Outline

- Overview of the Android Operating System
- Development tools
- Deploying application packages
- Step-by-step application development
- Calling native code from your application
The Android ecosystem

- An open source platform for mobile, embedded and wearable devices
- Google is the principle maintainer
- Other companies contribute to the system.
- Each device manufacturer can customize Android to suite their needs
### Android architecture

<table>
<thead>
<tr>
<th>Layer</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>User applications</strong></td>
<td>Use Java framework and, optionally, native code.</td>
</tr>
<tr>
<td><strong>Android framework</strong></td>
<td>Java classes under com.android</td>
</tr>
<tr>
<td><strong>Native framework layer</strong></td>
<td>User mode C, C++ code - compile to native platform or 32bit compatibility mode on 64 bits.</td>
</tr>
<tr>
<td><strong>Linux Kernel (GPL license)</strong></td>
<td>C code - compiled to native platform (x86, arm, mips)</td>
</tr>
</tbody>
</table>
Android versioning

- Platform version
  - 4.4.2 KitKat
  - 4.4.3 Jellybean MR2

- Framework API level
  - SDK compatibility
  - Each platform version has an API level

- NDK API level
  - API level for native headers
Browsing the Android Source

- Source at:
  - https://android.googlesource.com/

- Porting instructions (for system developers)

- New camera HAL
Reference to the framework APIs

com.android classes

Outline

- Overview of the Android Operating System
- Development tools
- Deploying application packages
- Step-by-step application development
- Calling native code from your application
Tools for application development

- Android SDK
  - Provides the Java framework classes
  - Compiles to java bytecode
  - Class framework is updated with every OS release

- Android NDK
  - C/C++ toolchain for compiling to machine code

- Android platform tools
  - `adb` (android debug bridge) : runs and debugs apps from your dev machine

- Android developer tools
  - Eclipse plug-in for Android
All tools packed in a Virtual Machine

To speed up setup, a virtual machine has been setup.
Tegra Android Development Pack

- Register for an account at:
  - https://developer.nvidia.com/user/register
- Sign-up for Gameworks Registered Developer Program
Tegra Note 7

Click 7 times to enable developer options
Application packages

- .apk files: compressed files
  - class byte code
  - resources (icons, sounds, etc).
  - Binary native files

- All .apks are signed
  - Default development key is created by SDK.
  - When updating an application, signature are checked.
Outline

- Overview of the Android Operating System
- Development tools
- Deploying application packages
- Step-by-step application development
- Calling native code from your application
Installing an application

- From application distribution markets
  - Google Play
  - Amazon AppStore

- From your local computer using `adb`

```
C:\work\tadp\2.0r8\TDK_Samples\tegra_android_native_samples_v10p14\prebuilt> adb install native_globe.apk
7015 KB/s (5826921 bytes in 0.811s)
  pkg: /data/local/tmp/native_globe.apk
Success
C:\work\tadp\2.0r8\TDK_Samples\tegra_android_native_samples_v10p14\prebuilt>
```
Enabling android debug bridge (adb)
Outline

- Overview of the Android Operating System
- Development tools
- Deploying application packages
- **Step-by-step application development**
- Calling native code from your application
Useful adb commands

C:\>adb devices
List of devices attached
0524513118124000E614 device

C:\>adb logcat
--------- beginning of /dev/log/system
I/Vold ( 21?): Vold 2.1 (the revenge) firing up
D/Vold ( 21?): Volume sdcard1 state changing -1 (Initializing) -> 0 (No-Media)
D/Vold ( 21?): Volume usbdrive state changing -1 (Initializing) -> 0 (No-Media)
I/SystemServer( 99?): Entered the Android system server!
I/SystemServer( 99?): Waiting for installld to be ready.
I/Installer ( 99?): connecting...
I/SystemServer( 99?): Power Manager
Hello Android!

In Eclipse
- File -> New -> Android Application Project

New Android Application
- Application Name: HelloAndroid
- Project Name: HelloAndroid
- Package Name: edu.stanford.cs231m.helloandroid
- Minimum Required SDK: API19: Android 4.4.2
- Target SDK: API19: Android 4.4.2
- Compile With: API19: Android 4.4.2
- Theme: Holo Light with Dark Action Bar

Choose the lowest version of Android that your application will support. Lower API levels target more devices, but means fewer features are available. By targeting API 18 and later, you reach approximately 95% of the market.
Hello Android!
Hello Android!
Import an project from existing code
- Package Name / version
- Required SDK and target SDK
- Application/Activities
- Permissions
Provides user interaction

Callbacks for life-cycle management
- `onCreate()`
- `onResume()`
- `onPause()`

An application can have multiple activities...
- Needs one launcher activity...
HelloAndroidActivity

package edu.stanford.cs231m.helloandroid;

import android.app.Activity;

public class HelloAndroidActivity extends Activity {

}
HelloAndroidActivity

Use `onCreate()` to create UI.

```java
public class HelloAndroidActivity extends Activity {

    @Override
    public void onCreate(Bundle settings) {
        TextView txtView = new TextView(this);
        txtView.setText("Hello Android!");
        setContentView(txtView);
    }
}
```
Launch!

Add Activity to AndroidManifest.xml:

```xml
<application
    android:allowBackup="true"
    android:icon="@drawable/ic_Launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme">
    <activity android:name="HelloAndroidActivity" android:label="@string/app_name">
        <intent-filter>
            <category android:name="android.intent.category.LAUNCHER"/>
            <action android:name="android.intent.action.MAIN"/>
        </intent-filter>
    </activity>
</application>
```
Launch! (take 2)
Launch with debugger
Launch (take 3!)
Views

- Can be composed in a tree hierarchy.
- The root View is the argument to setContentView
Creating a layout

```xml
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="horizontal">

  <TextView
      android:id="@id/textMain"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_weight="1"/>

  <LinearLayout
      android:layout_width="200dp"
      android:layout_height="wrap_content"
      android:orientation="vertical">

    <Button
        android:id="@id/button1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Button1"/>

    <Button
        android:id="@id/button2"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Button2"/>
  </LinearLayout>

</LinearLayout>
```
Accessing layout elements from Activity

```java
public class HelloAndroidActivity extends Activity {

    private TextView mMainText;
    private Button mButton1;
    private Button mButton2;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        // Setup the layout hierarchy
        setContentView(R.layout.main_layout);

        // Find the layout elements
        mMainText = (TextView) findViewById(R.id.txtMain);
        mButton1 = (Button) findViewById(R.id.button1);
        mButton2 = (Button) findViewById(R.id.button2);

        mMainText.setText("HelloAndroid!");

        super.onCreate(savedInstanceState);
    }
}
```

Use `findViewById`:

- `R.id.name` corresponds to the name given in the xml file.
Event listeners (and logging, too)

```java
// Button1 action on click
mButton1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        mMainText.setText("Button 1 was pressed!");
        Log.i(TAG, "Button 1 was pressed!");
    }
});

// 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!");
    }
});
```
Logcat

!Window -> Show View -> Other -> Android -> Logcat

<table>
<thead>
<tr>
<th>L</th>
<th>Time</th>
<th>PID</th>
<th>TID</th>
<th>Application</th>
<th>Tag</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>03-27 13:30:2...</td>
<td>304</td>
<td>304</td>
<td></td>
<td></td>
<td>nvudd Automatically enabled</td>
</tr>
<tr>
<td>I</td>
<td>03-27 13:30:2...</td>
<td>304</td>
<td>304</td>
<td></td>
<td></td>
<td>nvudd:tbv TBC probe cpu limit=89.00</td>
</tr>
<tr>
<td>I</td>
<td>03-27 13:30:2...</td>
<td>304</td>
<td>304</td>
<td></td>
<td></td>
<td>nvudd:tbv TBC probe ext original=43000, backoff</td>
</tr>
<tr>
<td>I</td>
<td>03-27 13:30:2...</td>
<td>8347</td>
<td>8347</td>
<td>edu.stanford.cs231m.helloandroid</td>
<td></td>
<td>HelloAndro... Button 1 was pressed!</td>
</tr>
<tr>
<td>I</td>
<td>03-27 13:30:2...</td>
<td>8347</td>
<td>8347</td>
<td>edu.stanford.cs231m.helloandroid</td>
<td></td>
<td>HelloAndro... Button 2 was pressed!</td>
</tr>
<tr>
<td>D</td>
<td>03-27 13:30:2...</td>
<td>233</td>
<td>263</td>
<td></td>
<td>volt_cap</td>
<td>VC: Volt update change is -18024.351</td>
</tr>
<tr>
<td>I</td>
<td>03-27 13:30:2...</td>
<td>8347</td>
<td>8347</td>
<td>edu.stanford.cs231m.helloandroid</td>
<td></td>
<td>HelloAndro... Button 2 was pressed!</td>
</tr>
<tr>
<td>D</td>
<td>03-27 13:30:2...</td>
<td>8347</td>
<td>8347</td>
<td>edu.stanford.cs231m.helloandroid</td>
<td>dalvikvm</td>
<td>GC_FOR_ALLOC freed 309K, 12% free 298M</td>
</tr>
<tr>
<td>I</td>
<td>03-27 13:30:2...</td>
<td>8347</td>
<td>8347</td>
<td>edu.stanford.cs231m.helloandroid</td>
<td></td>
<td>HelloAndro... Button 2 was pressed!</td>
</tr>
<tr>
<td>I</td>
<td>03-27 13:30:2...</td>
<td>8347</td>
<td>8347</td>
<td>edu.stanford.cs231m.helloandroid</td>
<td></td>
<td>HelloAndro... Button 2 was pressed!</td>
</tr>
<tr>
<td>I</td>
<td>03-27 13:30:2...</td>
<td>8347</td>
<td>8347</td>
<td>edu.stanford.cs231m.helloandroid</td>
<td></td>
<td>HelloAndro... Button 2 was pressed!</td>
</tr>
<tr>
<td>I</td>
<td>03-27 13:30:2...</td>
<td>8347</td>
<td>8347</td>
<td>edu.stanford.cs231m.helloandroid</td>
<td></td>
<td>HelloAndro... Button 2 was pressed!</td>
</tr>
<tr>
<td>I</td>
<td>03-27 13:30:2...</td>
<td>8347</td>
<td>8347</td>
<td>edu.stanford.cs231m.helloandroid</td>
<td></td>
<td>HelloAndro... Button 2 was pressed!</td>
</tr>
<tr>
<td>I</td>
<td>03-27 13:30:2...</td>
<td>304</td>
<td>304</td>
<td></td>
<td></td>
<td>nvudd Automatically disabled</td>
</tr>
<tr>
<td>Y</td>
<td>03-27 13:30:3...</td>
<td>2679</td>
<td>2679</td>
<td></td>
<td>NVSS</td>
<td>16  ---Desk---</td>
</tr>
<tr>
<td>Y</td>
<td>03-27 13:30:3...</td>
<td>226</td>
<td>1114</td>
<td></td>
<td>nvaudio_hw</td>
<td>nvaudio_dev_set_parameters : EQ_MODE</td>
</tr>
</tbody>
</table>
private BufferedWriter openLogFile()
{
    File appExternalDir = new File( Environment.getExternalStorageDirectory(),
        "HelloAndroid");

    if ( !appExternalDir.exists() )
    {
        if ( appExternalDir.mkdirs() )
        {
            Log.i(TAG, "External storage directory created: " + appExternalDir.toString());
        }
        else
        {
            Log.e(TAG, "Failed to create directory " + appExternalDir.toString());
            return null;
        }
    }

    File logFile = new File( appExternalDir, "log.txt");

    BufferedWriter writer = null;
    try {
        writer = new BufferedWriter( new FileWriter(logFile));
    } catch (IOException e) {
        Log.e(TAG, "Failed to create file " + logFile.toString());
        return null;
    }

    return writer;
}
Writing to the log file

```java
private void logMessage( String message )
{
    if ( mLogWriter != null )
    {
        try {
            mLogWriter.write(message);
            mLogWriter.newLine();
            mLogWriter.flush();
        } catch (IOException e) {
            Log.e(TAG, "Failed to write to log file");
        }
    }
}
```

```xml
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="edu.stanford.cs231n.helloandroid"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="19"
        android:targetSdkVersion="19" />

    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```
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!");
        logMessage("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 + ",\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

```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

- Overview of the Android Operating System
- Development tools
- Deploying application packages
- Step-by-step application development
- Calling native code from your application
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:

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

```c
#ifdef __cplusplus
extern "C"
#endif

int 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)
HelloAndroid.cpp

#include <jni.h>

#endif __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;
}

#endif __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 lib/edu/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");
}
```
Makefile options that are applied to all modules!

- Target ABI
- Choice of STL implementation
- Global compiler options.
Enable debug build with NDK_DEBUG
Launch: Debug as Native app

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

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