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

```java
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

```java
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! 
    ");
        Log.i(TAG, "Button 2 was pressed!");
        Log.e(TAG, "Button 2 was pressed!");

        mWorkerThread = new Thread(new Runnable() {
            @Override
            public void run() {
                longRunningTask(6000);
            }
        });

        mWorkerThread.start();
    }
});
```
Use Handlers to update UI

```java
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 + ",");
                return true;
            case MSG_ASYNC_TASK_COMPLETED:
                mMainText.append("Async task ended at " + currentTime + ",");
                return true;
            default:
                // The message was not handled, return false
                return false;
        }
    }
};
```
Add a Progress dialog

```java
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

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

Create the jni headers:

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

```c
* Class: edu_stanford_cs231m_helloandroid_HelloAndroidActivity
JNIEEXPORT 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

LOCAL_PATH := $(call my-dir)

include $(CLEAR_VARS)

LOCAL_MODULE := HelloAndroid
LOCAL_SRC_FILES := HelloAndroid.cpp

include $(BUILD_SHARED_LIBRARY)

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] StaticLibrary : libSttG++a
[armeabi] SharedLibrary : libHelloAndroid.so
[armeabi] Install : libHelloAndroid.so => libs/armeabi/libHelloAndroid.so

15:20:50 Build Finished (took 11s.372ms)
```cpp
#include <jni.h>

#ifdef __cplusplus
extern "C" {
#endif

/**
 * Class:     edu_stanford_cs231m_helloandroid_HelloAndroidActivity
 * Method:    square
 * Signature: (I)I
 */

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.

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..

```makefile
APP_PLATFORM := android-19
APP_ABI := armeabi-v7a
APP_STL := gnu/stl_static
```
Debugging Native Code

- Enable debug build with NDK_DEBUG
Launch: Debug as Native app

Need to wait for debugger to attach
Little trick to wait for debugger

Android.mk

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

LOCAL_MODULE := HelloAndroid
LOCAL_SRC_FILES := HelloAndroid.cpp
include $(BUILD_SHARED_LIBRARY)
```

```c
#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