**Bubble Level**

**Project Description:**

The goal of developing this project is to understand the working of Android activity life cycle and how we can work with device sensors along with Custom View for development of bubble level application using Canvas.

**Files / Pages:**

This project consists of few .java files (for same activity but different classes) and 3 .xml files for UI layout.

XML files involved are:

1. activity\_main.xml

Java files involved are:

1. ActivityMain.java
2. HandleData.java
3. OneD\_CV.java
4. RecordedValues.java
5. TwoD\_CV.java

**activity\_main.xml**

activity\_main.xml is the only screen in this application. This xml page consists of all required text views to display different data of X, Y and Z axis along with Inclination in degrees, Minimum inclination observed and Maximum inclination observed so far. This xml page also consists of custom views for 1 dimension and 2-dimension view. When the device is placed on flat surface, 2D view will be enabled and when device is not placed on flat surface but is held in Landscape or portrait mode, 1D view will be enabled. Below are the images of this application’s screen in android device.

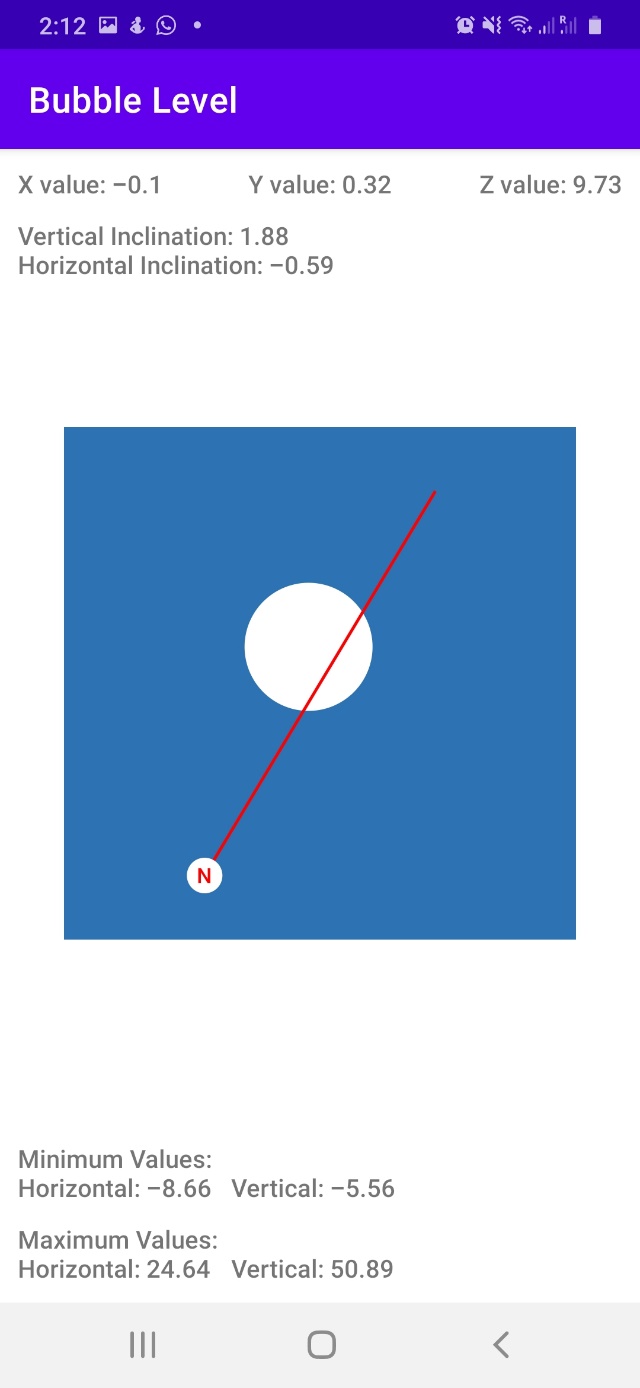
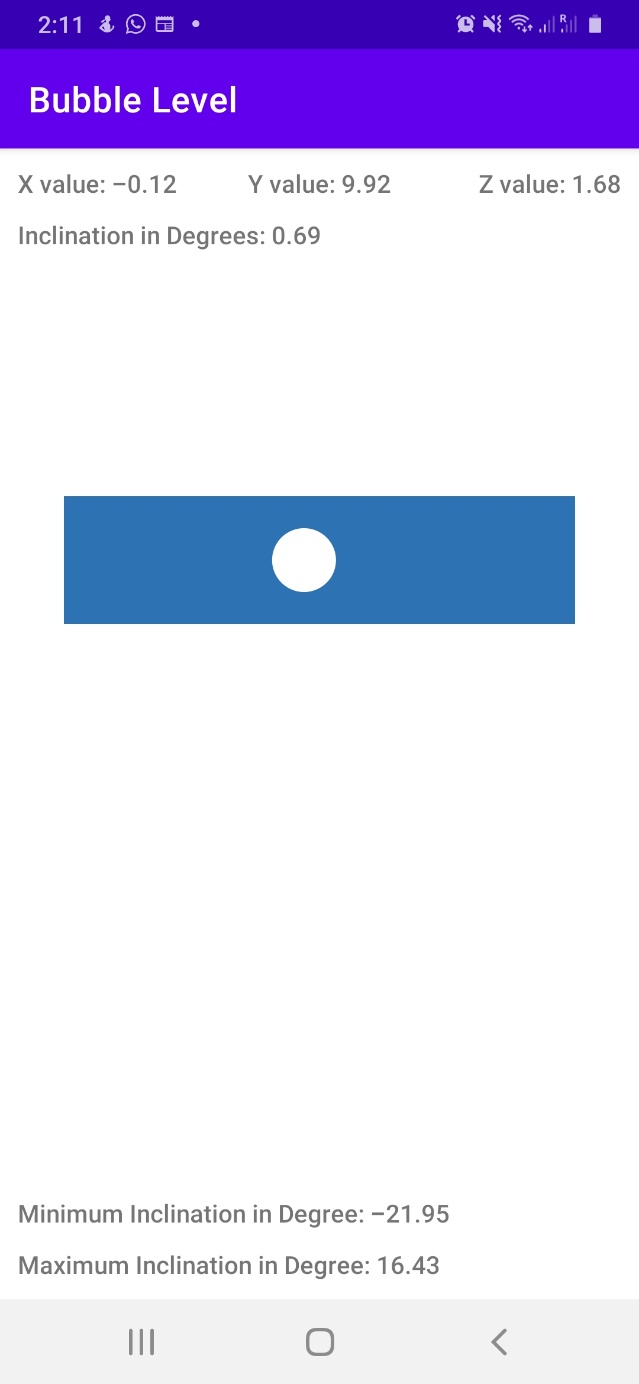
 

Figure: 1.1 Figure: 1.2

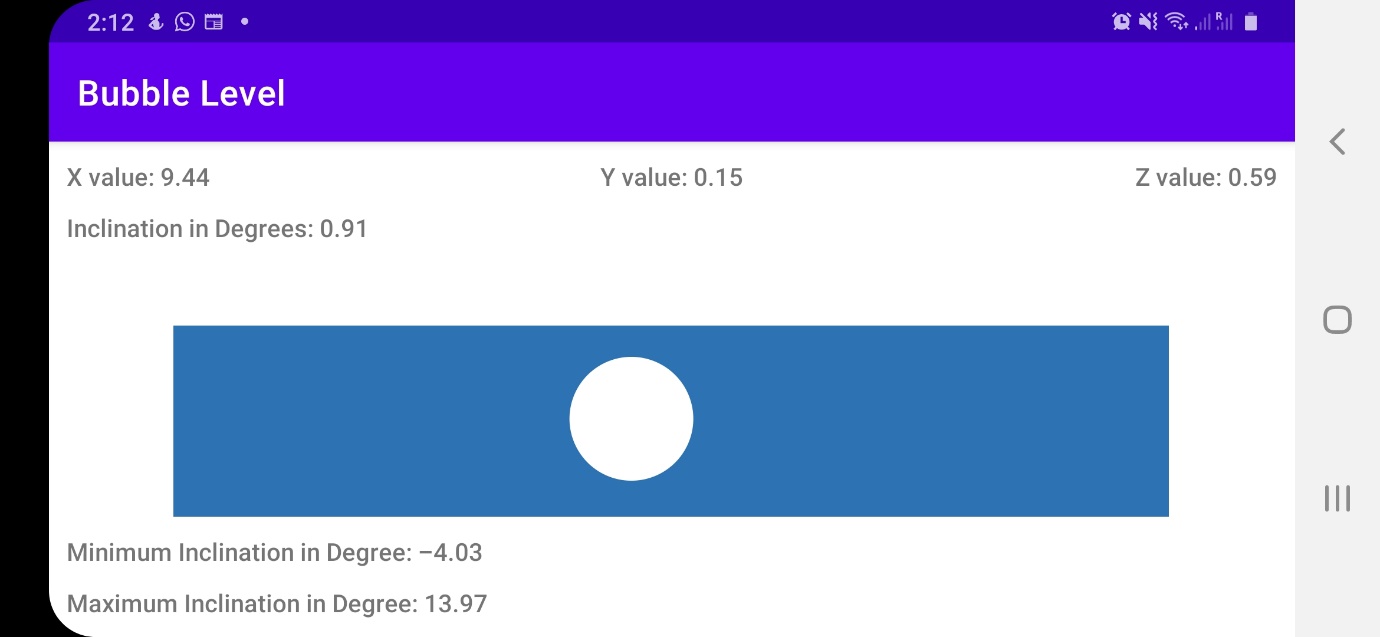


Figure: 1.3

As shown in figure 1.1, as the phone is placed on a flat surface, a 2D custom view will be displayed. As shown in figure 1.2 and 1.3, 1D view will change as per screen orientation from Landscape or Portrait mode.

**MainActivity.java**

MainActivity.java consist of the main class responsible to run this program i.e., onCreate() method of home activity. It also consists of below methods responsible to handle views and sensor events on home page of the Bubble Level application.

1. onCreate() method:

onCreate method consists of main logic to handle multiple events on this application. This method consists of a logic to handle which custom view from 1D and 2D custom views needs to be made visible at any instance of using this application. 1D custom view will be used when phone is not kept on flat surface and is in either portrait or landscape mode. This method also contains listeners registered to listen to values recorded by accelerometer and magnetic field sensors. These X, Y, Z axis values are then stored in list handled by handleData class which are stored in the form of object of classes OneD\_Values or TwoD\_Values as per device orientation. These values are then used to calculate inclination angle in degree which is also stored in object as mentioned above. Based on these inclination values, bubble in device will be moving on canvas. A logic to handle screen refreshing after each recorded value is written in onCreate method.

This method also consists of a logic to identify if required sensors are present on device or not. In case sensors are not present, none from above logics will be executed and an appropriate message will be displayed on UI to user. All the text views will be updated appropriately at all instance of recorded value. In case device’s Z-axis value is more than 5.0, device is assumed to be kept on flat surface and 2D custom view will be enabled. In case the value is in range of -5.0 to 5.0, device is identified to which from 2 orientation is held. This is identified using orientation sensor present on the device.

**HandleData.java**

HandleData.java will be called from MainActivity.java, OneD\_CV.java and TwoD\_CV.java. This class is a POJO (Plain Old Java Object) class used to handle maximum of 500 records at a time. This class will record values from sensors up to 500 instances. Once 500 values are recorded, next value will be stored from position 0 which will we overridden with previous value. This class consists of 1 constructor and 3 methods.

Constructor will be used to reset all data in this class. Once constructor is triggered, all the variables will be back to initial state.

Methods defined in this class are:

1. recordValues method:

This method will be used to record values in static list and increment global counter.

1. incrementIndex method:

This method will be called from recordValues method. A logic is written to increment global counter. In case counter goes above 500, initialize it to 0.

1. getLastData method:

This method is responsible to return last record that was inserted in list. This will always be current counter count minus 1.

**OneD\_CV.java**

OneD\_CV.java will be called from MainActivity.java. This class will be called when Z-axis value sensed by sensor is in range of -5.0 to 5.0 units. Below are the methods written in this class:

1. onMeasure method:

onMeasure method is overridden in this class. This method is responsible to identify the screen width and height. Based on these values, bubble device width and height will be defined along with offset. This method finally calls init method.

1. Init method:

Init method is responsible to initialize all the required variables and set the colour codes we will be using in this application to display bubble 1D view. This method is also responsible to create an object and reset required variables in HandleData class.

1. onDraw method:

onDraw method is responsible to draw the bubble device along with bubble on application screen. This method consists of logic required to render canvas designs.

**RecordedValues.java**

RecordedValues.java is also a POJO (Plain Old Java Object) class used to record data. Objects of this class will be stored in list of HandleData class. Each object will represent a set of values recorded by required sensors. This class consists of getters and setters along with 2 convert methods. One convert method is used to restrict float values to 2 digits after decimal point and another convert method is used to restrict double values to 2 digits after decimal point.

**TwoD\_CV.java**

TwoD\_CV.java will be called from MainActivity.java. This will be called when device is placed on flat surface. In such cases, the Z-axis values will be over 5.0 in positive direction and below -5.0 in negative direction. This class structure is similar to that of OneD\_CV except few logics to handle screen rendering. Below are the methods defined in TwoD\_CV.java:

1. onMeasure method:

This method is responsible to identify device width. Based on device width, canvas designs will be displayed according to different screen sizes. This method finally calls init method.

1. Init method:

Init method is responsible to initializing required variables and objects. This method is also responsible to initialize colour codes required to draw on canvas.

1. onDraw method:

onDraw method is similar to the one in 1D custom view. There is a basic difference of logic for drawing 2 dimensional shapes on canvas. There is an additional logic to handle displaying North direction on the screen which is not displayed in case of 1D view.

1. plotNorth method:

This method will be called from onDraw method. This method is responsible to handle rotation of direction line on 2-dimensional custom view. This method returns the float array consisting of points to be plotted on 2D custom view. These returned points are calculated as per x and y axis values recorded from magnetic field sensor.

**Below is the mathematical calculation for rendering 1D and 2D custom view:**

I have performed a lot of calculations to identify canvas size and size of elements drawn on canvas as per user screen size. All these calculations are performed based on device screenWidth and device screenHeight.

**Data structure:**

I have used 1D list to store maximum of 500 recorded values in the form of object of class. The same 1D list will be used to store 1D and 2D object data as per screen orientation and the way screen is placed. There is no requirement of using any database as we are storing these 500 records on temporary basis.

**Design Decisions:**

1. I have tried to keep the UI self-explanatory and easy to understand by users. I have added text views with appropriate and self-explanatory text to indicate their use to users without having any second thought of doubt about the same.
2. I have also handled 1D and 2D custom views appropriately so that they will not collide at with each other at any instance.
3. I have also added a toast message to display appropriate message in case magnetometer is missing on the device and if direction cannot be displayed on 2D custom view.
4. I have also used a proper combination of colour codes to make the UI distinguishable.