Conditionals (What's up?)

Lab 5

SUBMITTED BY:

CODY BROOKS

SUBMISSION DATE:

2015-10-06

Problem

The purpose of this lab is to create a program that will display the current orientation of the Esplora and will exit when the user presses the 'UP' button. The inputs to the program will be accelerometer data and time from `explore.exe` and the output will be the orientation of the device. The objective is to teach loops, conditional statements, functions and comparing floating point numbers.

Analysis

The problem states that the program must display the orientation of the Esplora while it is not moving. The Esplora is still when the magnitude of its acceleration is equal to one. The magnitude of acceleration is calculated using the formula $\sqrt{x^2+y^2+z^2}$ .When the Esplora is still, it's orientation is dependent on which of the three accelerometer values (X, Y or Z) is equal to 1 or -1. If the accelerometer has an X value of 1, then the Esplora is facing left. If the accelerometer has an X value of -1 then the Esplora is facing right. If the accelerometer has a Y value of 1, then the Esplora is facing the front. If the accelerometer has a Y value of -1, then the Esplora is facing the back. If the accelerometer has a Z value of 1, then the Esplora is facing up. If the accelerometer has a Z value of -1, then the Esplora is facing down.

Design

The problem requires that the the program runs until the user presses the 'UP' button on the Esplora. This means that the main `while` loop's condition will be based on the value of the 'UP' button that is collected using `scanf()`. To evaluate the orientation of the Esplora, the X, Y and Z accelerations are passed into the function `getDirection()`. To make the program easier to read, `getDirection()` returns an enum value describing the direction. The enum contains each direction (UP, DOWN, LEFT, RIGHT, FRONT and BACK) and is defined before the `main()` function.

The `getDirection()` function compares each X, Y and Z input to 1 and -1 to determine its orientation as outlined in the above Analysis. Because X, Y and Z are floating point values, conditional statements cannot use '=='. To solve this, a `closeTo()` function was created. This function takes the parameters `double tolerance`, `double target` and `double val`. `closeTo()` checks to see if val is within the limits `(target - tolerance)` and `(target + tolerance)` and returns a boolean value (using the `stdbool.h` library). `closeTo()` is then used within `getDirection()` to compare each X, Y and Z value to 1 and -1.

Testing

The main test is to see the output when the Esplora data is piped through `lab5.c`. By checking to see if the orientation displayed matches the actual orientation of the board, the main function of the program is checked. Another thing to check is to see if an orientation is displayed while the board is in motion. The same orientation should never be displayed twice because orientation should only be displayed if the orientation changes.

Place the board in the 'UP' position and start the program. Check to see if the orientation is displayed is 'UP'. Move the board and bring the board back to a still position in the same orientation as it started. Nothing should be displayed because the orientation did not change and nothing should be displayed while the board is in motion. Move the board to the next position and repeat the above steps.

Comments

The design choice of using enums instead of arbitrary integers had benefits and drawbacks. It made the program much easier to read and debug than if those enum constants were arbitrary integers. Because of the nature of how GCC compiles the source code the declaration of the enum `Direction` had to come outside of the `main()` function and before the prototype of the `getDirection()` function. Even though global variables are generally considered bad practice, it was worth having because it was never changed within the function.