Outline

- Overview of the Android Operating System
- Development tools
- Deploying application packages
- Step-by-step application development
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 suit 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 - compiled 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**
  - 5.0 Lollipop
  - 4.4 KitKat

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

- **NDK API level**
  - API level for native headers

- **Distribution**
Browsing the Android Source

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

Porting instructions (for system developers)
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
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
  - Android studio (doesn’t yet fully support all NDK features)
Setup options

- Tegra Android Development pack
- Install components manually
- Use a Linux virtual machine with the tools pre-installed.
<table>
<thead>
<tr>
<th>Title</th>
<th>Version</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>PhysX: Core PhysX SDK</td>
<td>3.3.0</td>
<td>2014/02/21</td>
</tr>
<tr>
<td>PhysX: APEX SDK</td>
<td>1.3.0</td>
<td>2014/02/21</td>
</tr>
<tr>
<td>Tegra Android Development Pack</td>
<td>2.0b8</td>
<td>2013/03/18</td>
</tr>
<tr>
<td>PhysX: 3D Studio MAX DCC plug-in</td>
<td>3.0.0</td>
<td>2014/02/21</td>
</tr>
<tr>
<td>PhysX: Maya DCC plug-in</td>
<td>3.0.0</td>
<td>2014/02/21</td>
</tr>
<tr>
<td>PhysX Lab</td>
<td>1.3.0</td>
<td>2014/02/21</td>
</tr>
</tbody>
</table>
Tegra Android Development Pack

- Register for an account at:
  - https://developer.nvidia.com/user/register

- Sign-up for Gameworks Registered Developer Program
NVIDIA Shield Tablet

Click 7 times to enable developer options
Outline

- Overview of the Android Operating System
- Development tools
- **Deploying application packages**
- Step-by-step application development
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.
Installing an application

- From application distribution markets
  - Google Play
  - Amazon AppStore

- From your local computer using adb
Enabling android debug bridge (adb)
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 usbd2 state changing -1 (Initializing) -> 0 (No-Media)
I/SystemServer( 99?): Entered the Android system server!
I/SystemServer( 99?): Waiting for installd to be ready.
I/Installer( 99?): connecting...
I/SystemServer( 99?): Power Manager

^C
Outline

- Overview of the Android Operating System
- Development tools
- Deploying application packages
- Step-by-step application development
Hello Android!

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

New Android Application
Configure Project

- Create custom launcher icon
- Create activity

- Mark this project as a library
- Create Project in Workspace

Location: C:\work\workspaces\cs231\HelloAndroid

- Add project to working sets

Working sets:

Finish button highlighted.
Hello Android!
Import an project from existing code
AndroidManifest.xml

- Package Name / version
- Required SDK and target SDK
- Application/Activities
- Permissions
Android Activity

- Provides user interaction

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

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

Java Class
Create a new Java class.

Source folder: HelloAndroid/src
Package: edu.stanford.cs231m.helloandroid

Name: HelloAndroidActivity
Modifiers: public
Superclass: android.app.Activity

Which method stubs would you like to create?
- public static void main(String[] args)
- Constructors from superclass
- Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)
- Generate comments

Finish   Cancel

Package Explorer

HelloAndroid
src
edu.stanford.cs231m.helloandroid
HelloAndroidActivity.java

import android.app.Activity;

public class HelloAndroidActivity extends Activity {

}
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>
    <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="Button" />
        <Button
            android:id="@+id/button2"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Button" />
    </LinearLayout>
</LinearLayout>
```
Accessing layout elements from Activity

Use findViewById

R.id.name corresponds to the name given in the xml file

```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);
    }
}
```
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>Time</th>
<th>PID</th>
<th>TID</th>
<th>Application</th>
<th>Tag</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>03-27 13:30:2...</td>
<td>304</td>
<td>304</td>
<td></td>
<td>nvuad</td>
<td>Automatically enabled</td>
</tr>
<tr>
<td>03-27 13:38:2...</td>
<td>304</td>
<td>304</td>
<td></td>
<td>nvuad:tbc</td>
<td>TBC probe cpu limit=89.00</td>
</tr>
<tr>
<td>03-27 13:38:2...</td>
<td>8347</td>
<td>8347</td>
<td>edu.stanford.cs231m.helloandroid</td>
<td>HelloAndro...</td>
<td>Button 1 was pressed!</td>
</tr>
<tr>
<td>03-27 13:38:2...</td>
<td>8347</td>
<td>8347</td>
<td>edu.stanford.cs231m.helloandroid</td>
<td>HelloAndro...</td>
<td>Button 2 was pressed!</td>
</tr>
<tr>
<td>03-27 13:38:2...</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>03-27 13:38:2...</td>
<td>8347</td>
<td>8347</td>
<td>edu.stanford.cs231m.helloandroid</td>
<td>HelloAndro...</td>
<td>Button 2 was pressed!</td>
</tr>
<tr>
<td>03-27 13:38:2...</td>
<td>8347</td>
<td>8347</td>
<td>edu.stanford.cs231m.helloandroid</td>
<td>dalvikvm</td>
<td>GC_FOR_ALLOC freed 309K, 12% free 2988M</td>
</tr>
<tr>
<td>03-27 13:38:2...</td>
<td>8347</td>
<td>8347</td>
<td>edu.stanford.cs231m.helloandroid</td>
<td>HelloAndro...</td>
<td>Button 2 was pressed!</td>
</tr>
<tr>
<td>03-27 13:38:2...</td>
<td>8347</td>
<td>8347</td>
<td>edu.stanford.cs231m.helloandroid</td>
<td>HelloAndro...</td>
<td>Button 2 was pressed!</td>
</tr>
<tr>
<td>03-27 13:38:2...</td>
<td>8347</td>
<td>8347</td>
<td>edu.stanford.cs231m.helloandroid</td>
<td>HelloAndro...</td>
<td>Button 2 was pressed!</td>
</tr>
<tr>
<td>03-27 13:38:2...</td>
<td>8347</td>
<td>8347</td>
<td>edu.stanford.cs231m.helloandroid</td>
<td>HelloAndro...</td>
<td>Button 2 was pressed!</td>
</tr>
<tr>
<td>03-27 13:38:2...</td>
<td>8347</td>
<td>8347</td>
<td>edu.stanford.cs231m.helloandroid</td>
<td>HelloAndro...</td>
<td>Button 2 was pressed!</td>
</tr>
<tr>
<td>03-27 13:38:2...</td>
<td>8347</td>
<td>8347</td>
<td>edu.stanford.cs231m.helloandroid</td>
<td>HelloAndro...</td>
<td>Button 2 was pressed!</td>
</tr>
<tr>
<td>03-27 13:38:2...</td>
<td>8347</td>
<td>8347</td>
<td>edu.stanford.cs231m.helloandroid</td>
<td>HelloAndro...</td>
<td>Button 2 was pressed!</td>
</tr>
<tr>
<td>03-27 13:38:2...</td>
<td>304</td>
<td>304</td>
<td></td>
<td>nvuad</td>
<td>Automatically disabled</td>
</tr>
<tr>
<td>03-27 13:38:2...</td>
<td>2679</td>
<td>2679</td>
<td></td>
<td>NVSS</td>
<td>16 ---Desk---</td>
</tr>
<tr>
<td>03-27 13:38:3...</td>
<td>226</td>
<td>1114</td>
<td></td>
<td>nvaudio_hw</td>
<td>nvaudio_dev_set_parameters : EQ_MODE</td>
</tr>
</tbody>
</table>
Creating a log file

```java
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;
}
```
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");
        }
    }
}
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) {
        }
    }
    while ( currentTime < startTime + taskDurationInMs );

    currentTime = System.currentTimeMillis();
    mMainText.append("Ended long running task at " + currentTime + "\n");
}
```
Use a separate Thread instead

```java
import java.io.FileWriter;
import java.io.BufferedWriter;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

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

```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;
}
```