Como capturar os valores do acelerômetro, em Java ME

Introdução

Como mostrado no Como recuperar informações de sensores em Java ME, na API MOBILE SENSOR no Nokia N97 encontra-se dois diferentes sensores de acelerômetros, um dá valores inteiros, outros valores em ponto flutuante. Este artigo mostra, como capturar valores destes acelerômetros em um MIDlet.

Os dispositivos S60 quinta edição suportam a MOBILE SENSOR API (JSR-256). O Nokia 5800 XM e o Nokia N97 são os primeiros dispositivos a ter esta API implementada. Os dispositivos S60 quinta edição (e a maioria dos dispositivos S60 terceira edição FP1 e os da terceira edição FP2) tem um acelerômetro, que providencia valores baseados na posição e movimento do dispositivo. Por exemplo, quando a tela do aparelho móvel e mudada de posição, a tela é atualizada para a posição

Os passos para se usar o acelerômetro são os seguintes:

- Encontre um sensor do tipo acelerômetro no dispositivo.
- Selecione o sensor dando valores inteiros ou em ponto flutuantes
- Passe a URL para o sensor
- Abra a SensorConnection para o sensor
- Implemente o método *dataReceived()* para usar os dados do sensor

O simples código abaixo mostra como pesquisar o tipo do acelerômetro do sensor do acelerômetro e como o tipo do valor do sensor pode ser selecionado. O método também captura a URL do sensor e retorna a correta SensorConnection para ele

Simples código

O simples código abaixo mostra como pesquisar o tipo do acelerômetro do sensor do acelerômetro e como o tipo do valor do sensor pode ser selecionado. O método também captura a URL do sensor e retorna a correta SensorConnection para ele.

```
    /**
     * Procura por sensores de acelerômetro e caso encontrado abre uma conexão do tipo
     * SensorConnection. Baseado no valor booleano de type_int
     * sensores TYPE_DOUBLE ou TYPE_INT é utilizado.
     * @return SensorConnection, o qual foi aberto com o sensor
     */
    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;
  }
    }
```

O seguinte método mostra, como configurar o DataListener para capturar os valores do sensor:

```
    /**
     * Inicializa (abre) o sensor e adiciona um DataListener
     */
    private synchronized void initSensor() {
        sensor = openSensor();
     if (sensor == null) return;
     try {
         sensor.setDataListener(this, BUFFER_SIZE);
  while(!isStopped){
      try{
          wait();
```


(C) Copyright Nokia 2013. All rights reserved.

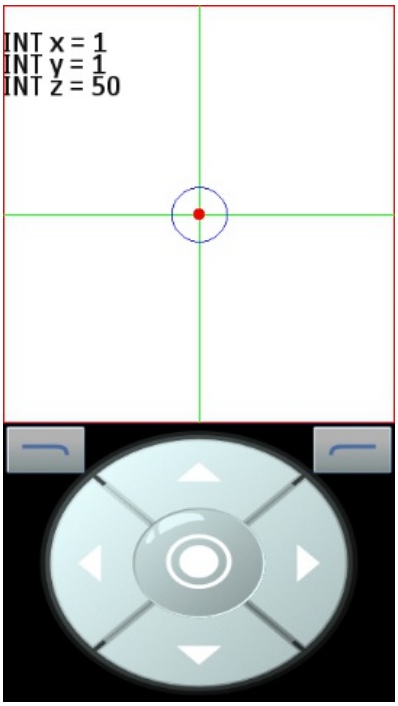
```
        }catch(InterruptedException ie){}
    }
        sensor.removeDataListener();
    }catch (IllegalMonitorStateException imse) {
            imse.printStackTrace();
          }catch (IllegalArgumentException iae) {
            iae.printStackTrace();
          }
       try {
           sensor.close();
       } catch(IOException ioe){
           ioe.printStackTrace();
       }
          if (isStopped) {
              sensor = null;
          }
      }
```

Finalmente, o método *dataReceived()* é implementado para dar os valores do sensor. O método *getIntegerDirections também* é listado, ele calcula a média dos valores armazenados.

```
       /**
        * Notificação dos dados recebidos pelo sensor
        * @param sensor - SensorConnection, o sensor de origem dos dados enviados
        * @param data - os dados recebidos
        * @param isDataLost - verdadeiro caso algum dado tenha sido perdido
        */
      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();
      }

      /**
       * Obtém a direção (axis_x, axis_y and axis_z values) do acelerômetro
       * Um valor médio é calculado do buffer de valores.
       * @param data As informações do sensor
       * @return directions Um array possuindo os valores axis_x, axis_y e axis_z
       */
  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;
  }
```

Os arquivos SensorTestMIDlet.jad e SensorTestMIDlet.jar estão disponíveis aqui. O MIDle mostra os valores do sensor e desenha um indicador gráfico baseado nesses valores, como mostrado na figura abaixo.



Uma imagem mostrando o SensorTest MIDlet sendo executado no Nokia N97 SDK 0.5.

Aplicação exemplo

(C) Copyright Nokia 2013. All rights reserved.

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

Veja também

- 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.