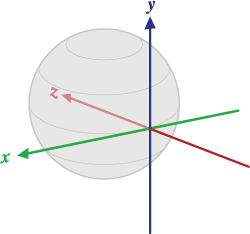
**Orientation Senor**

**Component Description:**

This is a class which is used to sense the device orientation. This class makes use of Sensor.TYPE\_ACCELEROMETER and Sensor.TYPE\_MAGNETIC\_FIELD to determine the device orientation.



Computes the device's orientation with respect to this

coordinate system.

It returns the result:

* *azimuth*, angle of rotation around the Z axis.
* *pitch*, angle of rotation around the X axis.
* *roll*, angle of rotation around the Y axis.

**Class** SensorData

Constants

1. final float angleAzimuth: angle of rotation around the Z axis

2. final float anglePitch: angle of rotation around the X axis

3. final float angleRoll: angle of rotation around the Y axis

**Interface:**

interface SensorListener{

public void onSensorChanged(SensorData objSensorData);

}

This interface needs to be implemented by a listener class. This method will be invoked by the OrientationSensor whenever the device orientation changes. objSensorData will contain angle data of the device orientation.

**Public methods:**

1. public void register()

Registers itself to sense the sensors data. It's needed to be called in activity's onResume() method.

2. . public void unregister()

Unregisters itself and frees sensors to make them available to other applications. It's needed to be called in activity's onPause () method.

3. public void addSensorListener(SensorListener objSensorListener)

Adds a listener to listen the orientation changes.

**Example usage:**

public class SensorActivity extends Activity implements OrientationSenor.SensorListener {

Context ctx;

OrientationSenor objOrientationSenor;

public SensorActivity() {

ctx = this;

objOrientationSenor=new OrientationSenor(ctx);

objOrientationSenor.addSensorListener(this);

}

protected void onResume() {

super.onResume();

try{

objOrientationSenor.register();

}

catch(Exception e){

Toast toast = Toast.makeText(ctx,

e.getMessage(), Toast.LENGTH\_SHORT);

toast.show();

}

}

protected void onPause() {

super.onPause();

objOrientationSenor.unregister();

}

@Override

public void onSensorChanged(OrientationSenor.SensorData objSensorData) {

// do whatever you want with sensor data(objSensorData).

// objSensorData.angleAzimuth : rotation around the Z axis

// objSensorData.anglePitch : rotation around the X axis

// objSensorData.angleRoll : rotation around the Y axis

}

}

**Configuration Instructions:**

AndroidManifest.xml requirements :

This component uses two features of android system. So the request for those features need to be added into the **Manifest** file.

<uses-feature

android:name=*"android.hardware.sensor.accelerometer"*

android:required=*"true"* />

<uses-feature

android:name=*"android.hardware.sensor.compass"*

android:required=*"true"* />

**Note:** The demo project OrientaionSensor is a library project. If you want to run the demo project, first you need to uncheck its library project option. You can use this component by either simply copying respective files in your project or adding the OrientaionSensor project as a library.

**File List with description for the file**

|  |  |  |
| --- | --- | --- |
| No. | File Name | Description |
| 1 | com.customc.orientaionsensor.OrientationSenor | This is a class which is used to sense the device orientation. This class makes use of Sensor.TYPE\_ACCELEROMETER and Sensor.TYPE\_MAGNETIC\_FIELD to determine the device orientation. |

**Contact information for the programmer:**

Yogesh Pangam.

**Known Bugs:**