How to get accelerometer sensor values in Java ME

Overview

24 May 2009

S60 5th Edition devices support Mobile Sensor API (JSR-256). Nokia 5800 XM and Nokia N97 are the first devices having it as an built-in API. The S60 5th Edition (and most of the S60 3rd Edition FP1 and the S60 3rd Edition FP2) devices has an accelerometer, which gives values based on the device position and movement. For example, when the device screen is turned to landscape position, also the screen is updated to landscape position.

As shown in the How to get information about sensors in Java ME, in Nokia N97 Mobile Sensor API finds two different accelerator sensors, one giving integer values, another giving double values. In this article it is shown, how to get values from these accelerators in a MIDlet.

Steps in using the accelerator is as follows:

- find a sensor of type "acceleration" in the device
- select the sensor giving integer or double values
- get the URL to the sensor
- open SensorConnection to the sensor
- implement the dataReceived() method for using the sensor data

Code samples

The code sample below shows, how to search the "acceleration" type of sensor and how the sensor value type can be selected. The method also gets the sensor URL and returns the correct SensorConnection to it.

```
    /**
     * Searches sensors of "accelerator" quantity and if found returns a
     * SensorConnection opened to it. Based on the boolean value of type_int
     * either TYPE_DOUBLE or TYPE_INT sensor is used.
     * @return SensorConnection, which has been opened to a sensor matching the criteria
     */
    private SensorConnection openSensor() {
  infos = SensorManager.findSensors("acceleration", null);
  if (infos.length==0) return null;
        int datatypes[] = new int[infos.length];
        int i = 0;
        String sensor_url = "";
        if (!type_int) {
            System.out.println("Searching TYPE_DOUBLE sensor...");
            while (!sensor_found) {
                datatypes[i] = infos[i].getChannelInfos()[0].getDataType();
                if (datatypes[i] == 1) { //ChannelType.TYPE_DOUBLE = 1
                    sensor_url = infos[i].getUrl();
                    System.out.println("Sensor: " + sensor_url + ": TYPE_DOUBLE found.");
                    sensor_found = true;
                }
                else i++;
            }
        }
        else if (type_int) {
            System.out.println("Searching TYPE_INT sensor...");
            while (!sensor_found) {
                datatypes[i] = infos[i].getChannelInfos()[0].getDataType();
                if (datatypes[i] == 2) { //ChannelType.TYPE_INT = 2
                    sensor_url = infos[i].getUrl();
                    System.out.println("Sensor: " + sensor_url + ": TYPE_INT found.");
                    sensor_found = true;
                }
                else i++;
            }
        }
        System.out.println("Sensor: " + sensor_url);
  try {
     return (SensorConnection)Connector.open(sensor_url);
  }catch (IOException ioe) {
            ioe.printStackTrace();
     return null;
  }
    }
```

The following method shows, how to set DataListener for listening sensor values:

```
    /**
     * Initializes (opens) the sensor and sets the DataListener
     */
    private synchronized void initSensor() {
        sensor = openSensor();
     if (sensor == null) return;
     try {
         sensor.setDataListener(this, BUFFER_SIZE);
  while(!isStopped){
      try{
          wait();
      }catch(InterruptedException ie){}
  }
      sensor.removeDataListener();
  }catch (IllegalMonitorStateException imse) {
            imse.printStackTrace();
        }catch (IllegalArgumentException iae) {
            iae.printStackTrace();
        }
     try {
```


(C) Copyright Nokia 2013. All rights reserved.

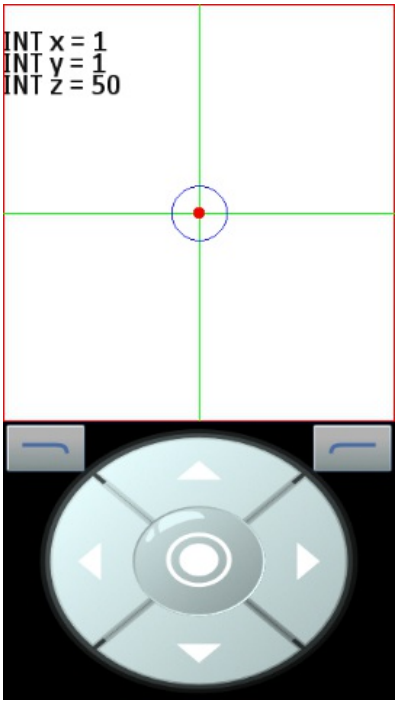
```
            sensor.close();
        } catch(IOException ioe){
            ioe.printStackTrace();
        }
            if (isStopped) {
                sensor = null;
            }
        }
```

Finally, the dataReceived() method is implemented for getting the sensor values. Also the getIntegerDirections() method is listed, it calculates the average values of the buffered sensor values.

```
        /**
         * Notification of the received sensor data.
         * @param sensor - SensorConnection, the origin of the received data
         * @param data - the received sensor data
         * @param isDataLost - true if some data has been lost
         */
        public void dataReceived(SensorConnection sensor, Data[] data, boolean isDataLost) {
            if (!type_int) {
                double[] directions = getDirections(data);
                x = directions[0];
                y = directions[1];
                z = directions[2];
            }
            else {
                int[] directions = getIntegerDirections(data);
                x_int = directions[0];
                y_int = directions[1];
                z_int = directions[2];
            }
            repaint();
        }

        /**
         * Gets the directions (axis_x, axis_y and axis_z values) from the accelerator
         * sensor data. An average value is calculated from the buffer values.
         * @param data The sensor data
         * @return directions The array containing the axis_x, axis_y and axis_z values
         */
    private static int[] getIntegerDirections(Data[] data) {
        int [][] intValues = new int[3][BUFFER_SIZE];
                int[] directions = new int[3];
     for (int i=0; i<3; i++){
         intValues[i] = data[i].getIntValues();
                        int temp = 0;
                        for (int j = 0; j<BUFFER_SIZE; j++) {
                            temp = temp + intValues[i][j];
                        }
                        directions[i] = temp/BUFFER_SIZE;
     }
        return directions;
    }
```

There are also the SensorTestMIDlet.jad and SensorTestMIDlet.jar files available here. The MIDlet shows the sensor values and draws a graphical indicator based on the values, as shown in the picture below.



An image showing the SensorTest MIDlet running in Nokia N97 SDK 0.5.

Example application

- SensorTestMIDlet.zip containing SensorTestMIDlet.jad, SensorTestMIDlet.jar and sources

See also

(C) Copyright Nokia 2013. All rights reserved.

- Mobile Sensor API (JSR-256) beta add-on for Nokia 5800 XpressMusic
- Nokia N97 SDK 0.5
- Mobile Sensor API (JSR-256) javadoc documentation and RI Binary
- Video of using Mobile Sensor API for controlling RC car in Nokia Developer Summit 2009 in Monaco
- How to get information about sensors in Java ME

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.