Using the Accelerometer for cool apps in JavaME

Note: This is an entry in the Asha Touch Competition 2012Q3

This article explains how to properly use the accelerometer for effects and user interaction for apps.

Introduction

With the new Asha Full Touch devices coming with JSR-256 and an accelerometer support you can explore new ways to use that feature for cool effects or user-interaction for you apps. In this Example i will show how to create a dice app that will start to roll once the phone is shaked. You can check out the full app also on the Nokia store.

App main

After shaking

The code

The code is split into 5 classes

- `Dice` - a domain class this represent a single die, it has the x,y position as well as the current state of the die (moving,rolling) it also has all the strips of the die sides.
- `DirectionSensor` - this utility class is responsible to listen to the accelerometer x,y,z axis and calculate the difference between the current and former x,y,z state and if the difference is large enough it send a message to `ShakeListener`
- `Main` - The MIDLet class
- `MyCanvas` - This class inherits the `GameCanvas` class and calls the `paint()` method of dice, its also responsible for the collision detection of the dice.
- `Rect` - This is a simple utility class to do collision test between 2 rects.

I'll post below parts of the code for the `DirectionSensor` class and `Main` class which both relates to the topic of this article which is the use of the accelerometer, for the full source code of all the class you can downloaded the attached source project.

`DirectionSensor` class - 3 important methods. `dataReceived(SensorConnection con, Data[] dataArray, boolean isDataLost)` - this method is needed in order to implements the `DataListener` of `SensorConnection` it calls `getIntegerDirection` to convert the raw buffer data to x,y,z values it can work it. It also compare the current values magnitude with the previous values magnitude and detects if a shake has occurred.

```
public void dataReceived(SensorConnection con, Data[] dataArray, boolean isDataLost)
{
 if (dataArray == null) {
  return;
 }
 if (dataArray.length < 3) {
  return;
 }
 double temp[] = getIntegerDirections(dataArray);
 if (!receivedFirstData)
 {
  receivedFirstData = true;
  acceleration[0] = (float) temp[1];
                     acceleration[1] = (float) temp[0];
                                 acceleration[2] = (float) temp[2];
  return;
 }
 else
 {
  if (Math.abs(vectorMagnitude(new float[]{(float) temp[1],(float) temp[0],(float) temp[2]})-vectorMagnitude(acceleration))>DIRECTIO
  {
```


(C) Copyright Nokia 2013. All rights reserved.

```
    acceleration[0] = (float) temp[1];
    acceleration[1] = (float) temp[0];
    acceleration[2] = (float) temp[2];
    listener.shakeStarted();
    return;
   }
   else
   {
    acceleration[0] = (float) temp[1];
    acceleration[1] = (float) temp[0];
    acceleration[2] = (float) temp[2];
    return;
   }
  }
 }
```

`vectorMagnitude(float[] arr)` - this is a helper function to calculate a 3d vector's magnitude - the x,y,z values of the accelerometer can be think of as 3d vector and you can calculate its magnitude and compare it to another 3d vector magnitude and if the difference between the two magnitudes are bigger than delta then the movment was big.

```
private static float vectorMagnitude(float[] arr)
{
 return (float) Math.abs(Math.sqrt(arr[0]*arr[0]+arr[1]*arr[1]+arr[2]*arr[2]));
}
```

`double[] getIntegerDirections(Data[] data)` - this function converts the raw Data buffer to x, y, z float values to work with.

```
private static double[] getIntegerDirections(Data[] data)
{
 double [][] intValues = new double[3][BUFFER_SIZE];
 double[] directions = new double[3];
 for (int i=0; i<3; i++){
  intValues[i] = data[i].getDoubleValues();
  double temp = 0;
  for (int j = 0; j<BUFFER_SIZE; j++) {
   temp = temp + intValues[i][j];
  }
  directions[i] = temp/BUFFER_SIZE;
 }
 return directions;
}
```

`Main` class - its important code is inside its startApp method, its main purpose is to create and find the proper sensor and its connection for use. Note that we `SensorManager.findSensors("acceleration", null)` can return several `SensorInfo` so you need to make sure you get the right one. In the example below we search for an acceleration sensor that support the reporting of Double values as its data type.

```
protected void startApp() throws MIDletStateChangeException
{
 try
 {
  Display.getDisplay(this).setCurrent(canvas);
  canvas.startRendering();
  sensor = null;
  SensorInfo infos[];
  infos = SensorManager.findSensors("acceleration", null);
  int datatypes[] = new int[infos.length];
  int i = 0;
  String sensor_url = "";
  boolean sensor_found = false;
  while (!sensor_found) {
   datatypes[i] = infos[i].getChannelInfos()[0].getDataType();
   if (datatypes[i] == ChannelInfo.TYPE_DOUBLE) {
    sensor_url = infos[i].getUrl();
    sensor_found = true;
   }
   else i++;
  }
  sensor = new DirectionSensor((SensorConnection) Connector.open(sensor_url));
  sensor.activate(this);

 } catch (Exception e)
 {
  e.printStackTrace();
  notifyDestroyed();
 }
}
```

As you can see in the `Main startApp()` method we ask the `SensorManager` for an accelerometer sensor and then we traverse the returned array and search for a sensor that supports channel type `Double` (sensor can support different channel types which is basically the way the data is transeferred back to you Double/Int).

In the `DirectionSensor` class the two main important methods are `dataReceived(SensorConnection con, Data[] dataArray, boolean isDataLost)` and `getIntegerDirections(Data[] data)` in the first method we call the second method to interpret the raw values from the sensor and then we calculate the different between its current and its previous state (this is done with a simple math calulation of the magnitude of the x,y,z vector)

Summary

I hope you download the binary and test out the app yourself and then check out the code to learn how to make more cool apps like this.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.