|  |
| --- |
| Android Tutorial – Part 6 |

|  |
| --- |
| 6-24-2018 |



Table of Contents

[Introduction 2](#_Toc517722795)

[Android Motion Sensors 3](#_Toc517722796)

[The Rotational Vector Sensor (Sensor.TYPE\_ROTATION\_VECTOR) 3](#_Toc517722797)

[References 7](#_Toc517722798)

# Introduction

This is the part six of the android tutorial series. It is a continuation from last week. In order to follow this successfully, it is required to have,

* A basic understanding given about android in last session.
* The environment set up.
* The project created during last tutorial, opened in Android Studio.
* AVD or an Actual device ready for app deployment.

To catch up, in the last session (Android Tutorial Part 5),

* Navigation Drawer implementation
* Adding dependencies to app’s Gradle build file
* Adding drawer to an activity
* Adding a custom menu resource
* Adding a drawable resource to app
* Changing app themes
* Changing tool bar – adding a toggle button
* FrameLayout in Android
* Fragments in android
* Sensor Framework in Android
* Listing down the sensors available in device
* Using Accelerometer sensor in the app
* Vibrating the device through app

<https://github.com/nadee158/android_tutorial_part_5.git>

With that knowledge in hand, in this session below areas will be covered,

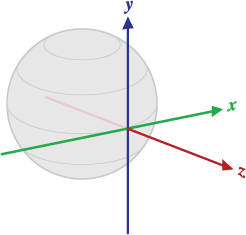
# Android Motion Sensors

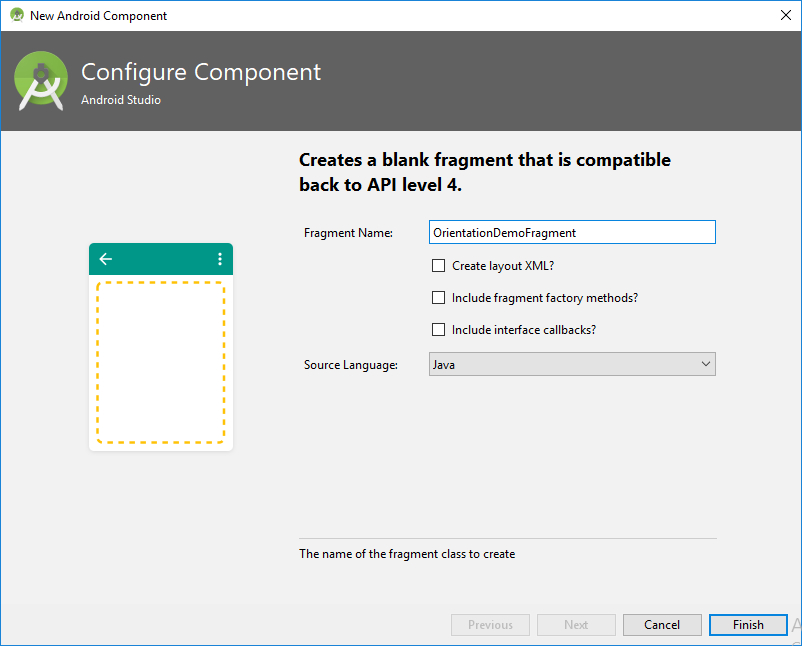
In the previous tutorial, we looked in to the sensor framework in Android. Also a demonstration was done on the Accelerometer Sensor. In the next sections, let’s look into other most commonly used sensors and a demonstration of their usage.

The **rotation vector sensor** and the **gravity sensor** are the **most frequently** used sensors for **motion detection and monitoring**.

## The Rotational Vector Sensor (Sensor.TYPE\_ROTATION\_VECTOR)

* The **rotational vector sensor** is particularly versatile and can be used for a wide range of motion-related tasks, such as
  + Detecting gestures.
  + Monitoring angular change.
  + Monitoring relative orientation changes.
* The rotational vector sensor is ideal if you are developing
  + a game,
  + an augmented reality application
  + a 2-dimensional or 3-dimensional compass
  + or a camera stabilization app
* In most cases, using this sensor is a **better choice** than using the **accelerometer** and **geomagnetic field sensor** or the **orientation sensor**.
* The rotation vector represents the orientation of the device as a combination of an angle and an axis, in which the device has rotated through an angle θ around an axis (x, y, or z).
* The three elements of the rotation vector are expressed as follows:

* + The magnitude of the rotation vector is equal to.
  + The direction of the rotation vector is equal to the direction of the axis of rotation.
* The three elements of the rotation vector are equal to the last three components of a unit quaternion (cos(θ/2), x\*sin(θ/2), y\*sin(θ/2), z\*sin(θ/2)).
* Elements of the rotation vector are unitless.
* The x, y, and z axes are defined in the same way as the acceleration sensor.
* The reference coordinate system is defined as a direct orthonormal basis as shown below;  
  
* This coordinate system has the following characteristics:
  + X is defined as the vector product Y x Z.
    - It is tangential to the ground at the device's current location and points approximately East.
  + Y is tangential to the ground at the device's current location and points toward the geomagnetic North Pole.
  + Z points toward the sky and is perpendicular to the ground plane
* Below is a demonstration of the usage of **Sensor.TYPE\_ROTATION\_VECTOR**

1. Create a new “**Fragment**” to display the data retrieved from the
   1. Right click on “**java/lk.uok.mit.fragment**”, select “**New**”🡪”**Fragment**”🡪”**Fragment (Blank)**”  
      
   2. Fill the details as shown below;  
      
      1. Fragment Name:- **RotationalVectorSensorDemoFragment**
      2. Check create layout XML option
      3. Fragment Layout Name:- **fragment\_rotational\_vector\_sensor\_demo**
      4. **Uncheck both “Include fragment factory methods” and “include interface callback”**
      5. Source Language:-Java
2. Add a **menu item** to the **drawer** to access the new “**RotationalVectorSensorDemoFragment**”
   1. Open the “**drawer\_view.xml**” inside “**res/menu**” folder and add a new menu item with id “**nav\_rotational\_vector\_sensor\_demo**” like below;  
      *<?***xml version="1.0" encoding="utf-8"***?>*<**menu xmlns:android="http://schemas.android.com/apk/res/android"**>  
       <**group android:checkableBehavior="single"**>  
       <**item  
       android:id="@+id/nav\_send\_message"  
       android:title="Send Message"** />  
       <**item  
       android:id="@+id/nav\_hello\_world"  
       android:title="Message List"** />  
       <**item  
       android:id="@+id/nav\_accelerometer\_demo"  
       android:title="Accelerometer Demo"** />  
       <**item  
       android:id="@+id/nav\_rotational\_vector\_sensor\_demo"  
       android:title="Rotational Vector Sensor Demo"** />  
       </**group**>  
      </**menu**>
3. Add an entry to the switch case statement inside “**NavigationItemSelectedListener**” inside “**MainActivity**”
   1. Open “**MainActivity**” inside “**java**” folder

Modify the “**setNavigationItemSelectedListene**r(” method of “**NavigationView**” inside “**onCreate**” method like shown below;  
 *//get a reference to the navigation view* NavigationView navigationView = findViewById(R.id.***nav\_view***);  
 navigationView.setNavigationItemSelectedListener(  
 **new** NavigationView.OnNavigationItemSelectedListener() {  
 @Override  
 **public boolean** onNavigationItemSelected(MenuItem menuItem) {  
 *// set item as selected to persist highlight* menuItem.setChecked(**true**);  
 *// close drawer when item is tapped* **mDrawerLayout**.closeDrawers();  
 *// Add code here to update the UI based on the item selected  
 //to determine which menu item is clicked, add a switch* Intent intent = **null**;  
 FragmentTransaction ft =**null**;  
 **switch** (menuItem.getItemId()) {  
 **case** R.id.***nav\_send\_message***:  
 *//navigate to SendMessageActivity* intent = **new** Intent(**context**, SendMessageActivity.**class**);  
 startActivity(intent);  
 **break**;  
 **case** R.id.***nav\_hello\_world***:  
 *//Navigate to HelloWorldActivity* intent = **new** Intent(**context**, HelloWorldActivity.**class**);  
 startActivity(intent);  
 **break**;  
 **case** R.id.***nav\_accelerometer\_demo***:  
 *//Open the AccelerometerDemoFragment  
 // Begin the transaction* ft = getSupportFragmentManager().beginTransaction();  
 *// Replace the contents of the container with the new fragment* ft.replace(R.id.***fragment\_content***, **new** AccelerometerDemoFragment());  
 *// or ft.add(R.id.content\_frame, new MainFragment());  
 // Complete the changes added above* ft.commit();  
 **break**;  
 **case** R.id.***nav\_rotational\_vector\_sensor\_demo***:  
 *//Open the RotationalVectorSensorDemoFragment  
 // Begin the transaction* ft = getSupportFragmentManager().beginTransaction();  
 *// Replace the contents of the container with the new fragment* ft.replace(R.id.***fragment\_content***, **new** RotationalVectorSensorDemoFragment());  
 *// or ft.add(R.id.content\_frame, new MainFragment());  
 // Complete the changes added above* ft.commit();  
 **break**;  
 }  
 **return true**;  
 }  
 });  
}

1. Modify the “**fragment\_accelerometer\_demo.xml**” to display the three values that could be retrieved from the Rotational Vector Sensor,
   1. Open the **fragment\_rotational\_vector\_sensor\_demo.xml**  file inside “**res/layout**” folder and add the code below;

Source code for this tutorial part can be found in Git Repository given below: - <https://github.com/nadee158/android_tutorial_part_6.git>

# References

Ableson, F. (2009, June 16). *Tapping into Android's sensors*. Retrieved from www.ibm.com: https://www.ibm.com/developerworks/library/os-android-sensor/index.html

*Android AutoCompleteTextView Example*. (2018, January 1). Retrieved from JavaTPoint: https://www.javatpoint.com/android-autocompletetextview-example

*Application Fundamentals*. (2018, May 2018). Retrieved from Android Developers: https://developer.android.com

CHUGH, A. (2018, April 2). *Android AutoCompleteTextView Example Tutorial*. Retrieved from JournalDev: https://www.journaldev.com/9574/android-autocompletetextview-example-tutorial

*Creating and Using Fragments*. (2015, 01 01). Retrieved from guides.codepath.com: https://guides.codepath.com/android/creating-and-using-fragments

google-developer-training. (2018, June 18). *Working with sensor data*. Retrieved from google-developer-training.gitbooks.io: https://google-developer-training.gitbooks.io/android-developer-advanced-course-practicals/unit-1-expand-the-user-experience/lesson-3-sensors/3-1-p-working-with-sensor-data/3-1-p-working-with-sensor-data.html

Green, P. (2015, April 03). *Using Android Sensors in Your App*. Retrieved from www.sitepoint.com: https://www.sitepoint.com/using-android-sensors-application/

Hathibelagal, A. (2017, January 27). *Android Sensors in Depth: Proximity and Gyroscope*. Retrieved from Envato.com: https://code.tutsplus.com/tutorials/android-sensors-in-depth-proximity-and-gyroscope--cms-28084

*Sensors Overview*. (2018, June 19). Retrieved from Android Developers: https://developer.android.com/guide/topics/sensors/sensors\_overview

Tutorialspoint. (2018, June 18). *Android - Sensors*. Retrieved from Tutorialspoint: https://www.tutorialspoint.com/android/android\_sensors.htm