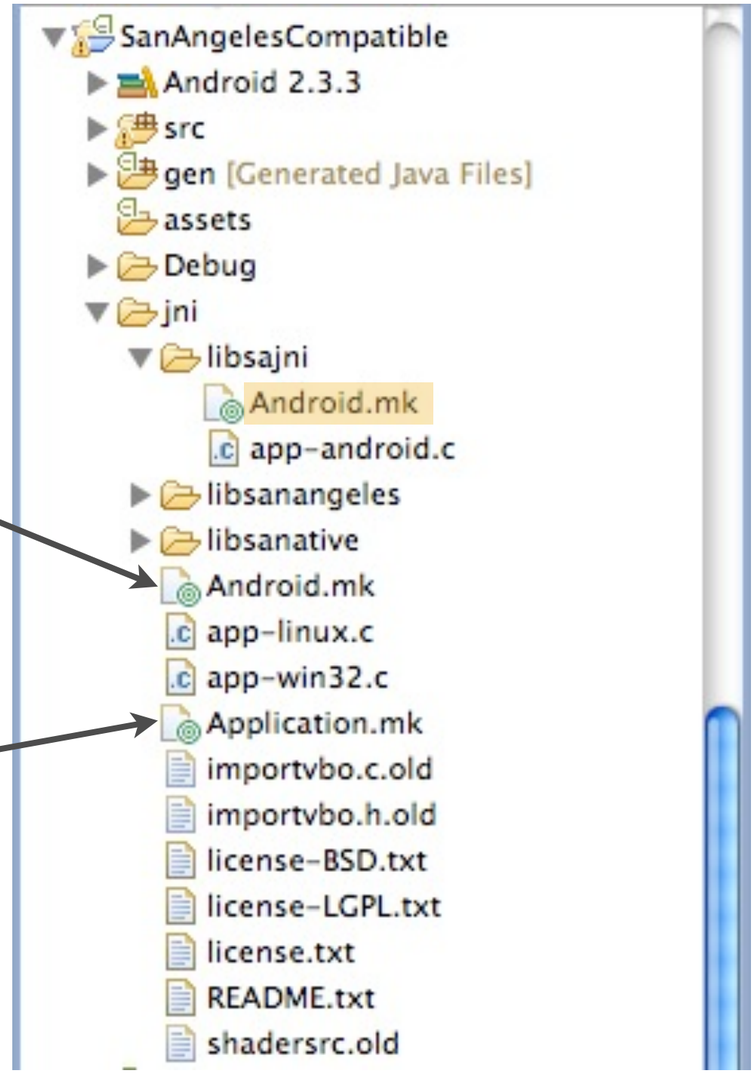
Using NativeActivity Compatibly

Dynamic libraries

– create stub libraries as interfaces to your game engine

```
JNI_ROOT_FOLDER   := $(call my-dir)
include $(JNI_ROOT_FOLDER)/libsanangeles/Android.mk
include $(JNI_ROOT_FOLDER)/libsanative/Android.mk
include $(JNI_ROOT_FOLDER)/libsajni/Android.mk
```

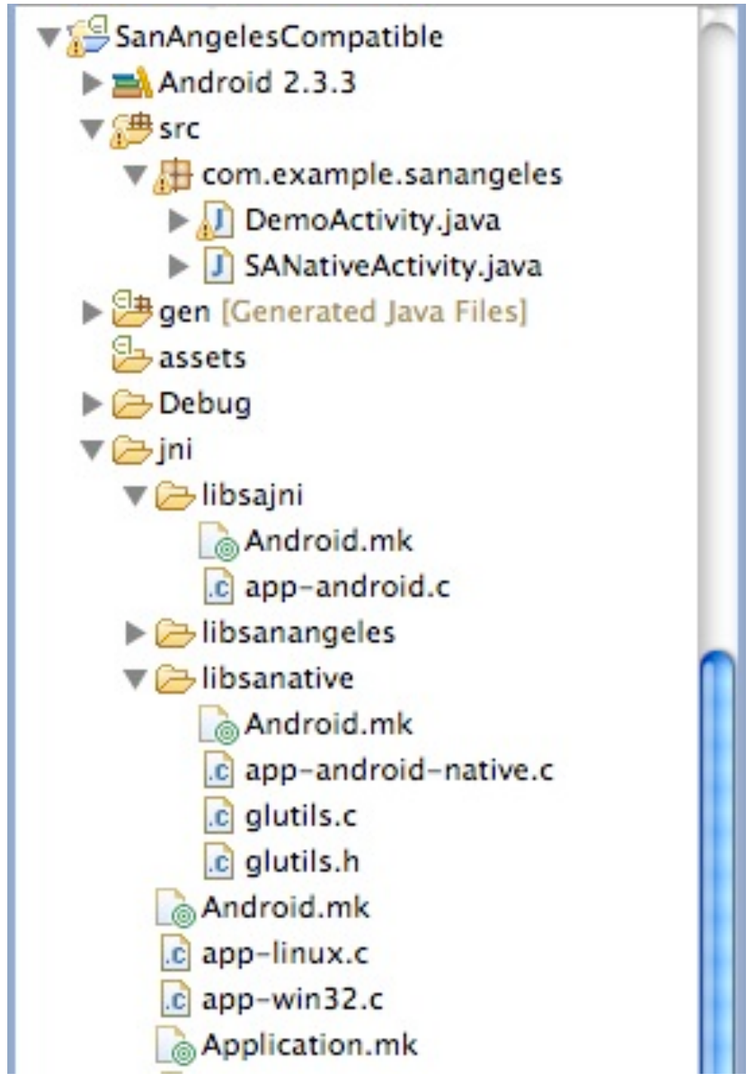
```
APP_PLATFORM := android-9
APP_MODULES := libsanangeles libsanative libsajni
APP_OPTIM := release
APP_ABI := armeabi armeabi-v7a
```



Using NativeActivity Compatibly

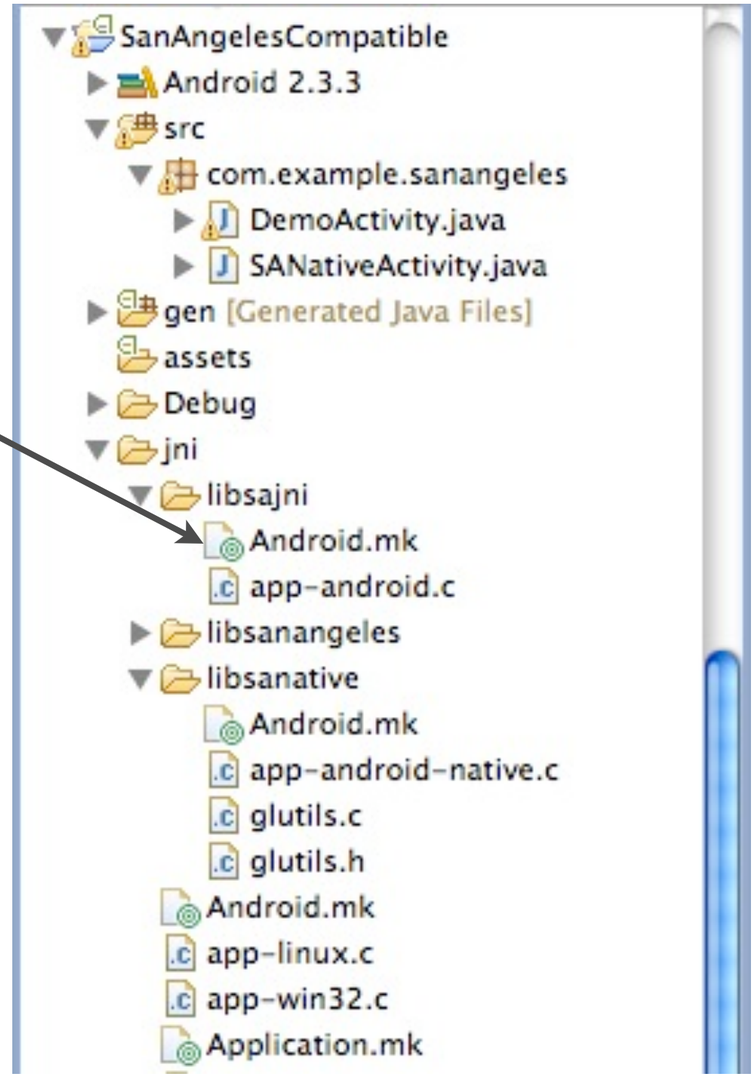
Dynamic Libraries

– create multiple libraries as interfaces to your game engine



Using NativeActivity Compatibly

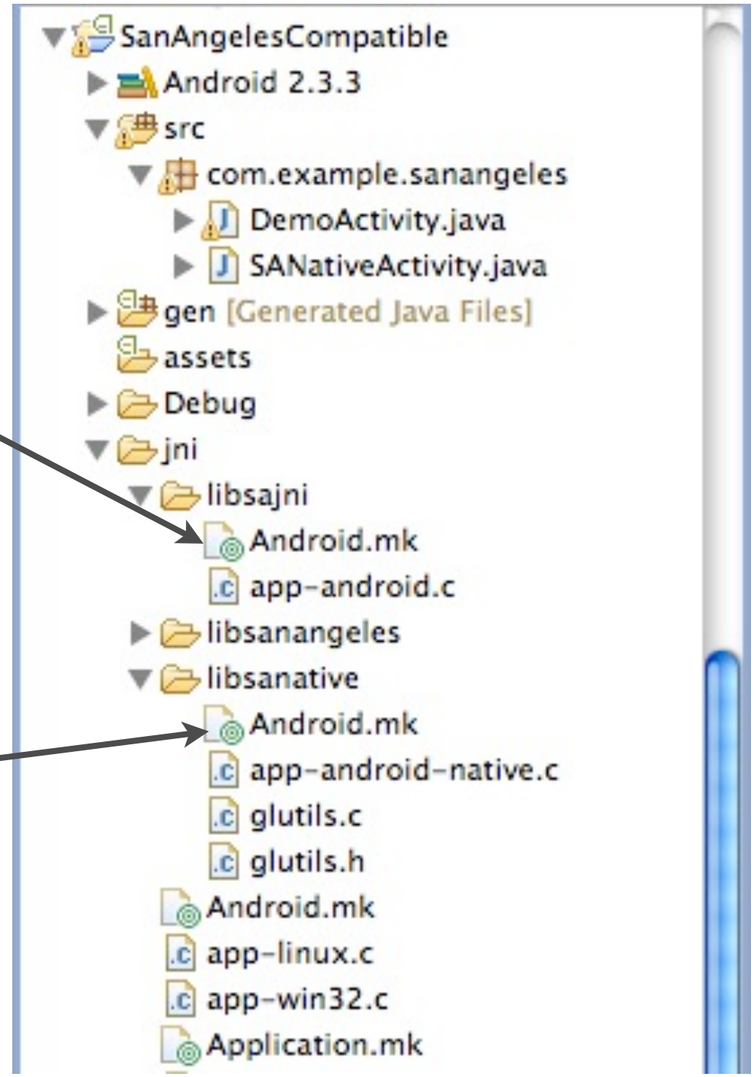
```
LOCAL_PATH   := $(call my-dir)
include $(CLEAR_VARS)
LOCAL_MODULE := sajni
LOCAL_CFLAGS := -DANDROID_NDK
LOCAL_SRC_FILES   := app-android.c
LOCAL_LDLIBS := -lGLESv2 -ldl -llog
LOCAL_SHARED_LIBRARIES    := sanangeles
include $(BUILD_SHARED_LIBRARY)
```



Using NativeActivity Compatibly

```
LOCAL_PATH   := $(call my-dir)
include $(CLEAR_VARS)
LOCAL_MODULE := sajni
LOCAL_CFLAGS := -DANDROID_NDK
LOCAL_SRC_FILES   := app-android.c
LOCAL_LDLIBS := -lGLESv2 -ldl -llog
LOCAL_SHARED_LIBRARIES    := sanangeles
include $(BUILD_SHARED_LIBRARY)
```

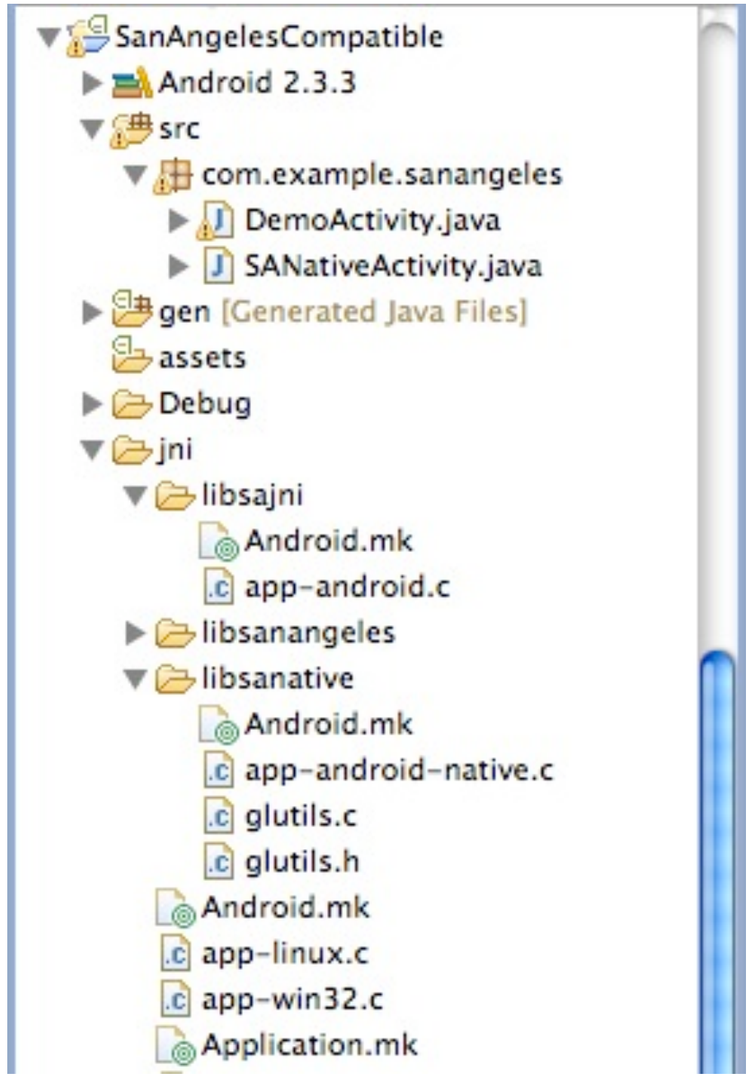
```
LOCAL_PATH   := $(call my-dir)
include $(CLEAR_VARS)
LOCAL_MODULE := sanative
LOCAL_CFLAGS := -DANDROID_NDK
LOCAL_SRC_FILES   := glutils.c \
    app-android-native.c
LOCAL_LDLIBS := -lGLESv2 -ldl -llog -lEGL -landroid
LOCAL_STATIC_LIBRARIES    := android_native_app_glue
LOCAL_SHARED_LIBRARIES    := sanangeles
include $(BUILD_SHARED_LIBRARY)
$(call import-module,android/native_app_glue)
```



Using NativeActivity Compatibly

Dynamic libraries

- the third library must be loaded manually before the native activity library
- we can extend NativeActivity to do this

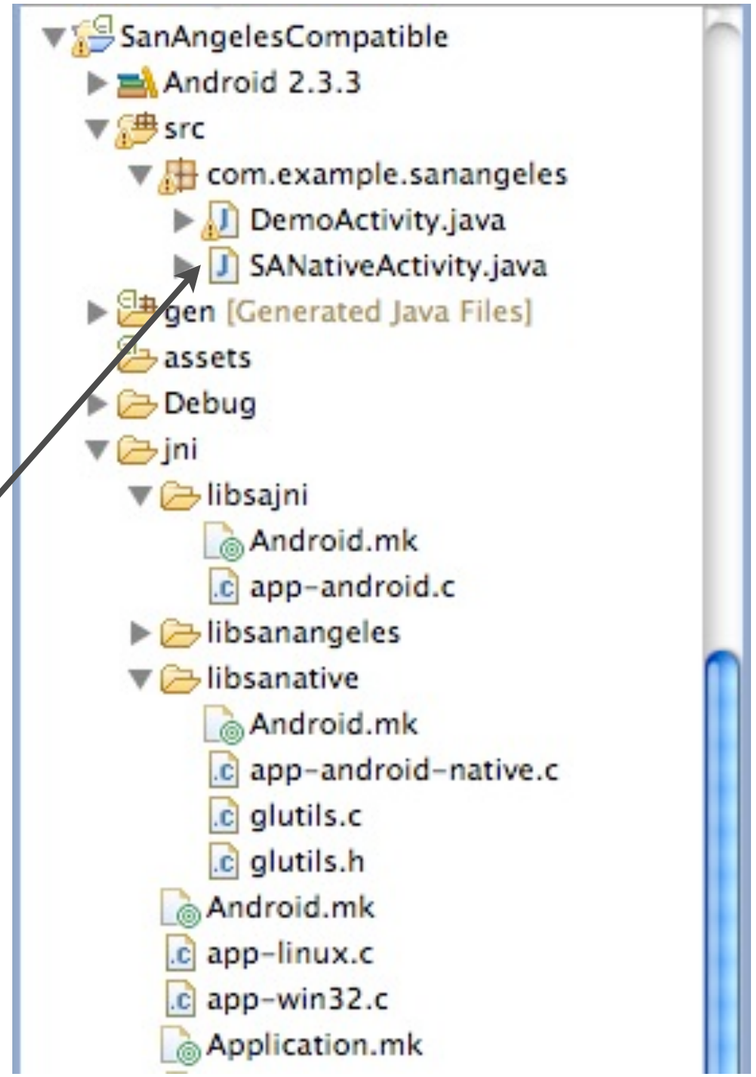


Using NativeActivity Compatibly

Dynamic libraries

- the third library must be loaded manually before the native activity library
- we can extend NativeActivity to do this

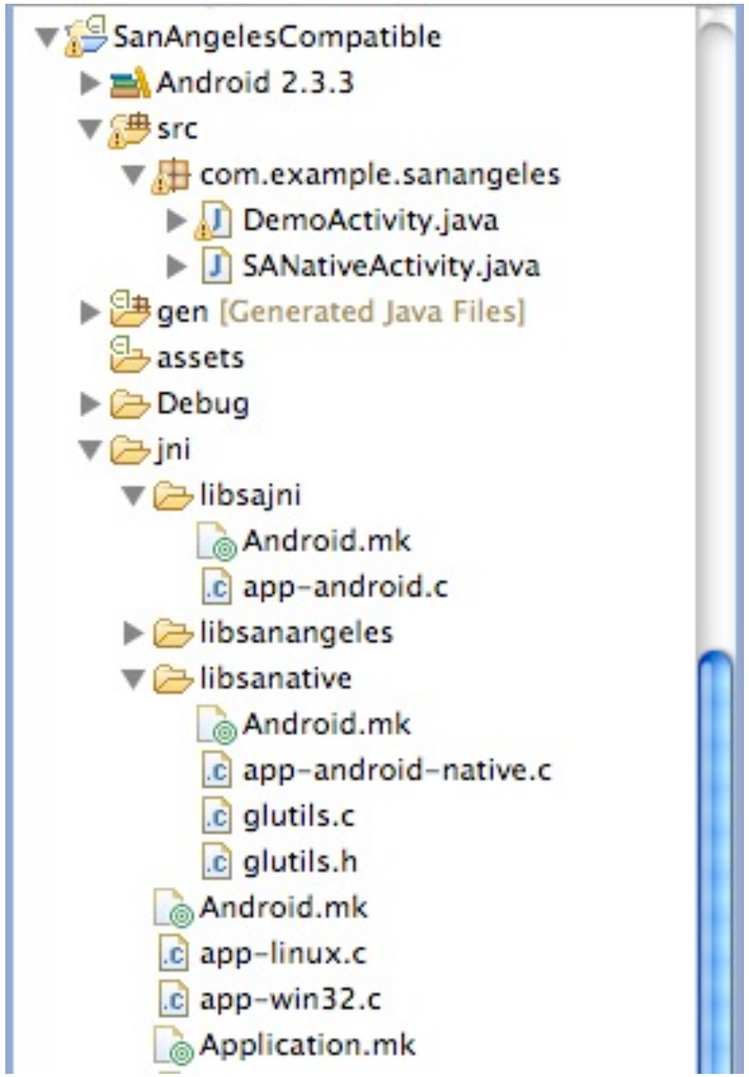
```
package com.example.sanangeles;

public class SANativeActivity extends
android.app.NativeActivity {

    static {
        System.loadLibrary("sanangeles");
    }
}
```



Using NativeActivity Compatibly

Dynamic libraries

- the third library must be loaded manually before the native activity library
- we can extend NativeActivity to do this
- with our JNI-based class, we load in order

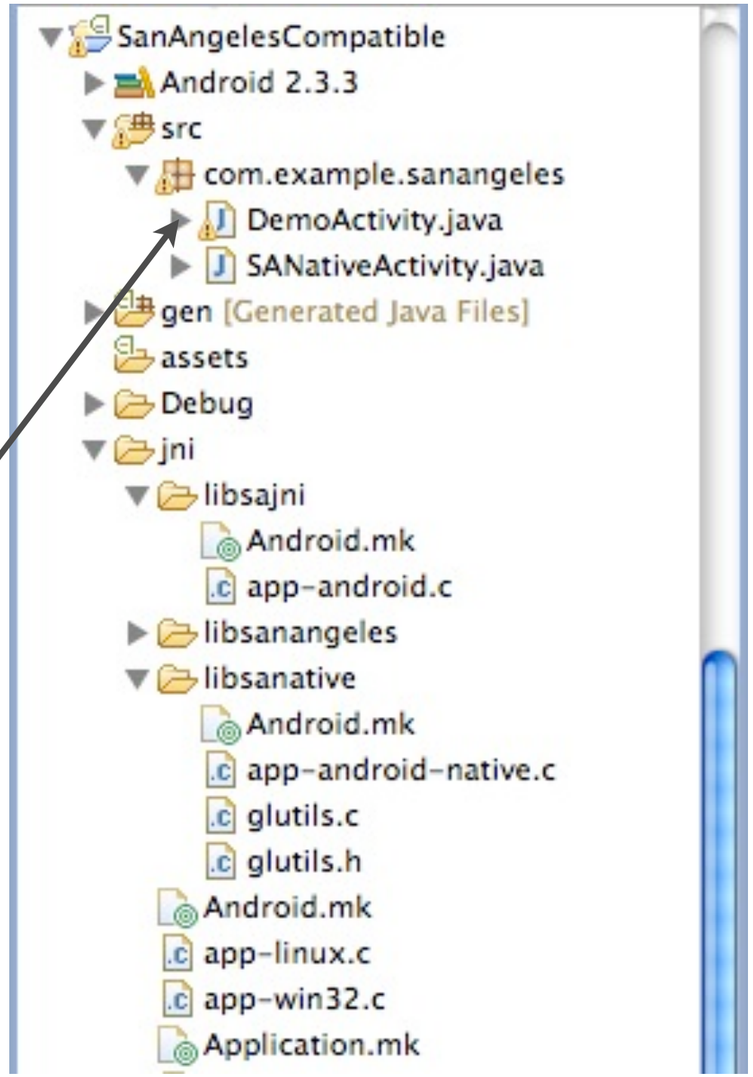


Using NativeActivity Compatibly

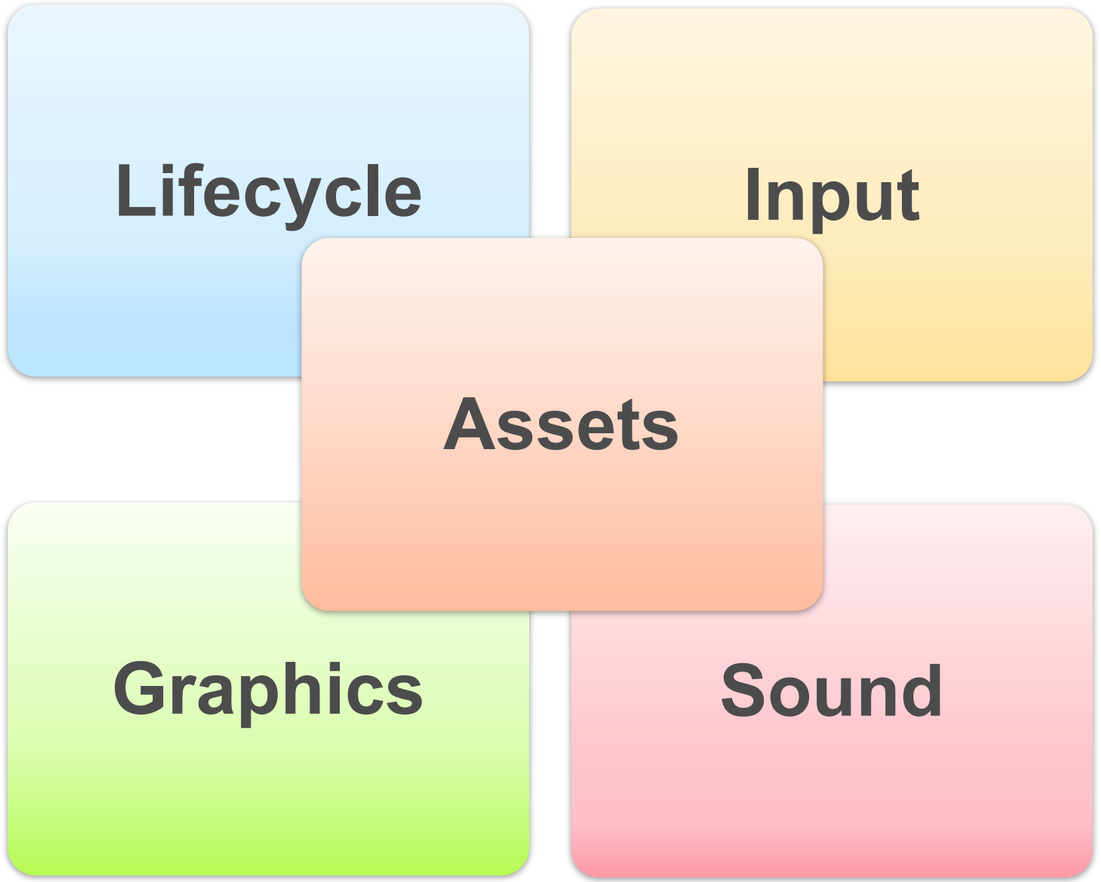
Dynamic libraries

- the third library must be loaded manually before the native activity library
- we can extend NativeActivity to do this
- with our JNI-based class, we load in order

```
package com.example.sanangeles;

public class DemoActivity extends
android.app.Activity {
    static {
        System.loadLibrary("sanangeles");
        System.loadLibrary("sajni");
    }
...
```

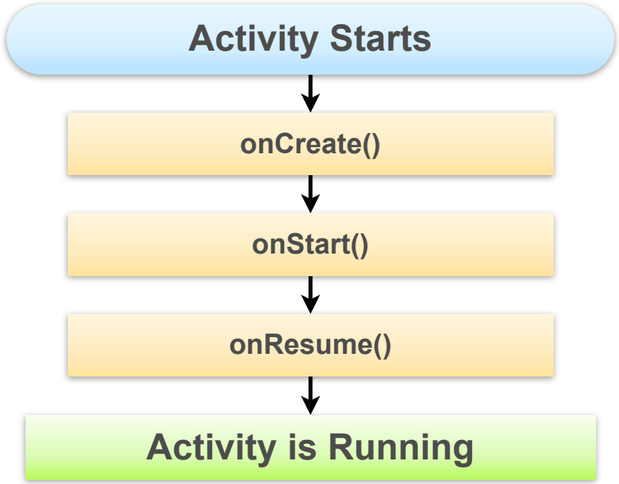


Key Game Components

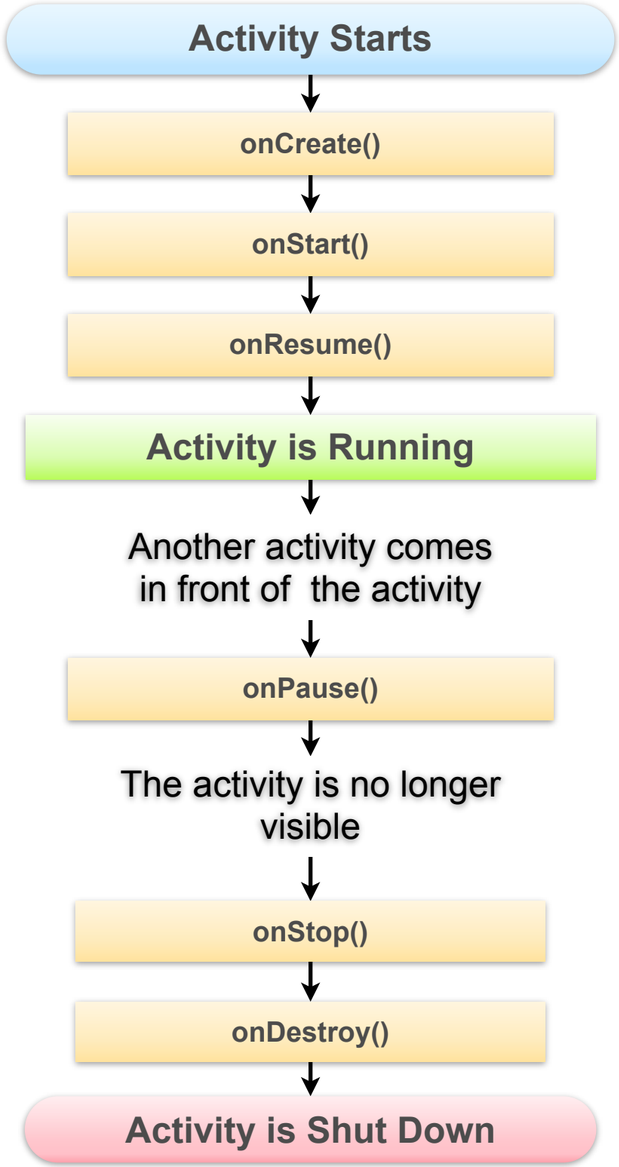


Lifecycle Activity Lifecycle

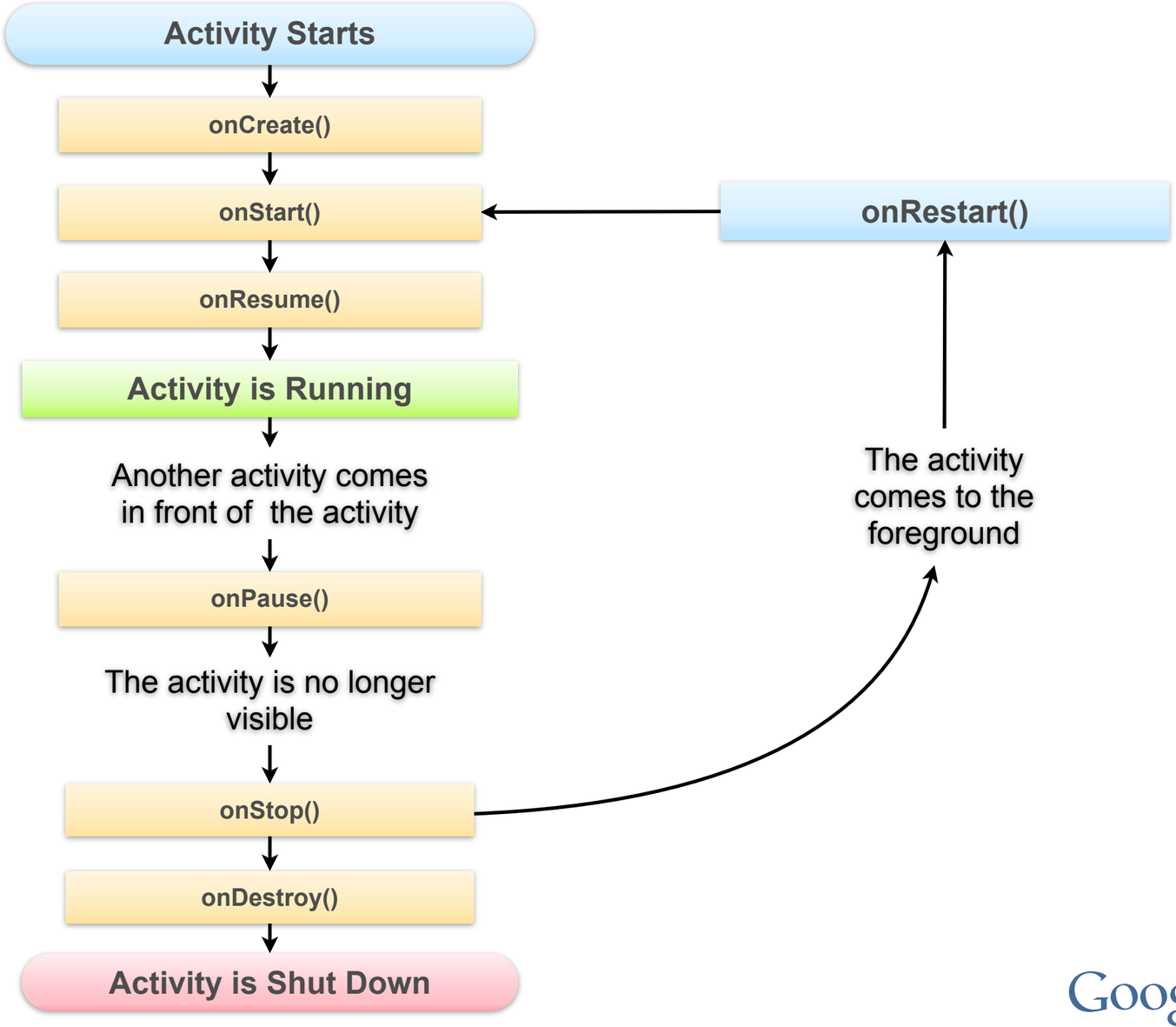
Lifecycle Activity Lifecycle



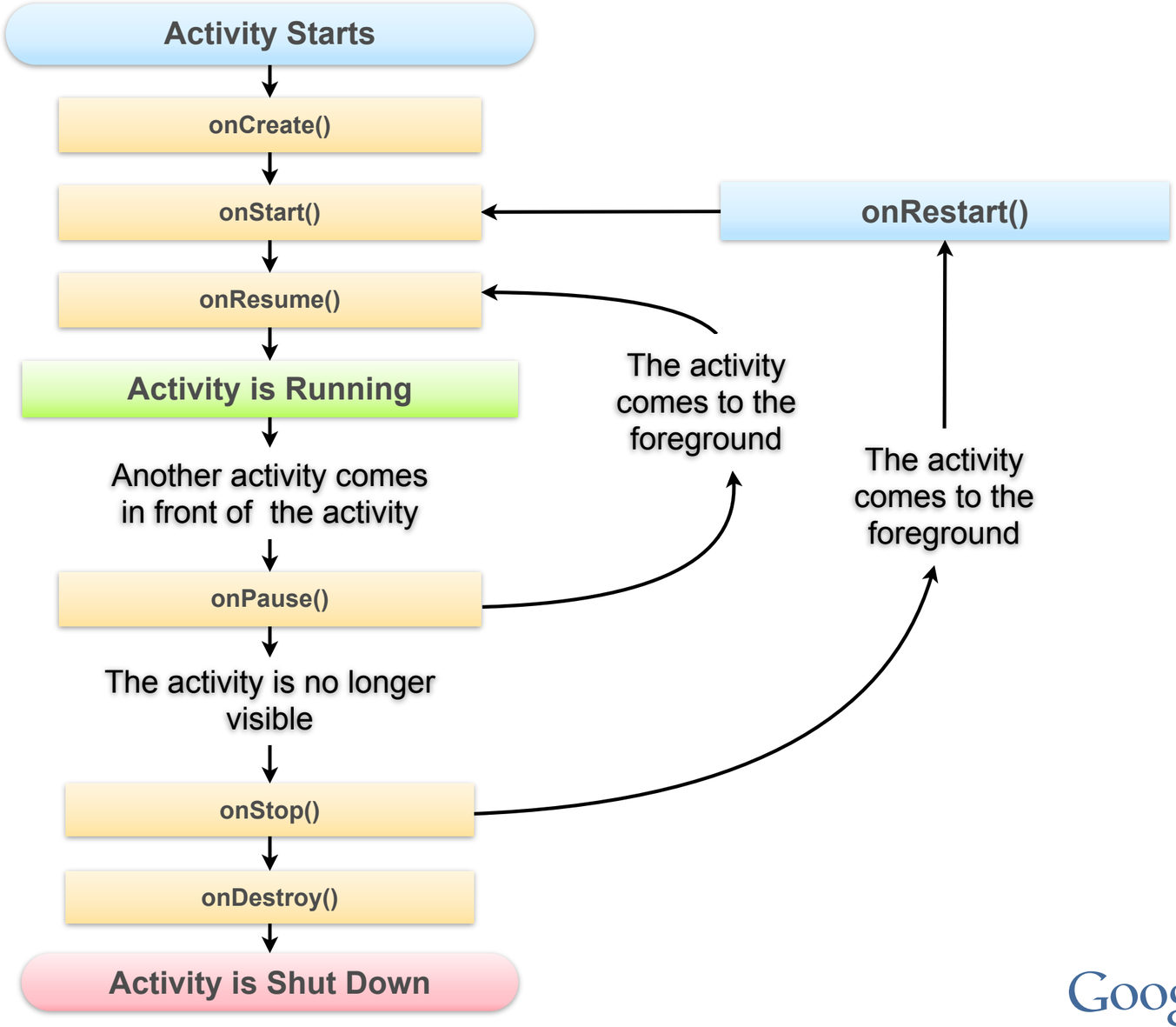
Lifecycle Activity Lifecycle



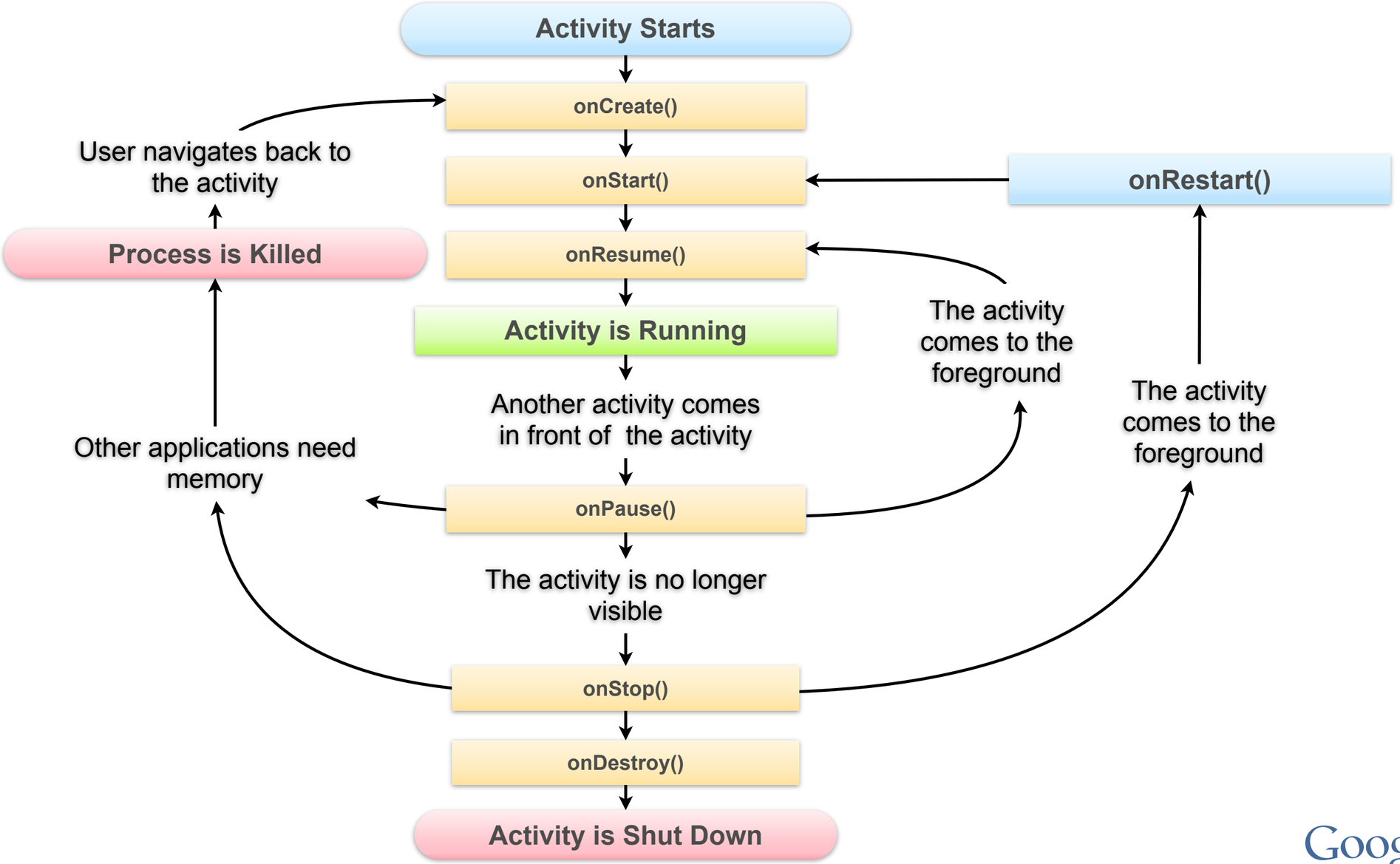
Lifecycle Activity Lifecycle



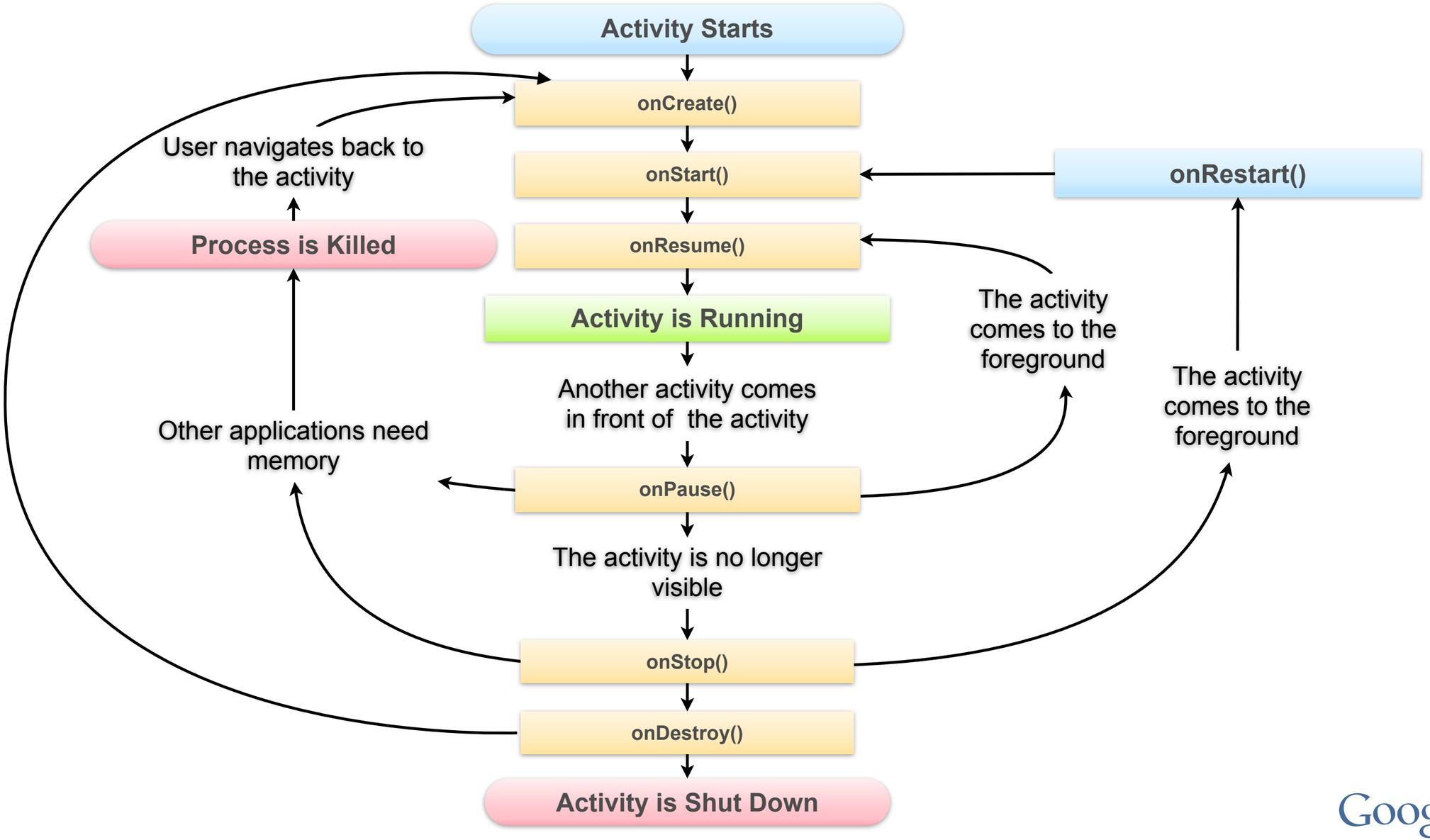
Lifecycle Activity Lifecycle



Lifecycle Activity Lifecycle



Lifecycle Activity Lifecycle



Lifecycle Process Lifecycle

Android keeps application processes around for as long as possible

– Even when the activity is closed with `finish()` or the back key is pressed

The foreground activity process is considered the most important

– It can still be killed as a last resort

Visible activities are also given high priority

– Backgrounded activities are considered less important

– Processes with no active activities are killed first

Lifecycle Implications of the Process Lifecycle

Be careful with static initializers

- Your library cannot be unloaded while your process is active
  - Don’t assume that your static initializers will be run when your activity is created

Be careful with threads

- Threads are tied to the process, not the activity
  - Your threads will continue to run at reduced priority even if your activity is backgrounded or destroyed
- Gameplay should stop during onPause
  - It might make sense to keep certain threads running - your activity may still be visible
- Make sure all native code is cleaned up during onDestroy
  - Users expect that your application will no longer be active when its activity is destroyed

Input The Basics

Key handling

- Your device most likely has capacitive buttons
  - Look for complete state transitions of `onKeyDown`/`onKeyUp`
  - Avoid overriding `dispatchKeyEvent`
- Use `onKeyMultiple`/`onKeyLongPress`
  - Keep your game behaving like an Android application

Touch/Multitouch handling

- With `onTouchEvent`
  - take advantage of built-in gesture processing
    - use `ScaleGestureDetector` and `GestureDetector`
- Most devices now support multitouch
  - determine if your device can track distinct points
    - `hasSystemFeature(FEATURE_TOUCHSCREEN_MULTITOUCH_DISTINCT)`

Input Other Input

Trackball handling

- With `onTrackballEvent`
  - Unhandled events turn into dpad events

Motion handling

- With `onSensorChanged`
  - Must be registered with the `SensorManager`
  - Disable unneeded sensors, especially when paused
    - Active sensors will quickly drain the battery

Input Androidify Your Game!

Don’t break the “hard buttons”

- Back: dismiss dialogs, exit menus, pause the game
- Menu: pause, launch options dialog
- Home: pause (**not** quit!)

Don’t break the volume buttons

- Don’t eat volume button messages
- `setVolumeControlStream(AudioManager.STREAM_MUSIC)`

Don’t let the “hard buttons” break your game

- Very easy to press accidentally on some devices
  - Pause and give user the option to exit when back is pressed
  - Save game state whenever possible onPause/onStop

Input Androidify Your Game!

Don’t break the “hard buttons”

- Back: dismiss dialogs, exit menus, pause the game
- Menu: pause, launch options dialog
- Home: pause (**not** quit!)

Don’t break the volume buttons

- Don’t eat volume button messages
- `setVolumeControlStream(AudioManager.STREAM_MUSIC)`

Don’t let the “hard buttons” break your game

- Very easy to press accidentally on some devices
  - Pause and give user the option to exit when back is pressed
  - Save game state whenever possible onPause/onStop

Graphics OpenGL

GLSurfaceView

- Creates the compatible context
  - Contexts are shared between native code and Dalvik
- Creates a rendering thread
- Releases the context during onPause
- Not the most efficient way to multitask with OpenGL, but it avoids some driver issues

When context is lost

- Reload resources
  - Textures
  - Buffer objects
  - Shaders
- Names don’t persist

Graphics Texture Formats

Android encourages innovation in hardware

– Chipset vendors use several kinds of proprietary texture compression formats

– OpenGL ES supports multiple compressed formats, none mandatory

Solutions

– Uncompressed Textures

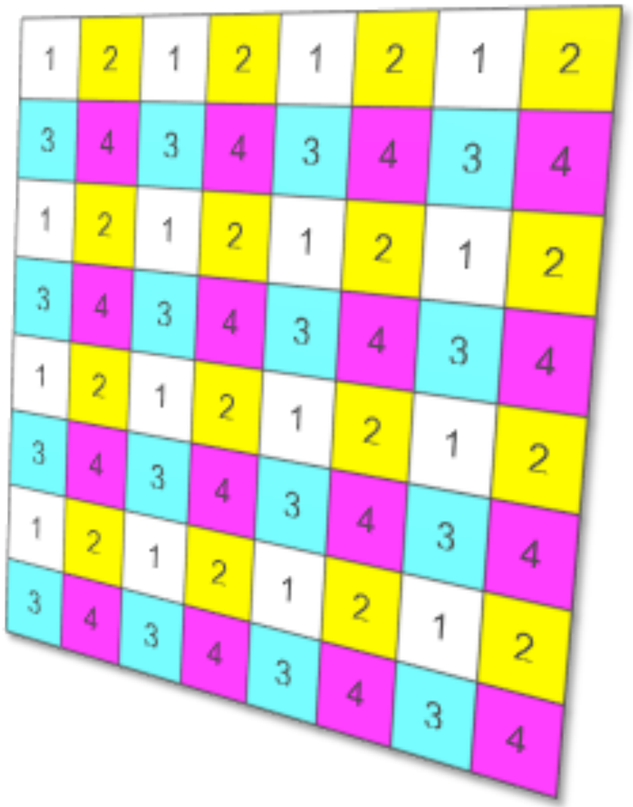
– Use ETC1 – broad support in 2.0

- Optimized around photographic data
- No Alpha

– Use Multi-APK (coming soon)

– Put textures on an asset server

- Detect support at runtime
- Download to each device



Graphics Drawing for Performance

Draw order is important

– Has no impact on some platforms, means everything on others

– Draw front to back to make best use of early rejects

Use VBOs

– Avoid unnecessary state changes

Textures

– Use power of 2 sized textures

- It can be very slow
- On some chipsets not all address modes work!

Graphics Writing Compatible Shaders

Query, query

- Make sure to query for number of attributes, varying, and uniforms supported
  - Varies between devices
  - Not respecting will make your application crash!!
  - “Supported” does not mean “performs well”

Conditions

- Avoid as much as possible
  - if necessary, put them last in the shader
- Use mix(), clamp(), etc to create the same result

discard()

- Avoid discard()
  - if necessary it should be last

Sound Audio Solutions

Full track audio

- `MediaPlayer`
  - Plays multiple formats with codec support
  - Plays from filesystem or from network
  - Has high latency, memory, and CPU usage for quick sound effects

Sound effects and clips

- `SoundPool`
  - Plays multiple formats with codec support
  - Preloads audio files to a pool
    - Plays them with low latency

Sound Audio Solutions

Application-generated audio

- `AudioTrack`
  - Plays PCM audio only with minimal latency
  - Useful for applications that wish to perform sound processing or mixing

OpenSL/ES

- Combines capabilities of `MediaPlayer` and `AudioTrack`
- Available only in the NDK
- Requires Android 2.3 and higher

Assets The JNI Solution

Get the APK file name

- `context.getApplicationInfo().sourceDir`

Use the asset manager code to get an AssetFileDescriptor

- `afd = context.getAssets().openFd("ToothyDroid.png");`

Read the offset and length from the AssetFileDescriptor

```
- long offset = afd.getStartOffset();
  long length = afd.getDeclaredLength();
  afd.close();
```

Open the APK file at the specified offset and read the specified length

Assets The JNI Solution

Use a native Zip Utility

- Slightly less future-proof
- Potentially faster
  - Make sure to cache the Zip directory
  - Get the offset and length if the item isn't compressed

```
sourceDir
```

an AssetFileDescriptor

```
enFd("ToothyDroid.png");
```

AssetFileDescriptor

```
long offset = afd.getStartOffset();
long length = afd.getDeclaredLength();
afd.close();
```

Open the APK file at the specified offset and read the specified length

Assets The JNI Solution

Use a native Zip Utility

– Slightly less future-proof

– Potentially faster

- Make sure to cache the Zip directory
- Get the offset and length if the item isn't compressed

Use native AssetManager

– Gingerbread and Beyond

```
sourceDir

an AssetFileDescriptor
enFd("ToothyDroid.png");

ptor
```

```
– long offset
  long length
  afd.close();
```

Open the APK file at the specified offset and read the specified length

Troubleshooting, Tips, and Tricks

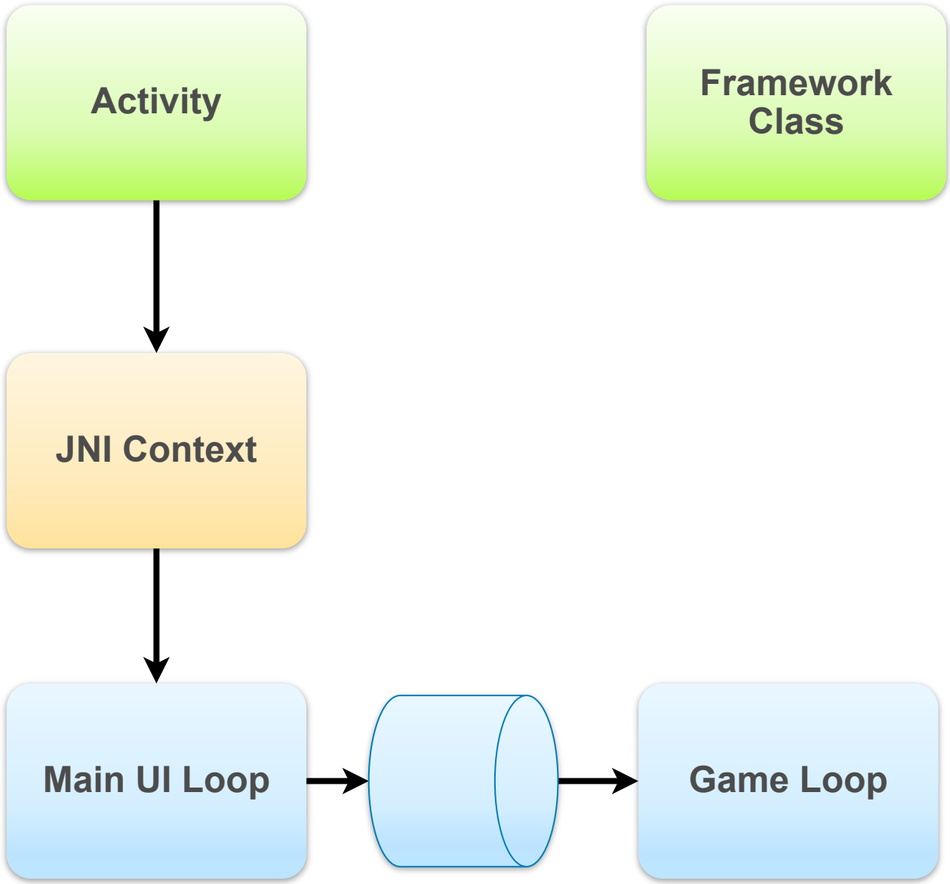
Targeting tricky, trying tasks

Troubleshooting: Threading, Dalvik, and the NDK

Problem: App crashes mysteriously

Solution: Check your threading

– JNI contexts are per-thread

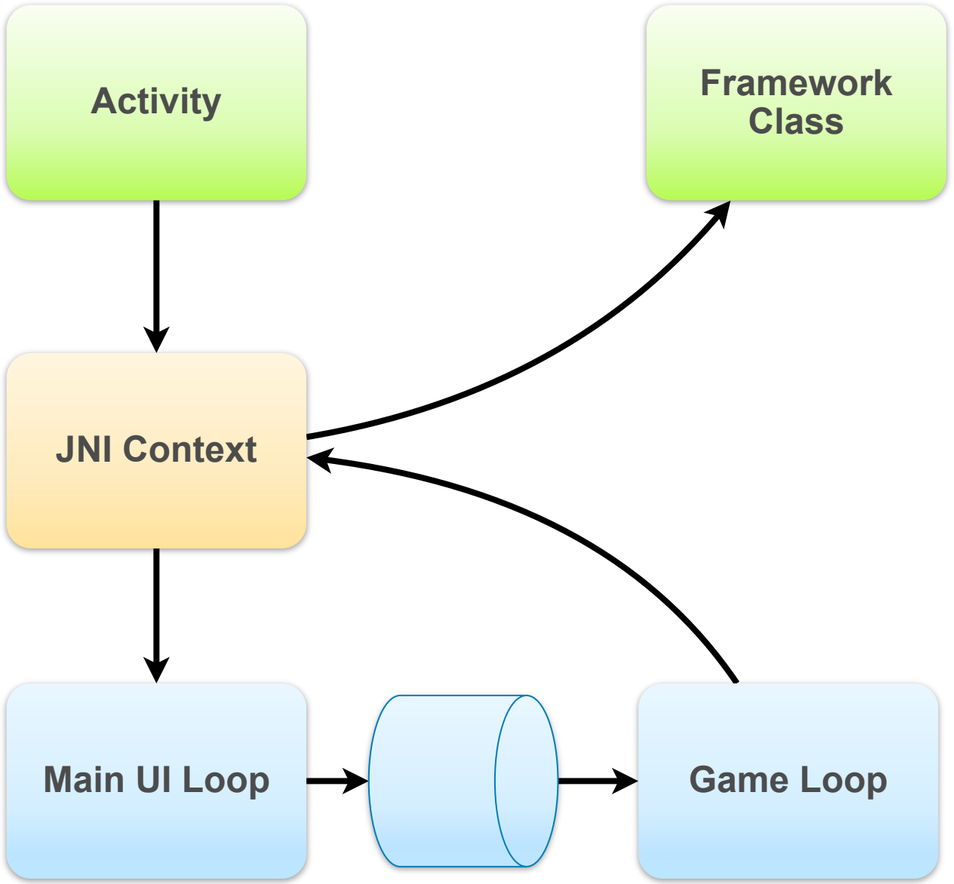


Troubleshooting: Threading, Dalvik, and the NDK

Problem: App crashes mysteriously

Solution: Check your threading

– JNI contexts are per-thread

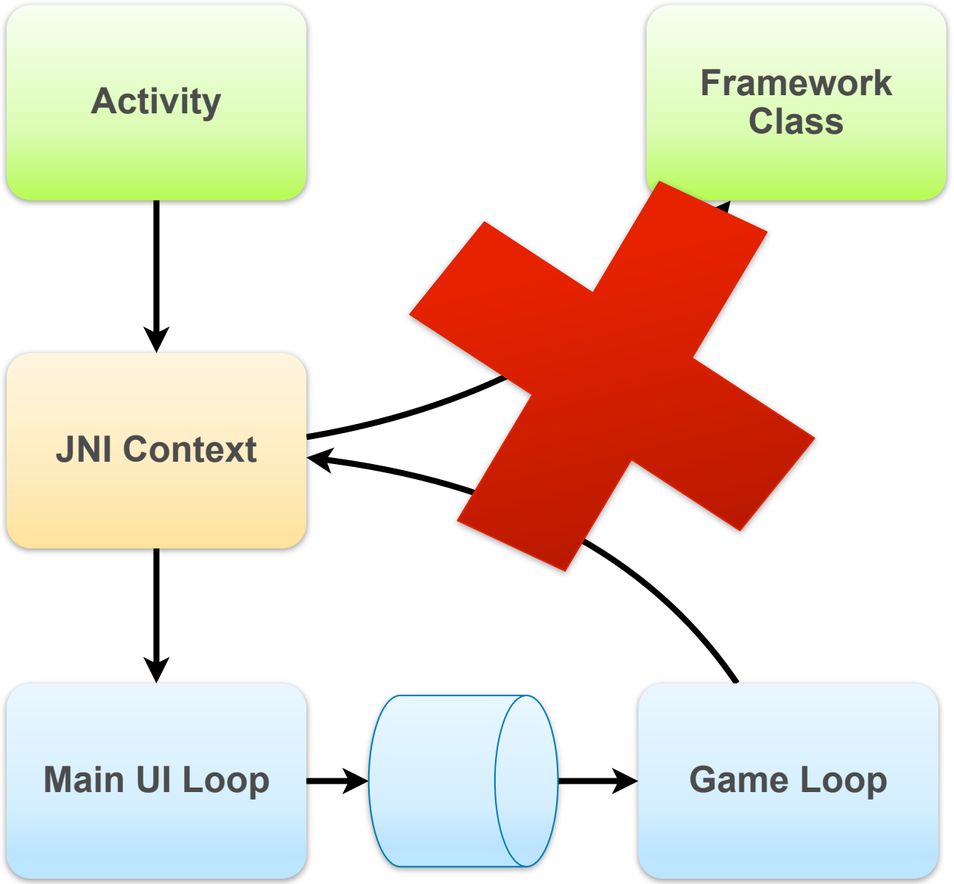


Troubleshooting: Threading, Dalvik, and the NDK

Problem: App crashes mysteriously

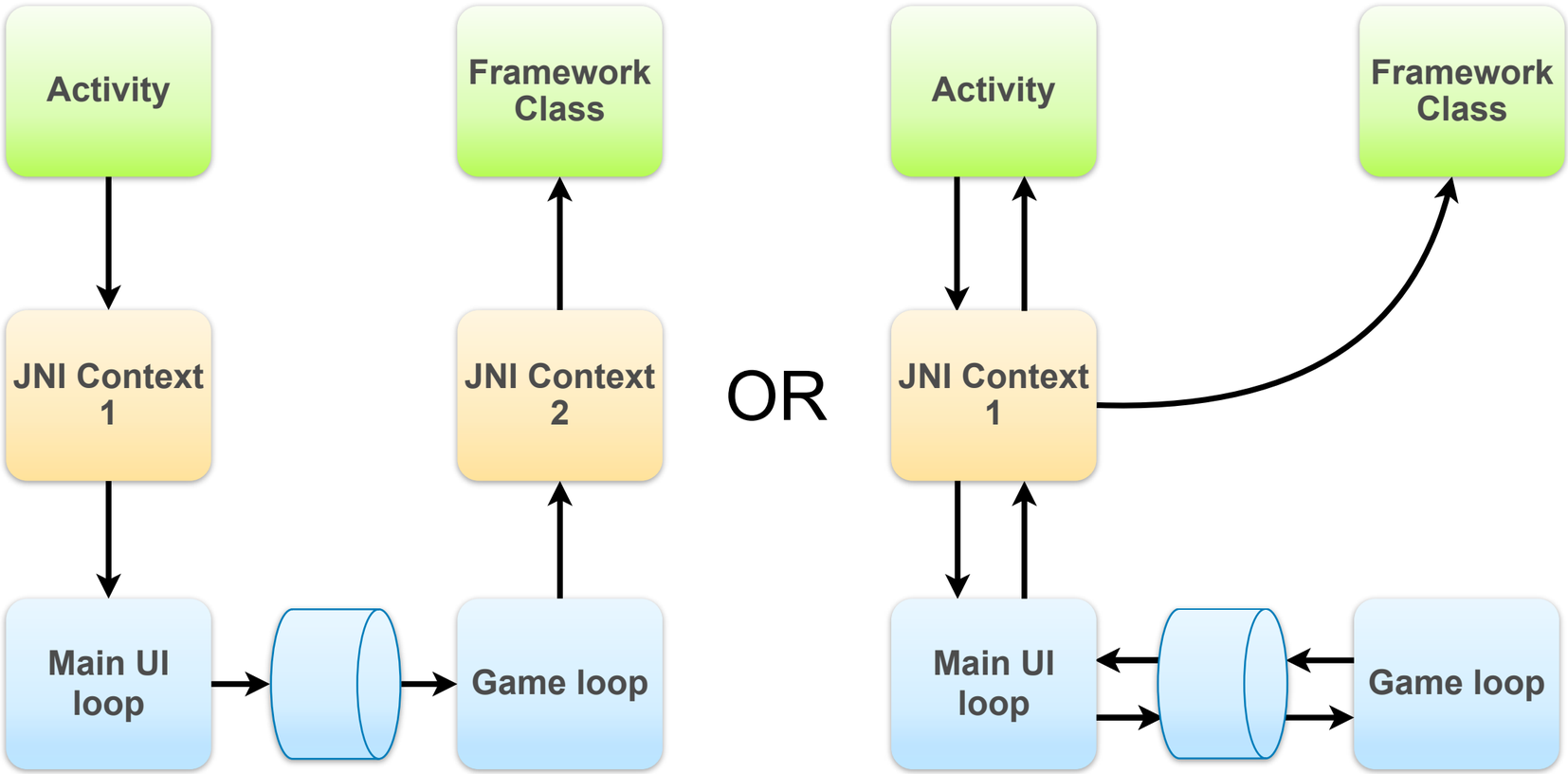
Solution: Check your threading

– JNI contexts are per-thread



Threading Options

One JNI Context Per Thread



Troubleshooting: Performance

Check for older hardware:

- ARM7 CPUs (use cpu-features lib)
- No support for OpenGL 2

Fallbacks for older HW:

- Use 16-bit textures
  - `BitmapFactory.Options.inPreferredConfig = Bitmap.Config.ARGB_4444`
- Use smaller render target, scale up
  - Framework will do it for you with `SurfaceHolder.setFixedSize()`
  - Pixels are expensive, especially on tablets

Troubleshooting: Performance

Check for older hardware:

– ARM7 CPUs (use cpu-features lib)

– No support for OpenGL 2

Fallbacks for older HW:

– Use 16-bit textures

- `BitmapFactory.Options.inPreferredConfig = Bitmap.Config.ARGB_4444`

– Use smaller render target, scale up

- Framework will do it for you with `SurfaceHolder.setFixedSize()`
- Pixels are expensive, especially on tablets

Troubleshooting: Performance

Check for older hardware:

– ARM7 CPUs (use cpu-features lib)

– No support for OpenGL 2

Fallbacks f

– Use 16-b

- `BitmapF`
  `Bitmap.`

– Use sma

- Framew
  `Surface`
- Pixels are expensive, especially on tablets

If all else fails: Market filters

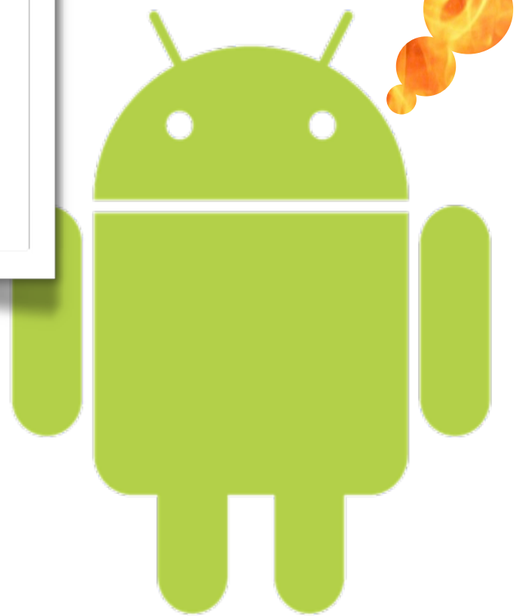
– Instruction set

  – ARMv5/ARMv7

– GL version

– Platform release

– Use Device Availability



Tips: Standing Out from the Crowd

Start Small

– Download assets as-needed

– Hook me early instead of making me wait

Use Android backup/restore service (save to the cloud)

– Keep my progress when I switch devices

Support multiple control schemes

– Trackball, D-pad, virtual stick…

– Let me decide what works best

Support install-to-SD!

– Android Market users will knock your ratings if you don’t

Tips: Standing Out from the Crowd

Profiling

– In-game profiling to select optimal settings per-device

Social

– Hook into social networks

– Use OpenID

Build a live wallpaper

– Own the user's home screen with stats, avatars

Analytics

– Market and in-game

In-app payments

– Consider alternative monetization techniques

Troubleshooting: Development Tools

NVIDIA Debug Manager for Eclipse

– Simplifies setup for native debugger under Eclipse

- developer.nvidia.com/tegra/nvidia-debug-manager-android-ndk

WinGDB

– Use Visual Studio to debug Android NDK applications

- www.wingdb.com

vs-android

– Native MSBuild files for Android toolchain

- vs-android.googlecode.com

Troubleshooting: Performance Tools

NVIDIA PerfHUD ES

– developer.nvidia.com/tegra/perfhud-es

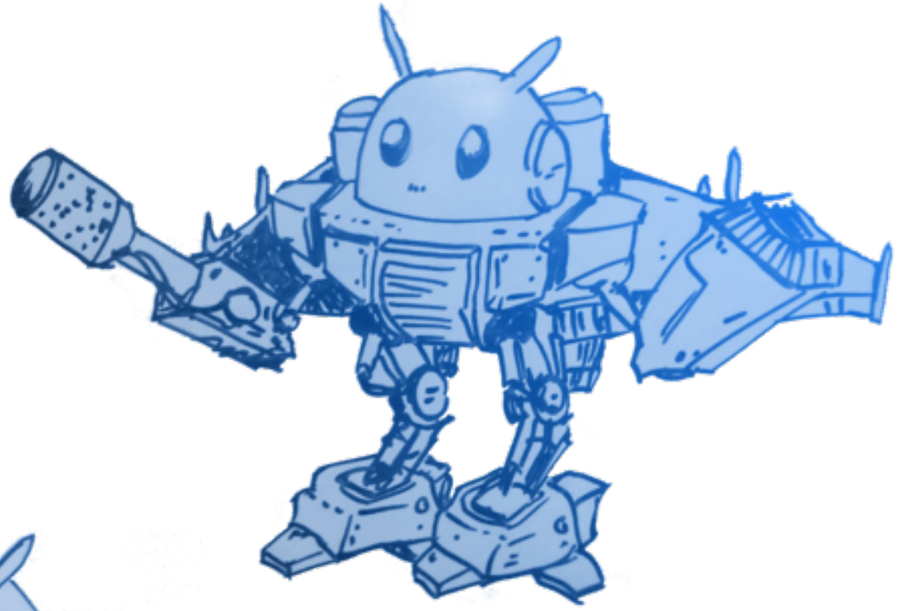
Adreno Profiler

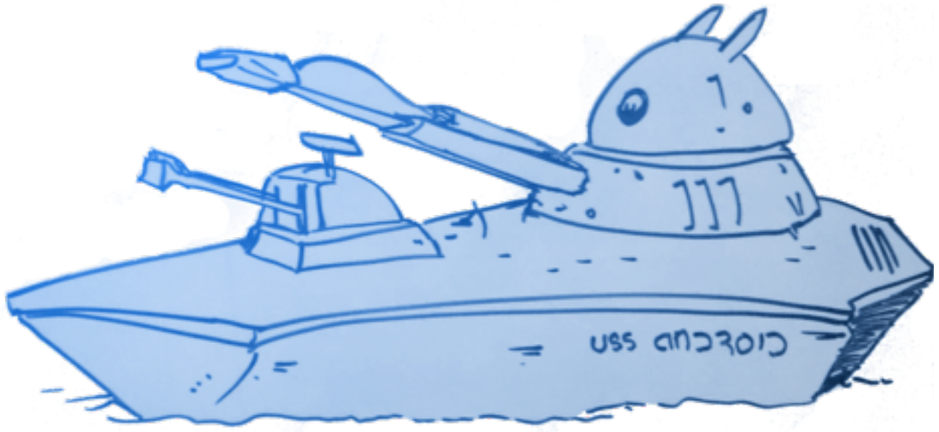
– developer.qualcomm.com/showcase/adreno-profiler

POWERVR Insider Utilities

– Including PVRTrace and PVRTune

– www.imgtec.com/powervr/insider/powervr-utilities.asp





Q&A

developer.android.com/sdk/ndk
@GoogleGameDev
#Android

Feedback: http://goo.gl/NudVs

Google I/O 11