CS297 Report

Accelerometer based motion gestures for Mobile Devices

Neel Parikh
neelkparikh@yahoo.com

Advisor: Dr. Chris Pollett

Department of Computer Science

San Jose State University

Spring 2008

Table of Contents

1. Introduction

Many smart phones (for example, iPhone from Apple) today use tiny sensors called accelerometers to provide enhanced user interface controls. Accelerometers measure the linear acceleration in the x,y,z directions and are often used for such things as controlling orientation of the display screen. The goal of this project is to extend the WebKit browser interface of Google's Android smart phone SDK so that accelerometer input can be used to function like a motion-sensing mouse. For instance, moving a phone up or down, left or right will correspond to scrolling in certain contexts. Another example could be to make the font size larger or smaller in the browser with our extension by moving the device closer or farther to the user. The Android platform is a software stack for mobile devices including an operating system, middleware and key applications. The Applications for Android platform are written using the Java programming language and run on Dalvik Virtual Machine (DVM), a custom virtual machine designed for use on embedded systems.

This report gives a detailed description about the research work and testing carried out on the Android Platform in the form of deliverables. Each deliverables involves experimenting with the Android SDK and developing independent applications which can be deployed and tested on the Android emulator. Section two talks about the tools used for the project and also gives a brief description about the basic building blocks of an Android application. This is followed by a detailed description of deliverables in terms of the objective, procedure, code snippets and the screenshot which assist in creating a better understanding of the project.

2. Tools used and basic structure of Android application.

In order to create an android application we need two basic things, namely: an integrated development environment (IDE) and the Android software development kit. Since the actual device is not available in market, I am using a standalone JAR application called SensorSimulator by OpenIntents project for simulating acceleration and sending inputs to the emulator.

2.1 Eclipse IDE:

In this project I will make use of Eclipse, an IDE written primarily in Java. In eclipse there is a provision for installing a custom plug-in called Android Development Tools (ADT) which supports creating, running, and debugging Android applications.

2.2 Android SDK:

It is a software development kit from Google which includes development tools, an emulator, and the libraries needed for building an Android application.

2.3 Basic structure of an Android Application

An Android application has four basic building blocks. They are:

2.3.1 Activity:

An activity usually represents a single screen in an application. Navigating from one screen to another is accomplished by starting a new Activity.

2.3.2 Intent and Intent Filters:

In Android 'Intent' is used to navigate from one screen to another. It consists of 'action' and the 'data' on which the action has to be performed. Intent Filter gives the description about the kind of Intents an Activity can handle. The Intent Filters are described in the 'AndroidManifest.xml' file.

2.3.3 Intent Receiver:

It is used when we want the code in our application to execute in reaction to an external event.

2.3.4 Service:

It is an application component which runs in the background and does not interact with the user for an indefinite period of time.

2.3.5 Content Provider:

It is used when an application wants to share its data stored in files, a SQLite database, with some other application.

3. Deliverable 1 – Hello world application.

3.1 Objective: The objective of Deliverable 1 was to test a simple Hello World program for the Android Platform. The output of this deliverable was an independent application which can be seen on the emulator screen (Fig 1). Once the user clicks on this application it will display a message to the user (Fig 2).

Fig 1. Hello World application on home screen of emulator

Fig 2. Hello World running on the emulator

3.2 Constructing UI: The initial task was to construct the UI needed for displaying the message to the user. This involved using a simple TextView to display a custom message. Given below is the code snippet for this step.

```
TextView tv = new TextView(this);

    /**
      Sets the string value of the TextView
    */
    tv.setText("Hello people, welcome to the world of Android");

    /**
      Set the activity content to an explicit view
      @param tv The desired content to display
    */
    setContentView(tv);
```

Here the UI was constructed directly in the source code. Another approach would be to create a XML file based layout. In this one can create all the UI elements like text boxes, buttons etc. and refer to each of these elements with the help of unique ids assigned to them in the main source code.

4. Deliverable 2 – Connecting Sensor Simulator to Android Emulator.

4.1 Objective: The objective of Deliverable 2 was to integrate the Sensor Simulator (Fig 3) with Android Emulator. After integration the simulator values would be reflected on the emulator screen (Fig 4).

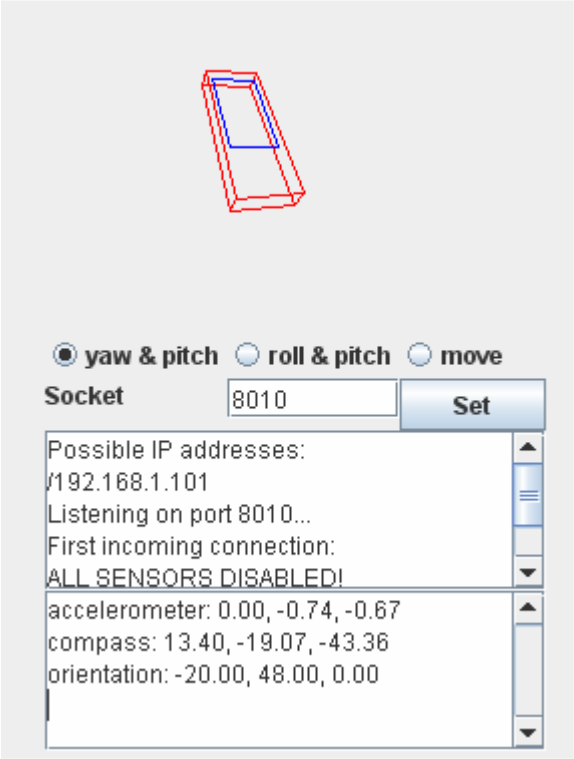

Fig 3. Sensor values of Accelerometer generated by moving the Simulator phone

The above figure shows the OpenIntents Sensor Simulator. This is a standalone jar application. It lets you simulate sensor data with the mouse in real time.

4.2 Connection: Connecting the Sensor Simulator to the Android emulator as follows -

1. First download the openintents zip package to the local machine.
2. Then start the simulator from the tools > SensorSimulator.jar (Java standalone application as shown in Fig 3).
3. Install the OpenIntents.apk (application package) on the Android emulator from the command prompt.
4. Launch the OpenIntents on the emulator and select the SensorSimulator from the options.
5. This pops up a UI where you can enter the IP address and socket number as shown in the Sensor Simulator (Fig 3.)

6. Then go to the testing tab and select ‘Connect’ button on the emulator. The first time you connect to the SensorSimulator, all sensors are disabled automatically. This is, because it is the Android application's responsibility to enable the sensors before reading values.
7. Now you can see the sensor data (Fig 4.) on the emulator screen with a small delay. The simulator data and the emulator data are in sync with each other.

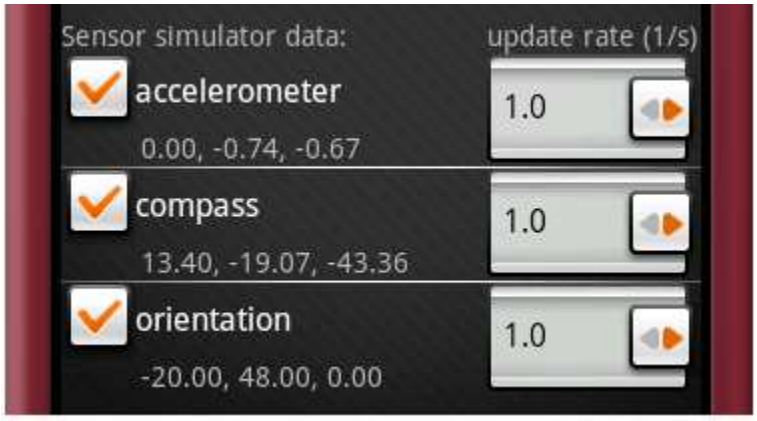

Fig 4. Sensor values on Android Emulator match the simulation values of the Sensor Simulator (Fig 3).

If you move the SensorSimulator phone with the mouse, the numbers will change in the SensorSimulator and correspondingly change on the Android emulator. These values can then be used in any application on the emulator.

4.3 Input file functionality: The other way I was able to change the sensor values of Accelerometer was by making a change in the gravity settings of the Sensor Simulator as show in the below figure (Fig 5).

Fig 5. Gravity settings of Sensor Simulator

As seen in the above figure, changing the values of the text boxes of x,y,z caused the change in the motion of the simulator phone, which led to change in the sensor values of Accelerometer. This was a manual process and hence required me to change the values every time to generate a small change in the sensor values. Hence, In order to overcome this manual work I made an addition functionality to the standalone jar application to automate the input values as shown in the below figure (fig 6).

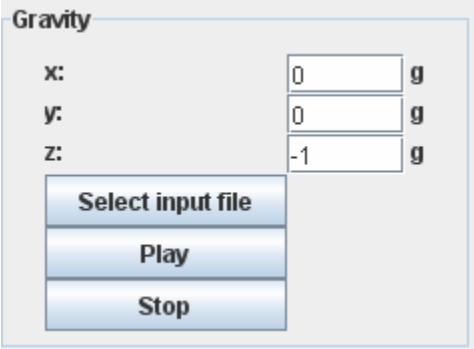

Fig 6. Additional functionality of playing the values from a file.

As seen in the figure, I added three buttons which helped me to select a specific input file stored anywhere on the local machine. Given below is the code snippet for this file choosing functionality.

```
else if (action.equals("Select input file")) {
    fc = new JFileChooser();
/**
 * Pops up an "Open File" file chooser dialog
 */
    int returnVal = fc.showOpenDialog(SensorSimulator.this);

   if (returnVal == JFileChooser.APPROVE_OPTION) {

            file_name = fc.getSelectedFile().getPath();
            choose_file.setToolTipText(file_name);
     }
```

This file had a list of values which I could pass to x,y,z Text Boxes in Gravity settings every two seconds. Once I select the input file and click 'Play', this helps me automate the input values of x,y,z and thereby avoid manual settings. I used a thread for this purpose in my java code which would do a file read and pass a new value to x,y,z every two seconds. These values would then be transferred to the emulator as shown in Fig 4.

5. Deliverable 3 – Generating a simple event.

5.1 Objective: The objective here was to generate a simple event using Accelerometer. The output would be to show an alert message on the screen during the fall of the phone (Fig 7) and as the values are changing. And finally when the phone strikes the ground another alert message pops up (Fig 8).

Fig 7. The UI screen with accelerometer values for [x,y,z] and alert message during the fall of phone.

Fig 8. The UI screen with accelerometer values and alert message when the phone has reached ground.

5.2 Description of the event: Initially we will assume that the phone is at rest on a surface. Hence the initial values of [x,y,z] will be [0,0,-1]. Then we will simulate an event that the phone is thrown and is falling down and gravitational force is acting on it. During the fall the [x,y,z] values will change rapidly to cause a change in the accelerometer values. When the phone is falling it will display an alert message. When the phone strikes the ground, the [x,y,z] values will come to rest and so will the accelerometer values. Again an alert message will be generated indicating that the phone has striked the ground. The values generated have also been written to a log file to help in debugging the code.

5.3 Description of code and files: The entire coding of this deliverable had been split up into four files.

5.3.1 Test.java: It contains the class which is initially executed and which connects to Sensor Simulator. The functionality for capturing and displaying the [x,y,z] values is coded in this file. Given below is the snapshot of displaying a value on screen.

```
/**
 * Display values of [x,y,z] on the screen
 */
   if(j==1){
        TextView temp1 = (TextView)findViewById(R.id.one);

temp1.setText("[x,y,z] = ["+ String.valueOf(String.format("%.2f",d1)) + ","

+String.valueOf(String.format("%.2f",d2))","

+String.valueOf(String.format("%.2f",d3))+"]");
            }
```

5.3.2 AccelerometerReader.java: This class is called from test.java and it basically enables the sensors on the phone. It also tests for a particular sensor (Accelerometer in our case) and enables it. The accelerometer values corresponding to [x,y,z] are generated in this class and the result is passed to the Test.java class where they are displayed. Given below is the snapshot of the reading the values from accelerometer

```
int sensorValues =
                    Sensor.getNumSensorValues(Sensors.SENSOR_ACCELEROMETER);

                    float[] out = new float[sensorValues];

                   Sensors.readSensor(Sensors.SENSOR_ACCELEROMETER, out);

                   android.util.Log.i("FIVE","in readAccelerometer");
                   return out;
```

5.3.3 main.xml: It is the UI file which shows the button "throw the phone"

5.3.4 main2.xml: It is the UI file which defines the TextViews to show the rapid changing values of accelerometer corresponding to [x,y,z] on screen. Also the alert messages are displayed on this screen.

For debugging purposed I wrote each value to a log file. Given below is a snapshot of the values being written to the log file.

```
05-13 04:55... I 676   Answer          7.348256587982178
05-13 04:55... I 676   Answer          -3.872981309890747
05-13 04:55... I 676   Answer          -5.291799545288086
05-13 04:55... I 676   IF              2nd value
05-13 04:55... I 676   FIVE            in readAccelerometer
05-13 04:55... I 676   FIVE            in readAccelerometer
05-13 04:55... I 676   Answer          6.919868469238281
05-13 04:55... I 676   Answer          6.200979232788086
05-13 04:55... I 676   Answer          5.259589195251465
05-13 04:55... I 676   FIVE            in readAccelerometer
05-13 04:55... I 676   FIVE            in readAccelerometer
05-13 04:55... I 676   Answer          6.919868469238281
05-13 04:55... I 676   Answer          6.200979232788086
05-13 04:55... I 676   Answer          5.259589195251465
05-13 04:55... I 676   FIVE            in readAccelerometer
05-13 04:55... I 676   FIVE            in readAccelerometer
05-13 04:55... I 676   Answer          0.0
05-13 04:55... I 676   Answer          -8.66025447845459
05-13 04:55... I 676   Answer          -5.0
```

Fig 9. Values being written to the log file when the code is executing.

6. Deliverable 4 – Implementing Shake feature.

6.1 Objective: The objective here was to implement the Shake feature based on Accelerometer. The user would enter some text as input (Fig 10) and on shaking the Sensor Simulator the Text Box entry would be cleared. (Fig 11).

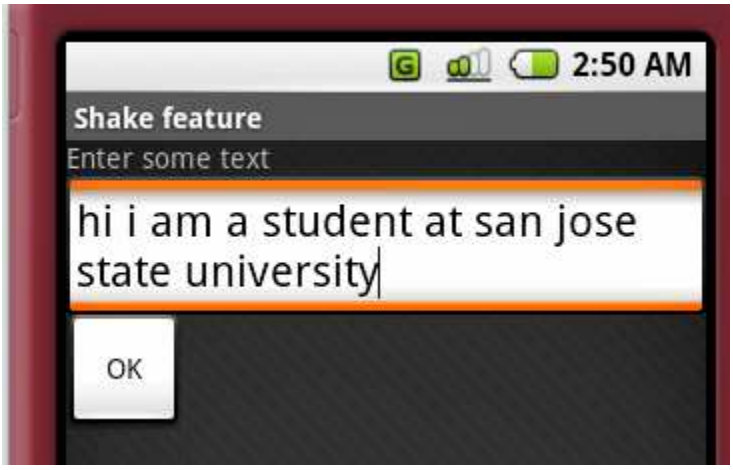

Fig 10. The UI screen with Text Box and connection button.

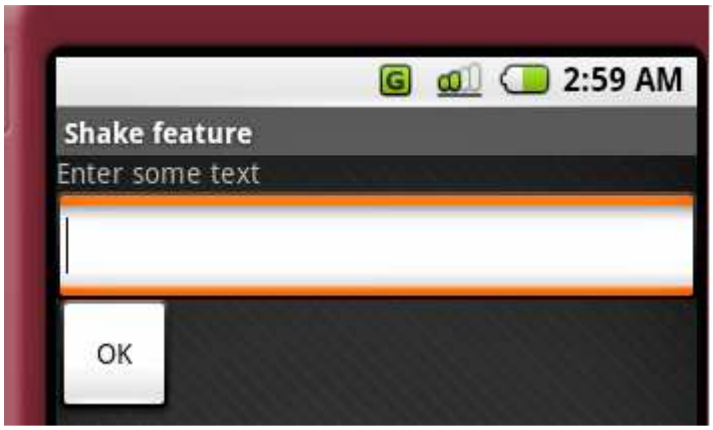

Fig 11. The cleared Text Box due to shake motion of Sensor Simulator

6.2 Description of the event: The basic principle of shake feature is that if a user wants to delete something that he previously entered then he can do it by shaking the phone. He does not have to explicitly use a 'back' or a 'clear' key. The implementation of this feature consisted of a Textbox where user can enter any data. The user makes a connection to the Sensor Simulator jar application. When the user shakes the graphical image of the phone in the sensor simulator, (Fig 12) the data in the Textbox is cleared. When the phone is shaken the accelerometer values generated are tested against a threshold value. If this generated value is greater than the threshold then it is detected as a 'shake' motion. The values generated are written to a log file to help in debugging the code.

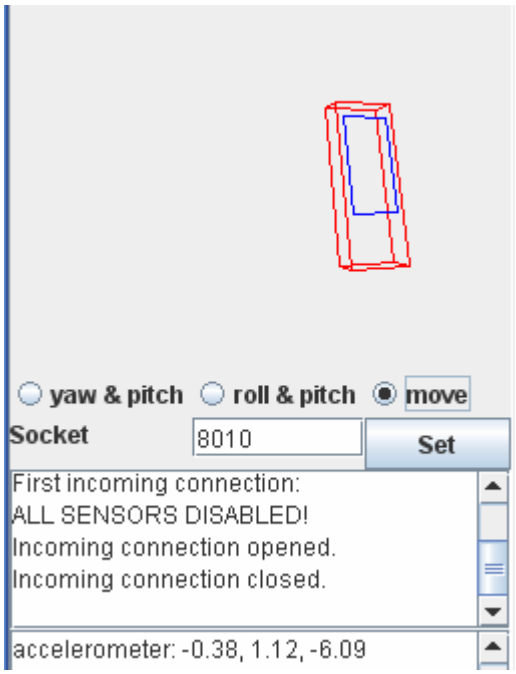

Fig 12. Shaking of Sensor Simulator phone

6.3 Description of code and files: The entire coding of this deliverable had been split up into three files.

6.3.1 Test.java: This class connects to the Sensor Simulator jar application. It is the class which is initially executed and which connects to Sensor Simulator. The functionality for detecting the shake motion and capturing the [x,y,z] values during shaking of the phone is coded in this file. These values are also written to the log file.

6.3.2 AccelerometerReader.java: This class is called from test.java and it basically enables the sensors on the phone. It also tests for a particular sensor (Accelerometer in our case) and enables it. The accelerometer values corresponding to [x,y,z] are generated in this class and the result is passed to the Test.java class where they are displayed. Given below is the snapshot of the reading the values from accelerometer.

6.3.3 main.xml: It is the UI file which shows the Text box and Sensor Simulator connection button.

The values generated by shaking the phone have been written to the Log file. Given below is a snapshot of the log file where the message of 'Shake detected' can be seen.

```
Hardware        Read line...
Hardware        Received: SensorSimulator
Hardware        Connected
THREE           connected to simulator
THREE           start initialization
FOUR            in constructor
Hardware        getSupportedSensors()
Hardware        Received: 3
Hardware        Received: accelerometer
Hardware        Received: compass
Hardware        Received: orientation
FIVE            enable sensor
SIX             in setEnableAccelerometer
Hardware2       enableSensor()
Hardware2       Send: accelerometer
Hardware2       Received: false
SEVEN           end setEnableAccelerometer
EIGHT           end constructor
NINE            completed initialization
FIVE            in readAccelerometer
FIVE            in readAccelerometer
Answer          -2.236241579055786
Answer          -1.5053974390029907
Answer          -1.8618059158325195
LEN is          11247038
SHAKE           shake detected
FIVE            in readAccelerometer
FIVE            in readAccelerometer
Answer          -0.2529144585132599
Answer          -0.8470873832702637
Answer          -0.6042823195457458
LEN is          1200177
```

Fig 13. The values being written to log file

8. Future Work

For the next semester I intend to create motion based features like scrolling a web page with motion detected by accelerometer. Another feature would be to increase or decrease the fonts or the zooming in and out based on motion of phone. If time permits, I also intent to experiment with sensors like the 'Compass' and 'Orientation' which are supported by the Android platform. It would be very nice to test the applications if an actual device or prototype is available within the time constraints.

9. Conclusion

The deliverables and experimental coding applications done during this semester have greatly increased my understanding of the working of Android platform. These deliverables have laid the fundamental groundwork needed for building concrete applications for CS298.

10. References

[1] Dennis Majoe, SQUEAK: A Mobile Multi Platform Phone and Networks Gesture Sensor, Proceedings of the 2007 IEEE 2nd International Conference on Pervasive Computing and Applications.

[2] [2005] Ian Okley. Tilt to Scroll: Evaluating a Motion Based Vibrotactile Mobile Interface, Proceedings of the First Joint Eurohaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems.

[3] Google's Android project.http://code.google.com/android/index.html

[4] Open Intents project. http://code.google.com/p/openintents/

[5] Android developer forums. http://www.anddev.org/