Advanced Topic in Computational Vision
Mini-Project in Mobile Vision

Intro to Android Development

CS 202-1-4171

Computer Science Department, BGU
Course Schedule

• Android Development (week 5).
• OpenCV for Android Projects (week 6).
• Submission of Project Proposals (week 7).
• Submission of Project Report (week 13).
• Project Demos + Submission of Project Software (end of exam period).
Outline

• Introduction
• Creating Development Environment
• Application Fundamentals
• Graphic User Interface (GUI)
• Getting Started (Hello World)
What is Android?

Android is a software stack for mobile devices that includes an operating system, middleware and key applications.

http://developer.android.com/
Android Background

• Android was founded in Palo Alto (2003).
• Google acquired Android (2005).
• Open Handset Alliance formed to develop open standards for mobile devices (2007).
Android Background

• Android available as open source (2008).
• Over 190 millions Android devices in use.
• Market Share:

<table>
<thead>
<tr>
<th></th>
<th>Feb’10</th>
<th>May’10</th>
<th>Apr’11</th>
</tr>
</thead>
<tbody>
<tr>
<td>RIM</td>
<td>42.1%</td>
<td>41.7%</td>
<td>29%</td>
</tr>
<tr>
<td>Apple</td>
<td>25.4%</td>
<td>24.4%</td>
<td>25%</td>
</tr>
<tr>
<td>Google</td>
<td>9%</td>
<td>13%</td>
<td>33%</td>
</tr>
<tr>
<td>Microsoft</td>
<td>15.1%</td>
<td>13.2%</td>
<td>7.7%</td>
</tr>
<tr>
<td>Palm</td>
<td>5.4%</td>
<td>4.8%</td>
<td>2.9%</td>
</tr>
</tbody>
</table>
Android Architecture
Android S/W Stack - Application

- Android provides a set of core applications: Email Client, SMS Program, Calendar, Maps Browser, Contacts, Etc...
- All applications are written using the Java language.
Android S/W Stack – App Framework

- Enabling and simplifying the reuse of components
  - Developers have full access to the same framework APIs used by the core applications.
  - Users are allowed to replace components.
Android S/W Stack - Libraries

• Including a set of C/C++ libraries used by components of the Android system
• Exposed to developers through the Android application framework
Android S/W Stack - Runtime

• Core Libraries
  ✓ Providing most of the functionality available in the core libraries of the Java language

✓ APIs
  ➢ Data Structures
  ➢ Utilities
  ➢ File Access
  ➢ Network Access
  ➢ Graphics
  ➢ Etc
Android S/W Stack – Runtime

• Dalvik Virtual Machine
  ✓ Providing environment on which every Android application runs
    ➢ Each Android application runs in its own process, with its own instance of the Dalvik VM.
    ➢ Dalvik has been written such that a device can run multiple VMs efficiently.
  ✓ Register-based virtual machine
Android S/W Stack – Runtime

• Dalvik Virtual Machine (Cont)
  ✓ Executing the Dalvik Executable (.dex) format
    ➢ .dex format is optimized for minimal memory footprint.
    ➢ Compilation
Android S/W Stack – Linux Kernel

- Relying on Linux Kernel 2.6 for core system services
  - Memory and Process Management
  - Network Stack
  - Driver Model
  - Security
- Providing an abstraction layer between the H/W and the rest of the S/W stack
Outline

• Introduction
• Creating Development Environment
• Application Fundamentals
• Graphic User Interface (GUI)
• Getting Started (Hello World)
Creating Development Environment

• The Android SDK supports several different integrated development environments (IDEs). Here we will focus on Eclipse because it is the IDE that is best integrated with the SDK and free.

• No matter which operating system you are using, you will need essentially the same set of tools:
  – Sun's Java Development Kit (JDK)
  – Eclipse IDE for Java Developer
  – Android Software Developer Kit (SDK)
  – Android Developer Tool (ADT), Plug-in for Eclipse
  – Phone driver (ADB)

See Course Website for Step-by-Step Instructions
Outline

• Introduction
• Creating Development Environment
• Application Fundamentals
• Graphic User Interface (GUI)
• Getting Started (Hello World)
Application Fundamental

• Understand applications and their components
• Concepts:
  – activity
  – service
  – broadcast receiver
  – content provider
  – intent
  – Android-Manifest
Applications

• Written in Java (Android Package, .apk suffix)
• Good separation (and corresponding security) from other applications:
  – Each application runs in its own process
  – Each process has its own separate VM
  – Each application is assigned a unique Linux user ID – by default files of that application are only visible to that application (can be explicitly exported)
Application Components

- **Activities** – visual user interface focused on a single thing a user can do
- **Services** – no visual interface – they run in the background
- **Content Providers** – allow data exchange between applications
- **Broadcast Receivers** – receive and react to broadcast announcements
Application Components

• Any application can start another application’s component.

• Android applications don't have a single entry point (there's no main() function, for example).

• To activate a component in another application, you must deliver a message to the system that specifies your intent to start a particular component.
Application Components

- **Intent** - defines a message to activate a component by passing an `Intent` to `startActivity`, `startService()`, `sendBroadcast()`.

- **Manifest File** -
  - Declaring the application's components
  - Identify any user permissions the application requires (i.e., internet access)
  - Declare hardware used (i.e., camera, bluetooth)
Activities

• Provides a screen with which users can interact in order to do something.
• One activity in an application is specified as the "main".
• Each activity can then start another activity in order to perform different actions.
• When a new activity starts, it is pushed onto the back stack and takes user focus (the previous activity is stopped).
Activities start each other
Creating an Activity

• Create a subclass of Activity (or an existing subclass of it).

• `onCreate()`: The system calls this when creating your activity.

• `onPause()`: The system calls this method as the first indication that the user is leaving your activity (another activity becomes in the foreground and has focus, but this one is still visible).
Activities – User Interface

• Each activity has a default window to draw in
• The content of the window is a view or a group of views (derived from View or ViewGroup)
• Example of views: buttons, text fields, scroll bars, menu items, check boxes, etc.
• View(Group) made visible via Activity.setContentView() method.
**Services**

- Does not have a visual interface
- Runs in the background indefinitely
- Examples
  - Network Downloads
  - Playing Music
  - TCP/UDP Server
- You can bind to an existing service and control its operation
**Content Providers**

- Store and retrieve data and make it accessible to all applications.
- It’s the only way to transfer data between applications in Android (no shared files, shared memory, pipes, etc.)
- Extends the class `ContentProvider`.
- Applications use a `ContentResolver` object to access the data provided via a `ContentProvider`. 
Broadcast Receivers

• Receive and react to broadcast announcements
• Extend the class BroadcastReceiver
• Examples of broadcasts:
  – Low battery, power connected, shutdown, time zone changed, etc.
  – Other applications can initiate broadcasts
**Intents**

- An intent is an `Intent` object with a message content.
- Activities, services and broadcast receivers are started by intents. ContentProviders are started by ContentResolvers:
  - An **activity** is started by `Context.startActivity(Intent intent)`
  - A **service** is started by `Context.startService(Intent service)`
  - An application can initiate a **broadcast** by using an `Intent` in any of `Context.sendBroadcast(Intent intent)`. 
Android Manifest

• Declare your activity in the manifest file in order for it to be accessible to the system

```xml
<manifest ...
  <application ...
    <activity android:name="ExampleActivity" />
    ...
  </application ...
  ...
</manifest>
```
Android Manifest (Intent Filters)

• Declare how other application components may activate it.

<activity android:name=".ExampleActivity"
    android:icon="@drawable/app_icon">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category
            android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
Shutting down components

• Activities
  – Can terminate itself via finish();
  – Can terminate other activities it started via finishActivity();

• Services
  – Can terminate via stopSelf(); or Context.stopService();

• Content Providers
  – Are only active when responding to ContentResolvers

• Broadcast Receivers
  – Are only active when responding to broadcasts
Outline

• Introduction
• Creating Development Environment
• Application Fundamentals
• Graphic User Interface (GUI)
• Getting Started (Hello World)
Graphic User Interface

• Familiarize with the main types of GUI components
• Concepts:
  – Layouts
  – Widgets
  – Menus
View Hierarchy

- All the views in a window are arranged in a tree
- You show the tree by calling `setContentView(rootNode)` in the activity
**Layout**

- Defines how elements are positioned relative to each other (next to each other, under each other, in a table, grid, etc.)
- Can have a different layouts for each ViewGroup.
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical">
    <TextView android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a Button" />
</LinearLayout>
**Widgets**

- All are View objects
- Examples:
  - TextFields
  - EditFields
  - Buttons
  - Checkboxes
  - RadioButtons
  - etc.
Input Events

• Usually handled by defining a Listener of the form On<something>Listener and register it with the View

• For example:
  – OnClickOnListener() for handling clicks on Buttons or Lists
  – OnTouchListener() for handling touches
  – OnKeyListerner() for handling key presses

• Alternatively, Override an existing callback if we implemented our own class extending View
Menus

- onCreateOptionsMenu()
- onOptionsItemSelected()
Outline

• Introduction
• Creating Development Environment
• Application Fundamentals
• Graphic User Interface (GUI)
• Getting Started (Hello World)
Hello Android

Hello, Android
Hello World, HelloAndroid!
Package Content

- **All source code here**
- **Java code for our activity**
- **Generated Java code**
  - Helps link resources to Java code
- **All non-code resources**
- **Images**
- **Layout of the activity**
- **Strings used in the program**
- **Android Manifest**
Revised HelloAndroid.java

```java
package com.example.helloandroid;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class HelloAndroid extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);
        tv.setText("Hello, Android – by hand");
        setContentView(tv);
    }
}
```

Inherit from the Activity Class

Set the view “by hand” – from the program
Run it!
<?xml version="1.0" encoding="utf-8"?>
<TextView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/textview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:text="@string/hello"/>

Further redirection to /res/values/strings.xml
<res/values/strings.xml>

<?xml version="1.0" encoding="utf-8"?>

<resources>
    <string name="hello">Hello World, HelloAndroid – by resources!</string>
    <string name="app_name">Hello, Android</string>
</resources>
package com.example.helloandroid;

import android.app.Activity;
import android.os.Bundle;
public class HelloAndroid extends Activity {

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.main);
    }
}

Set the layout of the view as described in the main.xml layout
package com.example.helloandroid;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class id {
        public static final int textview=0x7f050000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
Run it!
introduce a bug

package com.example.helloandroid;

import android.app.Activity;
import android.os.Bundle;

public class HelloAndroid extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Object o = null;
        o.toString();
        setContentView(R.layout.main);
    }
}
Run it!
For Next Week

• Android Developer Tutorials
  – Hello World
  – Hello Views
  – Etc...

• Run on Android Virtual Device