# A Sensors App

### Aims:

To introduce developing an Android App with sensor management

### Objectives:

* Utilising the SensorManager class
* Handling Sensor Events
* Sub classing a View to display sensor values

## 1. Introduction

Here we’ll use Android Studio and Java to develop an application that utilises sensor information.

## 2. Starting a new project

From windows start Android Studio and start a new Android Studio project. Choose an **Empty Activity**, name the project ‘**SensorsApp**’ and ensure the language is ‘**Java**’ before you click [Finish]. Wait for Gradle sync and indexing to finish (see bottom right of Android Studio).

From the Project explorer right click on …

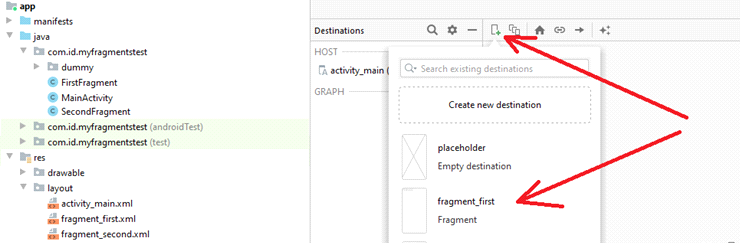
**app/java/com.*your-id*. sensorsapp**

to create a **New / Fragment / Fragment (Blank)**. In the resulting dialog select the ‘Fragment Name:’ to be ‘**FirstFragment**’

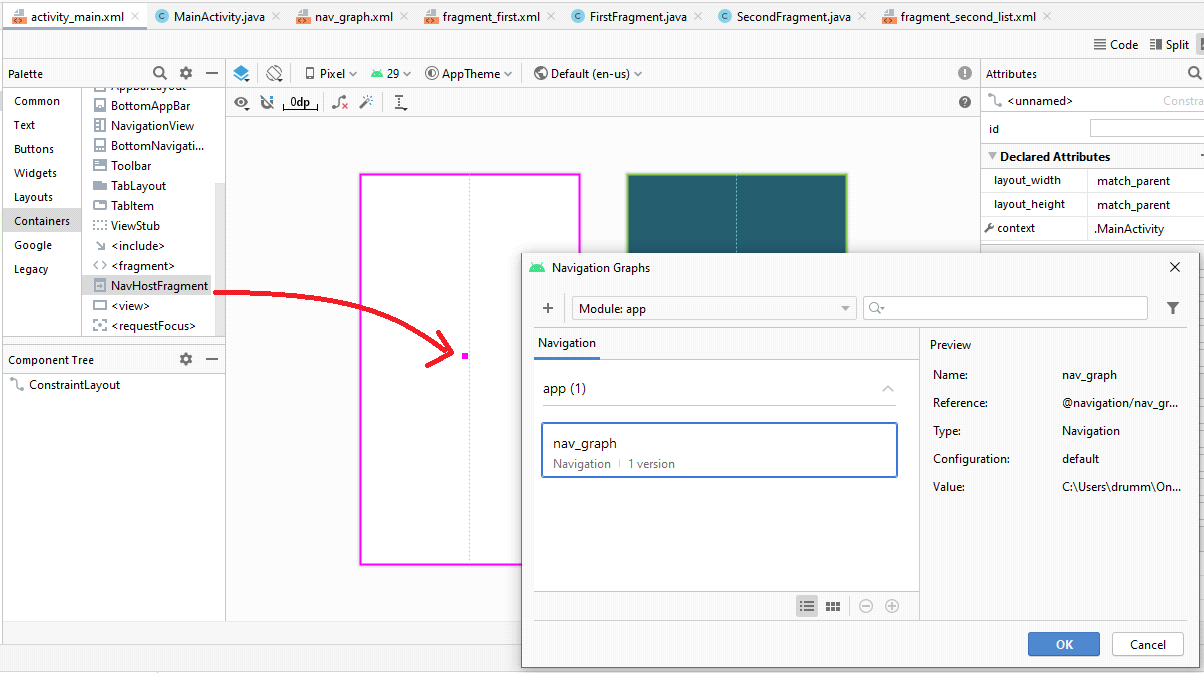
Right click on ‘**res**’ directory to create a new Android Resource File, from the resulting dialog set ‘File Name’: to be ‘**nav\_graph**’ and select ‘Resource type :’ to be ‘navigation’.

You maybe be prompted with a message to add additional libraries, click [OK]. *Note if you have issues later, then it could be related to missing or incorrect dependencies … see* [*https://developer.android.com/jetpack/androidx/releases/fragment*](https://developer.android.com/jetpack/androidx/releases/fragment)

In the Design View you will see an empty navigation graph. Hence, choose to add your **fragment\_first** to the graph



Go to **activity\_main.xml** file, delete the TextView, and hence drag and drop from Containers a **NavHostFragment** onto your design view, hence when prompted, choose the **nav\_graph** you created earlier.



From the nav\_host\_fragment’s attribute inspector, change…

* the id to **nav\_host\_fragment**.
* **layout\_width** and **layout\_height** to **match\_parent**.

## 3. Sub classing the View class

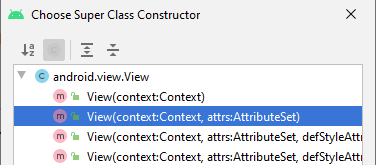
To demonstrate sensors it’ll be useful to create a custom **View** class. From the Android Explorer right click on ‘app/java/com.*yourpackage*.sensorsapp’ to create a **New->Java Class** called **MyView**.

In the resulting **MyView.java** file you’ll have a new class defined…

**package** com.example.ian.drawingapp;

**public class** MyView {  
}

Subclass View using the ‘extends’ keyword. Let Android Studio ‘Import View (view.android)’ and **‘Add constructor MyView(context:Context, attrs:AttributeSet)’** *… using [Alt Enter]*.



Your class definition should now look like…

**public class** MyView **extends** View {  
**public** MyView(Context context, @Nullable AttributeSet attrs) {  
 **super**(context, attrs);  
 }  
}

In your MyView class declare the fields

**private float x**,**y**,**z**;

**private** Paint **p** = **new** Paint();

Again using ‘ALT ENTER’ for import Paint and all subsequent class names.

Hence in the **MyView()** constructor create an instance of Paint and set its properties.

p=**new** Paint(Paint.***ANTI\_ALIAS\_FLAG***);

p.setStyle(Paint.Style.***FILL\_AND\_STROKE***);

p.setColor(Color.***GREEN***);

p.setStrokeWidth(3);

p.setTextSize(80);

setBackgroundColor(Color.LTGRAY);

Android will call MyView’s **onDraw()** method when needed. You can override this method.

Using ‘ALT INSERT’, ‘Override Methods…’ add

@Override

**protected** **void** onDraw(Canvas canvas) {

**super**.onDraw(canvas);

}

And add to this method code to draw the string to canvas.

canvas.drawText(**"Hello "**+**x**+**","**+**y**+**","**+**z**, 300+50\***x**, 600+50\***y**, **p**);

It’ll be useful to define a setter methods in MyView to set the x,y and z.

Using ‘ALT INSERT’, ‘Getter and Setter’ add…

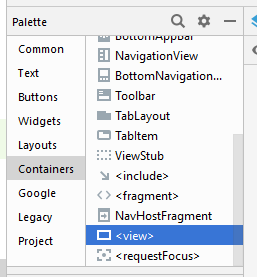
**public void** setX(**float** x) {  
 **this**.**x** = x;  
}  
  
**public void** setY(**float** y) {  
 **this**.**y** = y;  
}  
  
**public void** setZ(**float** z) {  
 **this**.**z** = z;  
}

Now that we have our sub class of **View** defined, we can declare and instantiate it and associate it with our user interface via our first fragments xml.

Go to **first\_fragment.xml**

Right click on the frame layout in the component tree to convert is to a **constraint layout**.

Drag a text view onto the design area and set its text to “Draw something in canvas”. You might also want to change the text size.



Drag a <view> onto the design area for this fragment. A dialog box will offer you a choice of classes to associate with this, choose MyView.

From the view’s attribute inspector, change…the id to **myview**.

Constrain the view horizontally and vertically.

#### 

Save and Run As Android Application.

## 4. Capturing sensor events

You can access your device’s sensors by declaring and instantiating a Sensor Manager. You can then associate it with a realisation of the SensorEventListener interface to handle events from the Sensors.

Firstly, in **FirstFragment.java**, declare as fields of **FirstFragment** class

SensorManager **sensorManager**;  
 Sensor **sensor**;  
 **float x**,**y**,**z**;  
 MyView **myView**;

Override FirstFragment’s **onViewCreated()** method. Hence, in the method bind these references to instances.

**sensorManager** = (SensorManager)getActivity().getSystemService(Context.***SENSOR\_SERVICE***);

**sensor**=**sensorManager**.getDefaultSensor(Sensor.***TYPE\_ACCELEROMETER***);

**myView**=view.findViewById(R.id.***myview***);

You will need to create an object to handle sensor events. Declare as a field

SensorEventListener listener = **new** SensorEventListener() {};

Use ‘ALT ENTER’ to let Android Studio ‘Implement Methods’ hence realise the SensorEventListener interface by defining an anonymous inner class, to give…

SensorEventListener listener = **new** SensorEventListener() {

@Override

**public** **void** onSensorChanged(SensorEvent event) {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** onAccuracyChanged(Sensor sensor, **int** accuracy) {

// **TODO** Auto-generated method stub

}

};

This hence instantiates an object that can bind to the sensor and handle sensor events. In its **onSensorChanged()** method you can capture sensor events and pass values to a view to display the them. For example, add to onSensorChanged() the following code

**x**=event.**values**[0];  
 **y**=event.**values**[1];  
 **z**=event.**values**[2];  
  
 **myView**.setX(**x**);  
 **myView**.setY(**y**);  
 **myView**.setZ(**z**);  
 **myView**.invalidate();

The code ,,,

* pulls x,y,z sensor values from the sensor event,
* sets the x,y,z values of your instance of MyView.

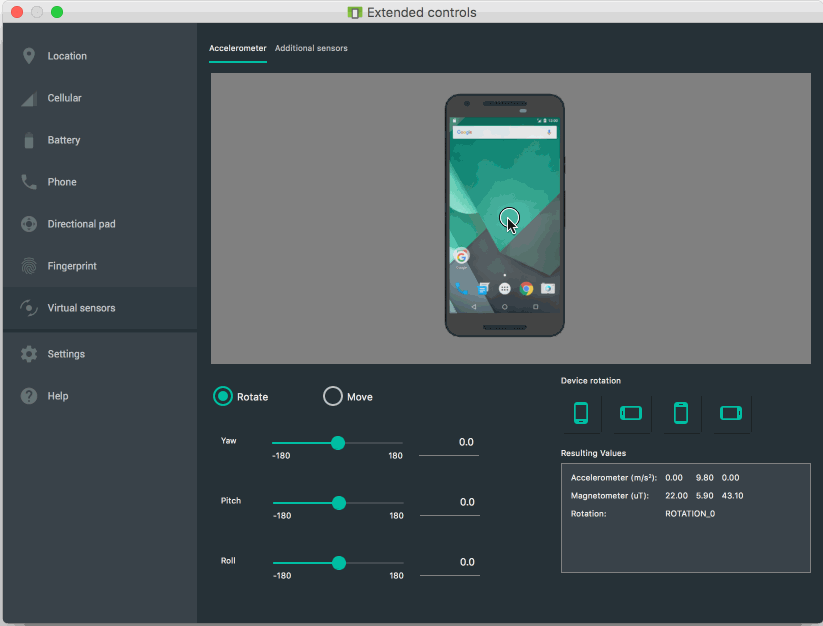
Notice the use of **myView.invalidate()** to schedule a redraw of its canvas.

Finally, in the **onViewCreated()** method associate the **listener** object which the **sensor** …

**sensorManager**.registerListener(listener,**sensor**,SensorManager.***SENSOR\_DELAY\_NORMAL***);

Save and run the Android Application.

Android studio’s emulator provides virtual sensors, try experimenting with this.



You might also want to try building to a real mobile device to see how the accelerometer values changes as you move the device.

#### EXERCISE

Try another sensor for example a Gyroscope the returns x, y and z to indicate the rotation of your device.

Declare as fields of the MyView class.

float gx=0,gy=0,gz=0;

and a display them with…

DecimalFormat df = new DecimalFormat("0.00");

someString = "Gyro x, y, z=" + df.format(gx) + "," + df.format(gy) + ", "+ df.format(gz));

Note the use of the DecimalFormat class for displaying the numbers more concisely.

So in the listener you could include…

if (event.sensor.getType() == Sensor.TYPE\_GYROSCOPE) {

gx = event.values[0];

gy = event.values[1];

gz = event.values[2];

}

To register the event handler to the sensor manager use

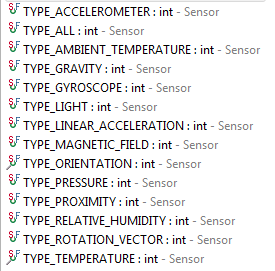
Sensor gyroscope;

gyroscope = sensorManager.getDefaultSensor(Sensor.***TYPE\_GYROSCOPE***);

sensorManager.registerListener(listener, gyroscope, SensorManager.***SENSOR\_DELAY\_NORMAL***);

#### EXERCISE

There are a large range of sensors available



Extend your application to display information for a variety of sensor types. To do so you could add UI controls to change the current senor viewed.

You may need to google around to understand the usage of some of the sensors available to you.

#### EXERCISE

You may wish to create more subclasses of View in other fragments, to display data from different sensors differently.

Try to display information from likes of accelerometer, gyroscope, etc with graphics, for example by drawing rectangles with changing size, lines graphs, moving around a circle or even moving an image view.

## COMPLETE CODE EXAMPLE

**MyView.java**

**package** com.id.sensorsapp;  
  
**import** android.content.Context;  
**import** android.graphics.Canvas;  
**import** android.graphics.Color;  
**import** android.graphics.Paint;  
**import** android.hardware.Sensor;  
**import** android.hardware.SensorManager;  
**import** android.util.AttributeSet;  
**import** android.view.View;  
  
**import** androidx.annotation.Nullable;  
  
**public class** MyView **extends** View {  
 **private float x**,**y**,**z**;  
 **private** Paint **p** = **new** Paint();  
  
 **public** MyView(Context context, @Nullable AttributeSet attrs) {  
 **super**(context, attrs);  
 **p**=**new** Paint(Paint.***ANTI\_ALIAS\_FLAG***);  
 **p**.setStyle(Paint.Style.***FILL\_AND\_STROKE***);  
 **p**.setColor(Color.***GREEN***);  
 **p**.setStrokeWidth(3);  
 **p**.setTextSize(80);  
 setBackgroundColor(Color.***LTGRAY***);  
 }  
  
  
 @Override  
 **protected void** onDraw(Canvas canvas) {  
 **super**.onDraw(canvas);  
 canvas.drawText(**"Hello "**+**x**+**","**+**y**+**","**+**z**, 300+50\***x**, 600+50\***y**, **p**);  
 }  
  
 @Override  
 **public void** setX(**float** x) {  
 **this**.**x** = x;  
 }  
  
 @Override  
 **public void** setY(**float** y) {  
 **this**.**y** = y;  
 }  
  
 @Override  
 **public void** setZ(**float** z) {  
 **this**.**z** = z;  
 }  
}

**FirstFragment.java**

**package** com.id.sensorsapp;  
  
*Default Imports here…*

**public class** FirstFragment **extends** Fragment {  
  
*Default code here…*

SensorManager **sensorManager**;  
 Sensor **sensor**;  
 **float x**,**y**,**z**;  
 MyView **myView**;  
  
 @Override  
 **public void** onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {  
 **super**.onViewCreated(view, savedInstanceState);  
 **sensorManager** = (SensorManager)getActivity().getSystemService(Context.***SENSOR\_SERVICE***);  
 **sensor**=**sensorManager**.getDefaultSensor(Sensor.***TYPE\_ACCELEROMETER***);  
 **myView**=view.findViewById(R.id.***myview***);  
 SensorEventListener listener = **new** SensorEventListener() {  
 @Override  
 **public void** onSensorChanged(SensorEvent event) {  
 **if**(event.**sensor**.getType()==Sensor.***TYPE\_ACCELEROMETER***)  
 {  
 **x**=event.**values**[0];  
 **y**=event.**values**[1];  
 **z**=event.**values**[2];  
  
 **myView**.setX(**x**);  
 **myView**.setY(**y**);  
 **myView**.setZ(**z**);  
 **myView**.invalidate();  
 }  
  
 }  
  
 @Override  
 **public void** onAccuracyChanged(Sensor sensor, **int** accuracy) {  
  
 }  
 };  
  
 **sensorManager**.registerListener(listener,**sensor**,SensorManager.***SENSOR\_DELAY\_NORMAL***);  
 }  
}

**Fragment\_first.xml**

*<?***xml version="1.0" encoding="utf-8"***?>*<**FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".FirstFragment"**>  
  
 <**view  
 android:id="@+id/myview"  
 class="com.id.sensorsapp.MyView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"** />  
</**FrameLayout**>

**nav\_graph.xml**

*<?***xml version="1.0" encoding="utf-8"***?>*<**navigation xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:id="@+id/nav\_graph"  
 app:startDestination="@id/firstFragment"**>  
  
 <**fragment  
 android:id="@+id/firstFragment"  
 android:name="com.id.sensorsapp.FirstFragment"  
 android:label="fragment\_first"  
 tools:layout="@layout/fragment\_first"** />  
</**navigation**>

**activity\_main.xml**

*<?***xml version="1.0" encoding="utf-8"***?>*<**androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity"**>  
  
 <**fragment  
 android:id="@+id/fragment"  
 android:name="androidx.navigation.fragment.NavHostFragment"  
 android:layout\_width="409dp"  
 android:layout\_height="729dp"  
 app:defaultNavHost="true"  
 app:navGraph="@navigation/nav\_graph"  
 tools:layout\_editor\_absoluteX="1dp"  
 tools:layout\_editor\_absoluteY="1dp"** />  
</**androidx.constraintlayout.widget.ConstraintLayout**>