Introduction to Mobile computing Assignment1

Py32577 Sreevikas Edukulla

This code is calculating the orientation of a device in three dimensions (pitch, roll, and yaw) using accelerometer and magnetometer data.

1)Finding resultant acceleration

double resultant = Math.sqrt(acclX*acclX + acclY*acclY + acclZ*acclZ);

2) Calculate pitch and roll angles from the accelerometer data

3) Calculate yaw angle from magnetometer data:

```
private void calculateAccRotation() {
    double resultant = Math.sqrt(acclX*acclX + acclY*acclY + acclZ*acclZ);
    pitchAcc = (float) Math.toDegrees(Math.asin(-(acclY)/resultant));
    rollAcc = (float) Math.toDegrees(Math.asin(acclX/resultant));
    float temp1 = (float) ((-magX) * Math.cos(rollAcc) + magZ *
Math.sin(rollAcc));
    float temp2 = (float) ((magX * Math.sin(pitchAcc) * Math.sin(rollAcc)) +
(magY * Math.cos(pitchAcc)) +  (magZ * Math.sin(pitchAcc) *
Math.cos(rollAcc)));
    yawAcc = (float) Math.toDegrees(Math.atan2(temp2,temp1));

}
```

Algorithm for getting rotation using gyroscope sensor data.

1) Find calculate change in angular velocity and multiply with time to get angle travelled.
2) Add previous displacement value with new get the actual rotation.
3) Divide with 360 as rotation values lies below 360

```
private void calculateGyrRotation(long eventTime) {
    deltaTime = eventTime - lastRecordingTime;
    lastRecordingTime = eventTime;
    pitchGyro = (float) (pitchGyro + ((-gyrX) * deltaTime * 0.000000001f *
57.3f))%360;
    rollGyro = (float) (rollGyro + ((gyrY) * deltaTime * 0.000000001f *
57.3f))%360;
    yawGyro = (float) (yawGyro + ((-gyrZ) * deltaTime * 0.000000001f *
57.3f))%360;
   // System.out.println(pitchGyro + "_____" + rollGyro + "___________" +
yawGyro);
```

```
}
```

Used weighted average as fusion algorithm

A constant value alpha is defined as 0.98, which is a weighting factor used to blend the gyroscope and accelerometer measurements.

Gyroscope is given a 98% value because it is very precise. and accelerometer values are used, it doesn't drift over time.

```
double  alpha = 0.98;
double result_pitch = pitchGyro * alpha + (1 - alpha) * pitchAcc;
double result_roll = rollGyro * alpha + (1 - alpha) * rollAcc;
double result_yaw = yawGyro * alpha + (1 - alpha) * yawAcc;
```

.