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

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



Table of Contents

[Introduction 2](#_Toc517644516)

[References 3](#_Toc517644517)

# 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,

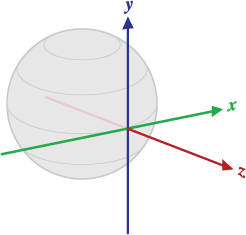
# Android Motion Sensors

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   
   previously we created a spate layout class and java class for a Fragment and then combined it to get a good understanding of a Fragment.  
   Here let’s use the android Studio’s inbuilt facility to generate a fragment.

Right click on “**java/lk.uok.mit.fragment**”, select “**New**”🡪”**Fragment**”🡪”**Fragment (Blank)**”  


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