DESIGN AND IMPLEMENTATION OF SENSOR BASED ORIENTATION APPLICATION

OVERVIEW:

In this application, three sensors, namely accelerometer, magnetometer, and gyroscope, are used to determine the orientation of a device. Initially, we calculate the orientation values by combining accelerometer and magnetometer but since these values give an inaccurate orientation of the device, gyroscope is introduced into the picture. The initial values calculated from accelerometer and magnetometer are combined with that of the values from the gyroscope after they go through some additional processing. The three-sensor combination is performed to compensate for the gyro drift and reduce noise in the orientation. The accelerometer and magnetometer data are used for a longer time period which is similar to using a low pass filter whereas the gyroscope is used for shorter stints and it's similar to using a high pass filter.

COMPONENTS USED:

Accelerometer	Handler	Manual Functions	Data elements
Gyroscope	Sensor Manager	On Click Listener	Override Methods
Magnetometer	Timer	Sensor Event Listener	UI Elements

- The main components used in the application will be the three sensors accelerometer, gyroscope, and magnetometer. The data from these sensors is stored to be used for determining the final orientation.
- A Sensor Manager variable is used to handle the sensors and any mathematical functions needed to process the sensor data.
- We will also be using a timer in the application because it's necessary to take the gyroscope sensor values over a certain period based on the requirement.
- Apart from this, we will be using a handler to update the UI whenever there is a change in the orientation to avoid the race condition.
- Manual functions are created to calculate the combined orientation of accelerometer and magnetometer, gyro orientation which includes all the additional processing, and another function to calculate the final orientation. Other functions are used to calculate the result of matrix multiplication, rotation matrix from orientation values and so on.
- Since the application only starts running when the button is clicked, we use the on-click listener as a kick-starter for the application to perform all the necessary calculation behind the screen.
- The Sensor Event listener checks whenever there is a change in each sensor and triggers the data processing.

- The override methods used are onResume(), onPause(), and onStop() to register and unregister the listeners respectively according to their requirement.

UI:

The UI of this application mainly consists of three numeric fields to display the azimuth, pitch, and roll values describing the orientation of the device. We add a button which when clicked will be able to kickstart the application and the necessary calculation is performed in the background to display the values in the above-mentioned fields.

The UI is a simple click-and-display, and the values will be continuously updating as per the change in the orientation of the device. It can be observed that even when the device is stopped rotating, the values will keep changing for a while. This is because the application is still processing all the values from the recent rotation of the device.

ALGORITHM:

In simple terms, we will be performing high pass filtering of the data which is obtained from the gyroscope and low pass filtering will be done for combined values obtained from accelerometer and magnetometer. Before this calculation is performed, each of these sensors' data will undergo additional processing.

We can divide the algorithm into different stages:

1. In the first stage, the application is ready to display the orientation of the device. Once the button is clicked, the sensors are registered, and the timer is started.
2. Once the first stage is done, the sensor event listeners will be detecting any changes in the orientation of the device. For this detection, we will be using the onSensorChanged function. If the sensor event detects any change in one of the sensors, the values are recorded, or the necessary functions are called to perform the necessary mathematical processing before the output is displayed on the screen.
3. Initially, after the accelerometer and magnetometer values are recorded, we will get a rotation matrix from these combined values for which we can use the SensorManager functions. From the rotation matrix, we will calculate the orientation matrix. These values are stored in an array and are used in future processing. These values are continuously calculated and stored whenever a change is detected in either of these sensors.
4. Once a change in gyroscope values is also recorded, we call a separate function to process the gyro data because it requires a lot of additional processing. Once the gyro data is collected, we normalize the data if required and then the data is integrated with respect to time. This produces a rotation vector expressing the change in rotation, like that of a quaternion. We then calculate the rotation matrix from which the orientation matrix will be calculated.
5. Once we have all these values, we use a separate function to calculate the Orientation with data from all these sensors. We use the high pass and low pass filtering technique as mentioned above and get the resultant orientation. Once this calculation is done, we will be assigning a task to the handler which will display the orientation values on the UI.

APPLICATION LIFECYCLE