Android Development
IMDL SPRING 2015

Why Android

• Learn a commonly used Tool
• Use it as a control device for robot
• Use it as a information relay
• Use it as a sensor
  • IMU
  • Camera
  • GPS
How to Develop

• Download one of 3 possible IDEs and the Android SDK
  • Eclipse/Android Package
    • http://developer.android.com/sdk/installing/index.html
  • IntelliJ
    • http://www.jetbrains.com/idea/features/android.html
  • Android Studio
    • http://developer.android.com/sdk/installing/studio.html
• Use IDE and Android Virtual Device
  • Create layout through XML (or programmatically)
  • Code Application in Java-Like environment
  • Test on Android Virtual Device (AVD) or physical phone

Android Studio
Application Project Layout

- src - holds all code folders, resource folders, and configuration files
- java – holds all code for application
- res - holds resources for your application (layouts, strings, pictures).
- drawables – holds icons, images, etc.
- menu,layout – holds xml files for activity (page) layouts or menu layout
- values - holds xml file for strings, styles, array values, etc.
- AndroidManifest.xml
- build.gradle

AndroidManifest.xml

- Define application components
  - Identify user permissions (sensors, internet access, etc)
  - Declare minimum and target API to use
  - Declare hardware and software features used
  - Declare API libraries used (Google Maps Library)
  - Declare application specific values (name, launcher, theme)
  - Declare various activities that run in application
Activities

• Single screen with a user interface
• The application starts with the MainActivity
• MainActivity can start other activities as needed
  • startActivity()
  • startActivityForResult()
• Activities consists of a hierarchy of views
  • Widgets – visual elements (buttons, text fields, etc.)
  • Layouts – Provide a layout for child views
• Activities follow the Activity Lifecycle

build.gradle

• Define SDK version to use for compiling
• Define dependencies such as:
  • Libraries
  • Projects
  • Files
• Handles transition for projects that use the old compiling method (Maven)
Activity Lifecycle

- The resumed state is the active running state
- onCreate is used for initial setup
- onResume is used for returning from the paused state
- onStart is used when first running or coming back from a hidden state
- onPause prepares the app to be paused (cancel battery draining functions)
- onStop prepares app for shutting down
- You don’t always need every lifecycle command

Borrowed from Android Developer site: http://developer.android.com/training/basics/activity-lifecycle/starting.html
Creating Activity Layout

- Open Layout XML file. Edit by:
  - Graphical Layout
  - XML file
- Create Layouts to contain widgets (buttons, text, etc)
- Create widgets to interact or show information
- Program lifecycle callbacks (and various other functions)

Layouts

- Usually defined by width, height, and orientation parameters
  <LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
  </LinearLayout>
Linear Layouts

- Widgets in horizontal or vertical ordered row

Relative Layouts

- Specify location of objects relative to each other
List View Layouts

- View Group that displays a list of scrollable items

Grid View Layouts

- View Group that displays a two-dimensional, scrollable grid
Button widget

• Widget for button interface

```xml
<Button android:id="@+id/my_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/my_button_text"/>
```

EditText widget

• Widget for entering text

```xml
<EditText android:id="@+id/txt_entry"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="@string/enter_txt"
    android:inputType="text"/>
```
TextView widget

• Widget for entering text
• Simple labels without the ability to edit during application runtime

```xml
<TextView android:id="@+id/txbbox"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/tbboxinfo"/>
```

Toast widget

• Quick message for user
• Toast.makeText(this, "Not connected",
    Toast.LENGTH_SHORT).show();
A few XML points

- `match_parent` – make view as big as its parent
- `wrap_content` – make view just large enough to fit internal content
- `android:id` – id (variable name) to be used through code to grab handle of widget
- `@string` – grab resource from string.xml under res/values

Starting an Application

- The first step when creating an application is deciding on the graphical layout
  - Navigate to the main layout xml file
  - Create layouts to format your display in an ordered format
  - Create text_view labels, edit_text boxes, buttons, etc.
Creating the Program

• The next step is to begin editing the MainActivity java file
  • In onCreate function, set the layout xml view to be used for the activity
    • This is the file that was edited on the last step
      setContentView(R.layout.activity_main);
  • Create handles to buttons
    Button btn = (Button) findViewById(R.id.mybutton);
  • Create Click Listeners for buttons
    btn.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        myFancyMethod(v);
      }
    });

Creating the Program

• Create handle to EditText
  EditText text = (EditText) findViewById(R.id.editText1);
• Set value of EditText
  text.setText("Test1");
If Using Android Studio

How to Start

Before installing Android Studio

• Java development kit
Setup Project

• Click “New Project...”

Setup Project

• Select appropriate SDK for application
Setup Project

• Choose layout

Setup Project

• Customize project names
Setup Project

- Activity_main.xml
Setup Project

- MainActivity

Advanced Features

- There are various features android has that benefits development
  - Sensors - Accelerometer, Gyro, Magnetometer
  - Location - GPS
  - Bluetooth
  - Camera/OpenCV
SensorManager

- The SensorManager gives control of the onboard sensors, allowed data gathering of the accelerometer, gyro, and magnetometer
- Make sure your application implements SensorEventListener
- Create a variable of class SensorManager
- After registering the sensor, the onSensorChanged function will trigger with any sensor changes
  - The function returns the type of sensor that is changed
    - TYPE_ACCELEROMETER
    - TYPE_MAGNETIC_FIELD
    - TYPE_GYROSCOPE
- Confirm permissions in AndroidManifest.xml
  - <uses-permission android:name="android.permission.INTERNET"/>
  - <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
  - <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>

OpenCV

- 2.4.10
- Minimum API 8 (Android 2.2)
Let's Practice