What's up

Lab 4

Section M

Submitted by:

Haadi-Mohammad Majeed

Submission Date

5/10/2018

Date

2/10/2018

Lab Problem

The objective of this lab is to create a programme that can correctly identify which side of the Dual Shock 4 controller is facing up when at rest based on the raw controller data. To do so, 3 or more functions must be used and upon hitting the triangle button, the program should end. Another objective was to only have it output if the side has changed, as such, no side would output consecutively. The initialization of a specific method is also necessary.

Analysis

In the beginning, we were given the barebones of the loop and 3 other lines of code, two to initialize all the variables, and the third to import the raw data into the programme. The output would have to consist of a line of code that corresponded with the side that was facing up, but without yielding the same output in multiple successions. This would have to be done with a line or few that checks if the current side was the previous.

Design

The creation of the method *close_to* is required and is used to see if a value with a set tolerance is near a desired value. This could be used to check if the magnitude of acceleration is 0, not accelerating, or 1, accelerating in some direction. For a second function, the mag function from Lab 3 would be reused. This would be used along with the previously listed function to check if the controller was moving or not. As for the third function, a variable was created, one for each side of the controller, which would get modified each time the loop cycled. This function had the job of making sure it would not output the same side twice, if the side it is currently on, was equal to the previous side, it would return a zero, failing the if statement causing the code to move on.

Testing

Due to the nature of the lab, a controller's input would be required for every test, and working with pre-recorded data is not the most efficient when just a small bit needed to be tested. Subsequently, a visit to an empty lab was called for where most of the code was competed and tested with live inputs. With live raw inputs, I was able to rapidly check for errors and potential errors within the code.

Comments

I enjoyed thinking out each step of the programme prior to programming. I started by writing pseudocode in my notebook to process what each step needed to achieve. Each function was developed as to return what I would need it to as to work with the others.

Implementation

Inside the main function

```
int t, b1, b2, b3, b4, side;
double ax, ay, az, gx, gy, gz;
b1 = 0;

while (b1 != 1) //while triangle isnt pushed
 {
    scanf("%d, %lf, %lf, %lf, %lf, %lf, %lf, %d, %d, %d, %d", &t, &ax, &ay, &az, &gx, &gy, &gz, &b1, &b2, &b3, &b4);

        if ((close_to(0.03, mag(ax, ay, az), 0.0) == 1)) {
            if (close_to(0.25, gy, 1) == 1) {
                if (sideCheck(side, 1)) {
                    printf("Remote is facing: TOP\n");
                }
                side = 1;
            }
            if (close_to(0.3, gy, -1) == 1) {
                if (sideCheck(side, 2)) {
                    printf("Remote is facing: BOTTOM\n");
                }
                side = 2;
            }
            if (close_to(0.3, gx, 1) == 1) {
                if (sideCheck(side, 3)) {
                    printf("Remote is facing: RIGHT\n");
                }
                side = 3;
            }
            if (close_to(0.3, gx, -1) == 1) {
                if (sideCheck(side, 4)) {
                    printf("Remote is facing: LEFT\n");
                }
                side = 4;
            }
            if (close_to(0.3, gz, 1) == 1) {
                if (sideCheck(side, 5)) {
                    printf("Remote is facing: BACK\n");
                }
                side = 5;
            }
            if (close_to(0.3, gz, -1) == 1) {
                if (sideCheck(side, 6)){
                    printf("Remote is facing: FRONT\n");
                }
                    side = 6;
                }
        }
        fflush(stdout);
}
return 0;
```

The 3 functions that allow the programme to work

```
/* Put your functions here */
int close_to(double tolerance, double point, double value) {
    if (value > (point + tolerance) || value < (point - tolerance)) {
        return 0;
    }
    else {
        return 1;
    }
}

double mag(double x, double y, double z) {
    return sqrt((pow(x, 2.0)) + (pow(y, 2.0)) + (pow(z, 2.0)));
}

int sideCheck(int currentSide, int previousSide)
{
    if(currentSide != previousSide)
        return 1;
    if(currentSide == previousSide)
        return 0;
}
```