



**nvidia**®

AN INTRODUCTION TO ANDROID DEVELOPMENT  
CS231M | Alejandro Troccoli

# Outline

- Running task on separate threads
- Calling native code from your application
- Introduction to the Camera2 API.

# Long running task

- Long running tasks on the main thread can block the UI
- App looks unresponsive

```
private void longRunningTask( long taskDurationInMs )
{
    long startTime = System.currentTimeMillis();
    mMainText.append("Starting long running task at " + startTime + "\n" );

    long currentTime = startTime;
    do
    {
        try {
            Thread.sleep( taskDurationInMs );
        } catch (InterruptedException e) {
        }

        currentTime = System.currentTimeMillis();

    } while ( currentTime < startTime + taskDurationInMs );

    mMainText.append("Ended long running task at " + currentTime + "\n");
}
```

# Use a separate Thread instead

```
private BufferedWriter mLogWriter = null;  
private Thread mWorkerThread = null;
```

```
// Button2 action on click  
mButton2.setOnClickListener( new View.OnClickListener() {  
  
    @Override  
    public void onClick(View v) {  
        mMainText.setText("Button 2 was pressed!\n");  
        Log.i(TAG, "Button 2 was pressed!");  
        logMessage("Button 2 was pressed!");  
  
        mWorkerThread = new Thread( new Runnable() {  
  
            @Override  
            public void run() {  
                longRunningTask(6000);  
            }  
        });  
  
        mWorkerThread.start();  
  
    }  
});
```

# Use Handlers to update UI

```
private Handler mHandler = null;

private final static int MSG_ASYNC_TASK_STARTED = 0;
private final static int MSG_ASYNC_TASK_COMPLETED = 1;
```

```
mHandler = new Handler( mHandlerCallback );
```

```
private Handler.Callback mHandlerCallback = new Handler.Callback() {

    @Override
    public boolean handleMessage(Message msg) {

        long currentTime = System.currentTimeMillis();
        switch( msg.what )
        {
            case MSG_ASYNC_TASK_STARTED:
                mMainText.append("Async task started at " + currentTime + "\n");
                return true;
            case MSG_ASYNC_TASK_COMPLETED:
                mMainText.append("Async task ended at " + currentTime + "\n");
                return true;
            default:
                // The message was not handled, return false
                return false;
        }
    }
};
```

# Add a Progress dialog

```
private Thread mWorkerThread = null;  
private Handler mHandler = null;  
private ProgressDialog mProgress = null;
```

```
mHandler = new Handler( mHandlerCallback );  
mProgress = new ProgressDialog(this);
```

```
switch( msg.what )  
{  
case MSG_ASYNC_TASK_STARTED:  
    mMainText.append("Async task started at " + currentTime + "\n");  
    mProgress.setTitle("Running async task");  
    mProgress.setMessage("Wait...");  
    mProgress.show();  
    return true;  
case MSG_ASYNC_TASK_COMPLETED:  
    mMainText.append("Async task ended at " + currentTime + "\n");  
    mProgress.dismiss();  
    return true;  
default:  
    // The message was not handled, return false  
    return false;  
}
```

# Outline

- Running task on separate threads
- Calling native code from your application
- Introduction to the Camera2 API.

# Adding native code: Java Native Interface

In the Java class, add a method without implementation and the **native** prefix

```
// Computes the square of a number  
private native int square(int n);
```

Create the jni headers:

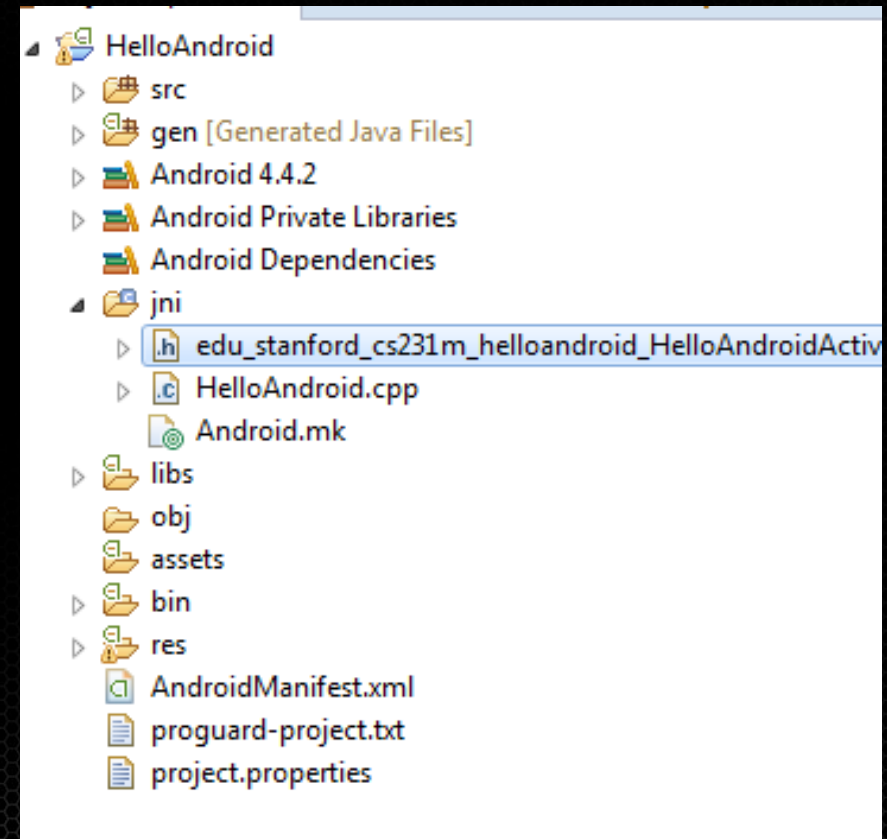
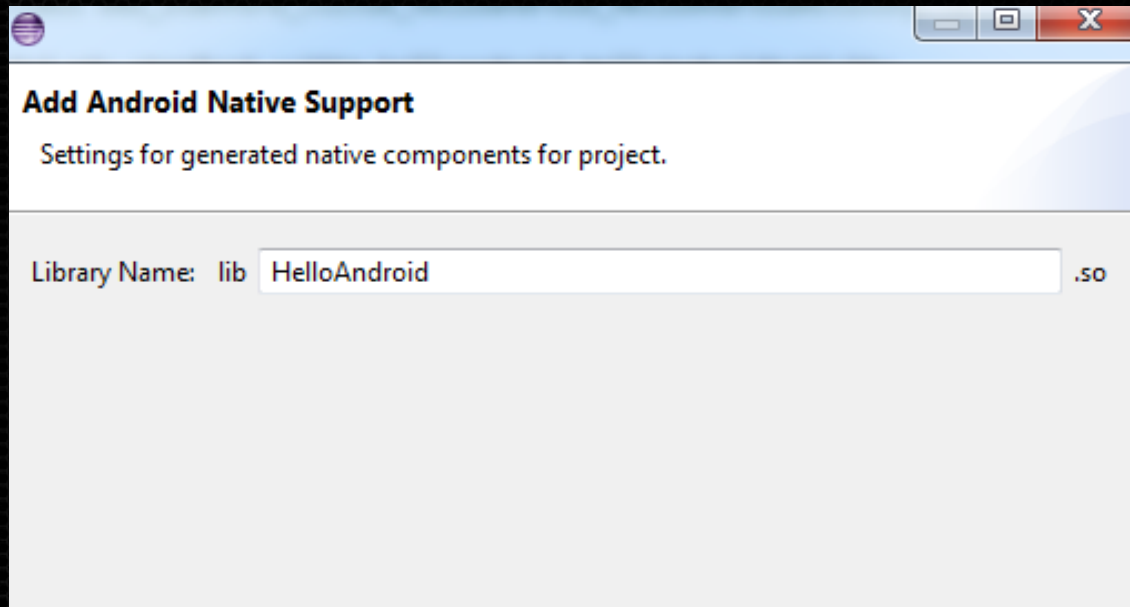
```
> javah -d jni -classpath .\bin\classes edu.stanford.cs231m.helloandroid.HelloAndroidActivity
```

```
#endif  
⊕ * Class:      edu_stanford_cs231m_helloandroid_HelloAndroidActivity  
JNIEXPORT jint JNICALL Java_edu_stanford_cs231m_helloandroid_HelloAndroidActivity_square  
    (JNIEnv *, jobject, jint);
```



# Adding native support

- Right-click on project -> Android Tools -> Add Native Support



# Android.mk

- Makefile for NDK

```
Android.mk HelloAndroidAc... main_layout.xml HelloAndroid M...  
LOCAL_PATH := $(call my-dir)  
  
include $(CLEAR_VARS)  
  
LOCAL_MODULE := HelloAndroid  
LOCAL_SRC_FILES := HelloAndroid.cpp  
  
include $(BUILD_SHARED_LIBRARY)
```

```
Problems Tasks Console Properties  
CDT Build Console [HelloAndroid]  
15:20:46 **** Build of configuration Default for project HelloAndroid ****  
"C:\\work\\tadp\\android-ndk-r9c-windows-x86\\android-ndk-r9c\\ndk-build.cmd" all  
[armeabi] Compile++ thumb: HelloAndroid <= HelloAndroid.cpp  
[armeabi] StaticLibrary : libstdc++.a  
[armeabi] SharedLibrary : libHelloAndroid.so  
[armeabi] Install : libHelloAndroid.so => libs/armeabi/libHelloAndroid.so  
  
15:20:58 Build Finished (took 11s.372ms)
```

# HelloAndroid.cpp

```
#include <jni.h>

#ifdef __cplusplus
extern "C" {
#endif
/*
 * Class:      edu_stanford_cs231m_helloandroid_HelloAndroidActivity
 * Method:     square
 * Signature:  (I)I
 */
JNIEXPORT jint JNICALL Java_edu_stanford_cs231m_helloandroid_HelloAndroidActivity_square
    (JNIEnv *jni, jobject thiz, jint n)
{
    return n * n;
}

#ifdef __cplusplus
}
#endif
```

# Let's run it!

- Modify the Java code to call square and run the app...

# Unfortunately, HelloAndroid has stopped.

```
W      12853  12853  e.  dalvikvm      No implementation found for native Ledu/stanford/cs231m/helloandroid/HelloAndroidActivity;.square:(I)I
E      12853  12853  e.  AndroidRuntime  java.lang.UnsatisfiedLinkError: Native method not found: edu.stanford.cs231m.helloandroid.HelloAndroidActivity.square:(I)I
E      12853  12853  e.  AndroidRuntime  at edu.stanford.cs231m.helloandroid.HelloAndroidActivity.square(Native Method)
E      12853  12853  e.  AndroidRuntime  at edu.stanford.cs231m.helloandroid.HelloAndroidActivity.access$1(HelloAndroidActivity.java:112)
E      12853  12853  e.  AndroidRuntime  at edu.stanford.cs231m.helloandroid.HelloAndroidActivity$1.onClick(HelloAndroidActivity.java:43)
```

Need to load the native library into the Java virtual machine!

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

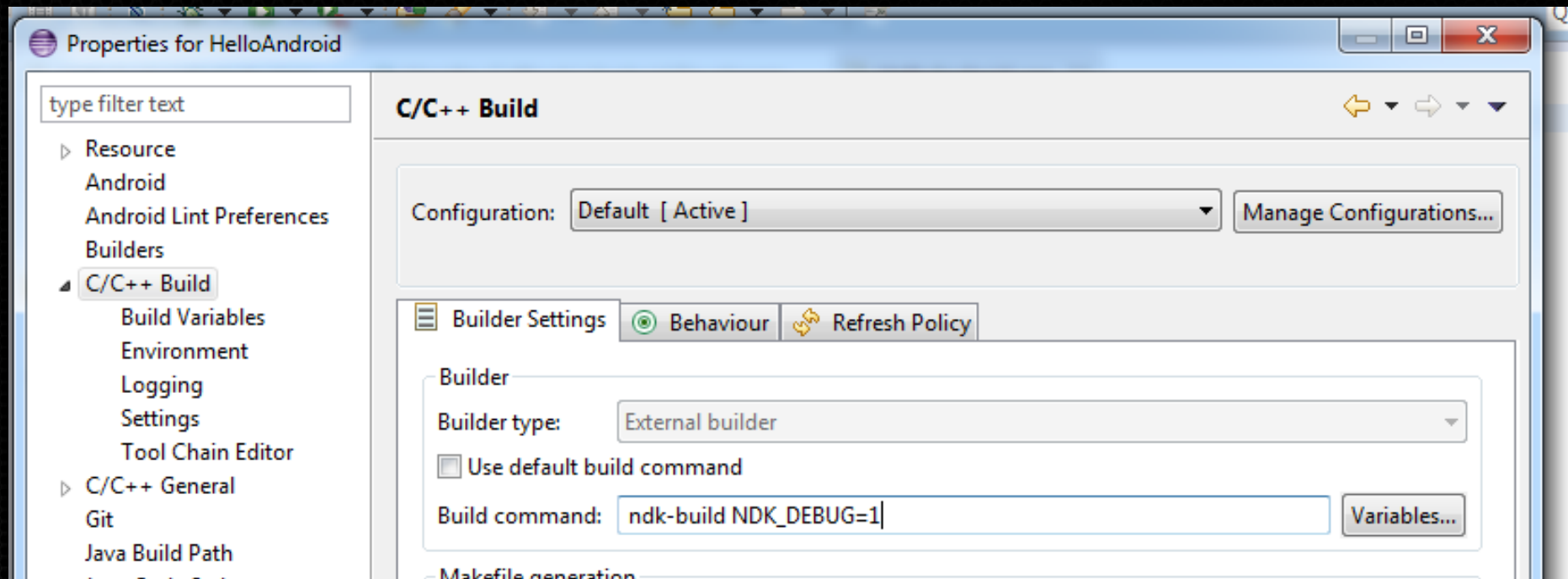
# Application.mk

- Makefile options that are applied to all modules!
  - Target ABI
  - Choice of STL implementation
  - Global compiler options..

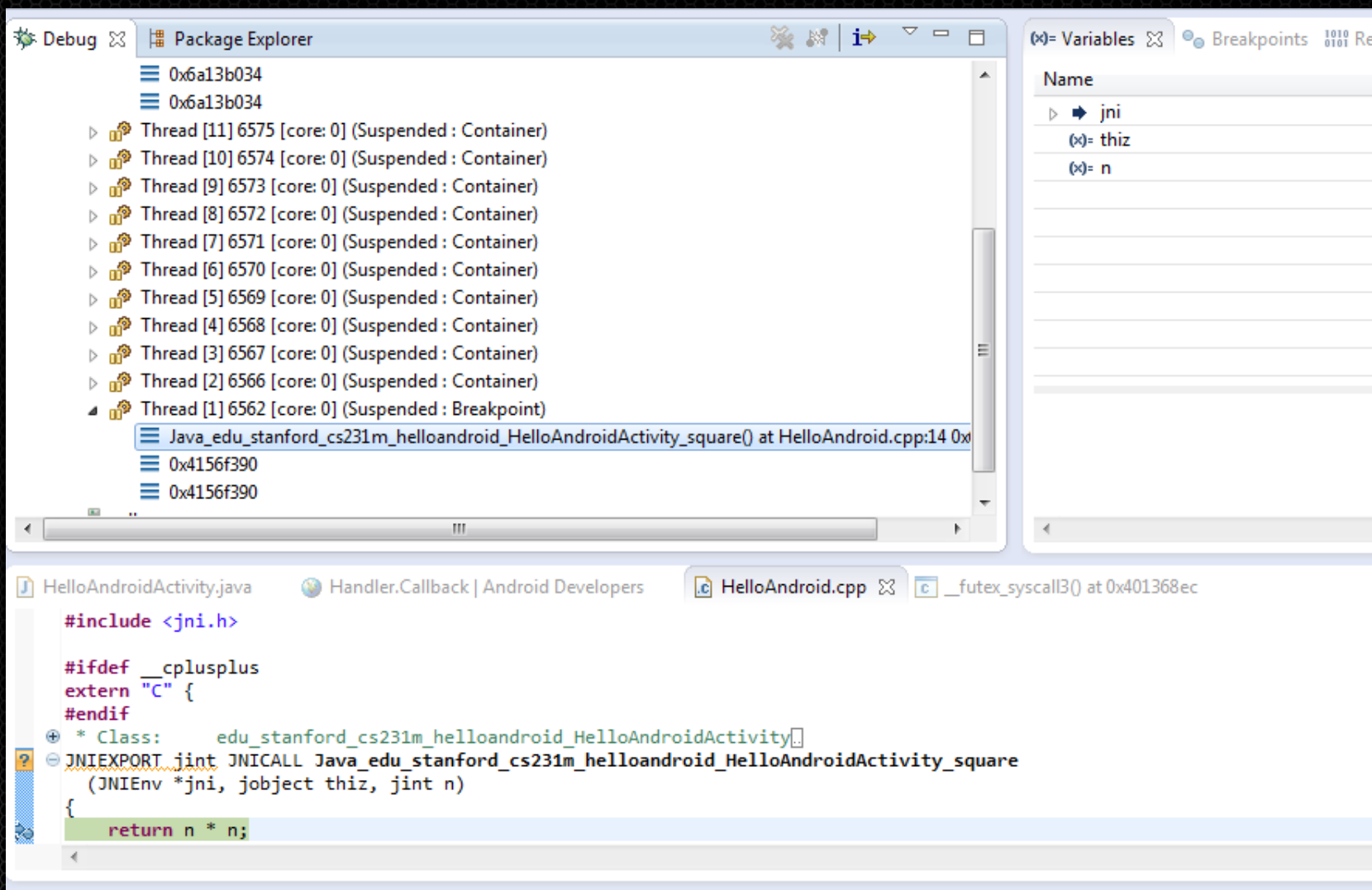
```
APP_PLATFORM := android-19
APP_ABI := armeabi-v7a
APP_STL := gnuSTL_static
```

# Debugging Native Code

- Enable debug build with NDK\_DEBUG



# Launch: Debug as Native app



The screenshot shows the Visual Studio Code interface during a native debug session. The Package Explorer on the left lists threads, with Thread [1] 6562 [core: 0] (Suspended : Breakpoint) selected. The Variables window on the right shows the following variables:

Name
jni
(*)= thiz
(*)= n

The bottom editor shows the C code for the JNI callback:

```
#include <jni.h>

#ifdef __cplusplus
extern "C" {
#endif

* Class:      edu_stanford_cs231m_helloandroid_HelloAndroidActivity
JNIEXPORT jint JNICALL Java_edu_stanford_cs231m_helloandroid_HelloAndroidActivity_square
(JNIEnv *jni, jobject thiz, jint n)
{
    return n * n;
}
```

Need to wait  
for debugger to attach



# Little trick to wait for debugger

```
ifeq (1, $(NDK_DEBUG))
LOCAL_CFLAGS += -DWAIT_FOR_DEBUGGER
endif

LOCAL_MODULE     := HelloAndroid
LOCAL_SRC_FILES  := HelloAndroid.cpp

include $(BUILD_SHARED_LIBRARY)
```

Android.mk

```
#if defined(WAIT_FOR_DEBUGGER)
void waitForDebugger()
{
    static volatile int _debug = 1;
    while( _debug )
    {

    }
}
#else
void waitForDebugger()
{}
#endif
```

Define `waitForDebugger()` and insert a call to wait in your program. Once the debugger attaches, pause the program and set `_debug` to 0.

# Outline

- Running task on separate threads
- Calling native code from your application
- **Introduction to the Camera2 API.**

# Camera 2 API

- See Lecture 3 tutorial on wiki
- Understand:
  - CameraManager
  - CameraDevice
  - CameraCaptureSession
  - Request

# CameraManager

- Lists available cameras in the system
- Provides access to cameras properties
- Allows you to open cameras
  
- Note: Need to add CAMERA permission in AndroidManifest.xml to open a camera.

# CameraDevice

- Gives you **exclusive** access to a camera.
- Allows you to create a CameraCaptureSession.
- Release the camera device when you are done with it!

# CameraCaptureSession

- Sets up the image capture data flow.
- Queues requests to the camera system
  - Single
  - Bursts