User Interaction: Intro to Android

Assoc. Professor Donald J. Patterson
INF 133 Fall 2013
Checking out the phone

- Take a look at the sensors

http://www.google.com/support/android/bin/topic.py?hl=en&topic=28930
Making the phone work

http://www.google.com/support/android/bin/topic.py?hl=en&topic=28930
Making the phone work

- Turn on developer mode
- "home"->"menu"->"settings"->"applications" -> "Development"
- "USB debugging" on
- "Stay awake" on
- "Allow mock locations" on
- Dial *#*#CHECKIN#*#* to update phone software

http://www.google.com/support/android/bin/topic.py?hl=en&topic=28930
How to handle the assignment

• Stage 1
  • get your environment working with an emulator
How to handle the assignment

- Stage 2
- get your environment working with a real phone
How to handle the assignment

- Stage 3
- get your environment working on a real phone with sensors
How to handle the assignment

- Start a new Android Application project for Android API 10
- Give your application permissions in AndroidManifest.xml
  - Add a “Uses Permission”
    - to use the Internet
      - android.permission.INTERNET
    - to use location
      - android.permission.ACCESS_FINE_LOCATION
      - android.permission.ACCESS_COARSE_LOCATION
- Make your application debuggable

http://developer.android.com/guide/developing/device.html
How to handle the assignment

Android Manifest Permissions

- android.permission.INTERNET (Uses Permission)
- android.permission.ACCESS_COARSE_LOCATION (Uses Permission)
- android.permission.ACCESS_FINE_LOCATION (Uses Permission)
How to handle the assignment

**Application Attributes**

<table>
<thead>
<tr>
<th>Name</th>
<th>Browse...</th>
<th>Debuggable</th>
<th>true</th>
</tr>
</thead>
<tbody>
<tr>
<td>Theme</td>
<td>Browse...</td>
<td>Vm safe mode</td>
<td>Browse...</td>
</tr>
<tr>
<td>Label</td>
<td>@string/sensorsimulatorsettings</td>
<td>Manage space activity</td>
<td>Browse...</td>
</tr>
<tr>
<td>Icon</td>
<td>@drawable/mobileshake_application0</td>
<td>Allow clear user data</td>
<td>Browse...</td>
</tr>
<tr>
<td>Description</td>
<td>Browse...</td>
<td>Test only</td>
<td>Browse...</td>
</tr>
<tr>
<td>Permission</td>
<td></td>
<td>Backup agent</td>
<td>Browse...</td>
</tr>
<tr>
<td>Process</td>
<td></td>
<td>Allow backup</td>
<td>Browse...</td>
</tr>
<tr>
<td>Task affinity</td>
<td></td>
<td>Kill after restore</td>
<td>Browse...</td>
</tr>
<tr>
<td>Allow task reparenting</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Has code</td>
<td></td>
<td>Restore needs application</td>
<td></td>
</tr>
<tr>
<td>Persistent</td>
<td></td>
<td>Restore any version</td>
<td></td>
</tr>
<tr>
<td>Enabled</td>
<td></td>
<td>Never encrypt</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Can save state</td>
<td></td>
</tr>
</tbody>
</table>

**Application Nodes**

- `<provider>` SensorSimulatorProvider
- `<activity>` SensorSimulatorSettingsActivity
How to handle the assignment

- **High-Level**
  - You are going to ask Android to give you information about the phone’s orientation
  - You are going to do something in response to the information (with U/I and audio)

http://developer.android.com/guide/developing/device.html
How to handle the assignment

- **The Main Problem**
  - Information from the phone’s sensors are going to arrive much much faster than the phone can redraw the U/I
  - If you don’t manage this, your application will crash while it backs up waiting for you U/I to draw
  - Let’s do it the wrong but easy to understand way first

http://developer.android.com/guide/developing/device.html
How to handle the assignment

- Step 1: Create a place in the U/I to show the sensor data
  - The U/I object is a static class named “R”
- Step 2: Access the Android Sensor Service
- Step 3: Create a SensorEventListener that will handle the asynchronous callbacks
- Step 4: Tell the phone you are ready to get sensor readings
- Step 5: Tell the phone you don’t want sensor readings any more
How to handle the assignment

- Step 1: Create a place in the U/I to show the sensor data
  - The U/I object is a static class named “R”

- Step 2: Access the Android Sensor Service

- Step 3: Create a SensorEventListener that will handle the asynchronous callbacks

- Step 4: Tell the phone you are ready to get sensor readings

- Step 5: Tell the phone you don’t want sensor readings any more
How to handle the assignment

- Step 1: Create a place in the U/I to show the sensor data
How to handle the assignment

• Step 1: Create a place in the U/I to show the sensor data
  • The U/I object is a static class named “R”
• Step 2: Access the Android Sensor Service
• Step 3: Create a SensorEventListener that will handle the asynchronous callbacks
• Step 4: Tell the phone you are ready to get sensor readings
• Step 5: Tell the phone you don’t want sensor readings any more
How to handle the assignment

- Step 1: Create a place in the U/I to show the sensor data
  - The U/I object is a static class named “R”
- Step 2: Access the Android Sensor Service
- Step 3: Create a SensorEventListener that will handle the asynchronous callbacks
- Step 4: Tell the phone you are ready to get sensor readings
- Step 5: Tell the phone you don’t want sensor readings any more
How to handle the assignment

```java
mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
```

- **Observer**
  - `+notify()`

- **Subject**
  - `+observerCollection`
  - `+registerObserver(observer)`
  - `+unregisterObserver(observer)`
  - `+notifyObservers()`

- **ConcreteObserverA**
  - `+notify()`

- **ConcreteObserverB**
  - `+notify()`

`notifyObservers()` for `observer in observerCollection` call `observer.notify()`
How to handle the assignment

- Step 1: Create a place in the U/I to show the sensor data
  - The U/I object is a static class named “R”
- Step 2: Access the Android Sensor Service
- Step 3: Create a SensorEventListener that will handle the asynchronous callbacks
- Step 4: Tell the phone you are ready to get sensor readings
- Step 5: Tell the phone you don’t want sensor readings any more
How to handle the assignment

- Step 1: Create a place in the U/I to show the sensor data
- The U/I object is a static class named “R”
- Step 2: Access the Android Sensor Service
- Step 3: Create a SensorEventListener that will handle the asynchronous callbacks
- Step 4: Tell the phone you are ready to get sensor readings
- Step 5: Tell the phone you don’t want sensor readings any more
mEventListnerLight = new SensorEventListener() {

    @Override
    public void onSensorChanged(SensorEvent event) {
        float[] values = event.values;
        lastLightValue = values[0];
        updateUI();
    }

    @Override
    public void onAccuracyChanged(Sensor arg0, int arg1) {
    }
};
How to handle the assignment

- Step 1: Create a place in the U/I to show the sensor data
  - The U/I object is a static class named “R”
- Step 2: Access the Android Sensor Service
- Step 3: Create a SensorEventListener that will handle the asynchronous callbacks
- Step 4: Tell the phone you are ready to get sensor readings
- Step 5: Tell the phone you don’t want sensor readings any more
How to handle the assignment

- Step 1: Create a place in the U/I to show the sensor data
  - The U/I object is a static class named “R”
- Step 2: Access the Android Sensor Service
- Step 3: Create a SensorEventListener that will handle the asynchronous callbacks
  - Step 4: Tell the phone you are ready to get sensor readings
- Step 5: Tell the phone you don’t want sensor readings any more
How to handle the assignment

```java
mSensorManager.registerListener(mEventListenerLight, mSensorManager.getDefaultSensor(Sensor.TYPE_LIGHT), SensorManager.SENSOR_DELAY_FASTEST);
```
How to handle the assignment

- Step 1: Create a place in the U/I to show the sensor data
  - The U/I object is a static class named “R”
- Step 2: Access the Android Sensor Service
- Step 3: Create a SensorEventListener that will handle the asynchronous callbacks
- Step 4: Tell the phone you are ready to get sensor readings
- Step 5: Tell the phone you don’t want sensor readings any more
How to handle the assignment

• Step 1: Create a place in the U/I to show the sensor data
  • The U/I object is a static class named “R”

• Step 2: Access the Android Sensor Service

• Step 3: Create a SensorEventListener that will handle the asynchronous callbacks

• Step 4: Tell the phone you are ready to get sensor readings

• Step 5: Tell the phone you don’t want sensor readings any more
How to handle the assignment
What it looks like when it’s working

- Demo
the UI thread
public class MainActivity extends Activity {
    private TextView mTextViewLight;
    private SensorManager mSensorManager;
    private SensorEventListener mEventListenerLight;
    protected float lastLightValue;

    private void updateUI() {
        runOnUiThread(new Runnable(){
            @Override
            public void run() {
                mTextViewLight.setText("Light is "+lastLightValue);
            }
        });
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mTextViewLight = (TextView) findViewById(R.id.editText1);
        mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
        mEventListenerLight = new SensorEventListener()
        {
            @Override
            public void onSensorChanged(SensorEvent event){
                float[] values = event.values;
                lastLightValue = values[0];
                updateUI();
            }

            @Override
            public void onAccuracyChanged(Sensor arg0, int arg1) {
            }
        };
    }
}
@Override
public void onResume() {
    super.onResume();
    mSensorManager.registerListener(mEventListenerLight,
        mSensorManager.getDefaultSensor(Sensor.TYPE_LIGHT),
        SensorManager.SENSOR_DELAY_FASTEST);
}

@Override
public void onStop() {

    mSensorManager.unregisterListener(mEventListenerLight);
    super.onStop();
}
Hints

• Playing a sound
• The key is the MediaPlayer call
• Do not instantiate more than one MediaPlayer object

```java
static MediaPlayer mp;
AssetFileDescriptor afd;

//synchronized so that each call of playAudio is completed before another begins.
synchronized void playAudio(AssetFileDescriptor afd){
    if(mp.isPlaying()){
        return;
    }
    mp.reset();
    try{
        mp.setDataSource(afd.getFileDescriptor(), afd.getStartOffset(), afd.getLength());
        mp.prepare();
    }
    catch(Exception e){
        Log.d("playAudio", "Exception: "+e.getStackTrace()[0].toString()+" afd: "+afd.toString());
    }
    mp.start();
}

mp = new MediaPlayer();
afd = getApplicationContext().getResources().openRawResourceFd(R.raw.spin1);
```

• http://developer.android.com/guide/topics/media/index.html