CONDITIONALS (“WHAT’S UP”)

LAB 4 SECTION 1

SUBMITTED BY:

ETHAN GRIESMAN

SUBMISSION DATE:

9/27/2022

Problem:

The objective for this lab was to acquire skills/proficiency using if and else-if statements in the C programming language, understand and interpret the acceleration of the DualShock controller as it relates to gravity, experiment with testing multiple conditions within code, and understand tolerances and comparing them to floating point values.

Analysis:

Approaching this problem using the concepts learned, as well as functions used in the previous labs, we were instructed to create code that parsed through incoming data regarding time, the acceleration and orientation along the x, y, and z axis, as well as the total numbers being pressed and explicitly output the orientation of the controller in certain states. These states were: lying flat on a surface with the top facing up, lying flat on a surface with the bottom facing up, having the lightbar pointing up, having the lightbar pointing down, tilting the controller to the left, and tilting the controller right. From there, we were to create code that broke, or ended, the program once the triangle button on the controller had been pressed.

Design

The first step I took to solve the problem of outputting the orientation of the DualShock controller was to first utilize the acceleration magnitude and numButtons functions from the previous lab. While acceleration was not being directly outputted in the lab, it was necessary to act as a restraint for measuring and ultimately outputting data regarding orientation. Next, I defined a function specifically for orientation, assigning it to a char, or character. Within the function, I had it acting on three parameters:

1) The value of the tolerance (essentially the amount +- the value of the orientation in space we were willing to accept in order to output the data), assigned to a double.
2) The target value, or the value we want both the x, y, and z orientations to be to make a successful, accurate measurement for the remainder of the program, assigned to an integer.
3) The input from the x, y, and z gyroscope sensors within the controller, each separately assigned to a double.

The next function I created was titled trianglePress, and its purpose was to assign the boolean 1, or true, value to the state in which the triangle button was being pressed on the controller. Once pressed, a break was inserted to end the program. My third function was called "close_to" which essentially tests to see if the data (namely, the absolute value of the point minus the value) collected is strictly less than (<) the tolerance range of 0.01. This was done to ensure maximum precision in measuring the acceleration and gyroscope values for the lab procedure.

RESULT:
Demo'd to TA 9/27/2022

Testing

Unlike my previous lab, I verified my results in this lab by running my program multiple times, and modifying the logic each time to see if I could write it in a more concise way. For instance, I changed the tolerance several times until I settled on the final value, which wasn't too small or close to zero such that a measurement wouldn't be made.

Comments

My greatest mistake for this lab was almost accidentally running the lab without writing code to prevent the program from repeating the previous orientation. In the future, this could be useful if I need to print out unique line of output for a given program. I also had a little difficulty at first making sense of the orientation outputs within my if statements, accidentally assigning RIGHT to LEFT and vis versa. This could have been avoided by more thoroughly reading through the instructions

```
/*-------------------------------------------------------------------
-----------
-                                    SE/CprE 185 Lab 04
-             Developed for 185-Rursch by T.Tran and K.Wang
Ethan Griesman
CPR E 185 Section 6
9/25/2022
--------------------------------------------------------------------
---------*/

/*-------------------------------------------------------------------
-----------
-                                Includes
--------------------------------------------------------------------
---------*/
#include <stdio.h>
#include <math.h>


/*-------------------------------------------------------------------
-----------
-                                Defines
--------------------------------------------------------------------
---------*/


int close_to(double tolerance, double point, double value);
double mag(double x, double y, double z);
int numButtons(int b1, int b2, int b3, int b4);
char orientation(double tolerance, int target, double gx, double gy,
double gz);
int trianglePress(int press);

/*-------------------------------------------------------------------
-----------
-                                        Implementation
--------------------------------------------------------------------
---------*/
int main(void) {
    int t, b1, b2, b3, b4;
    double ax, ay, az, gx, gy, gz;
      char preVal;


    while (1) {
        scanf("%d, %lf, %lf, %lf, %lf, %lf, %lf, %d, %d, %d, %d",
&t, &ax, &ay, &az, &gx, &gy, &gz, &b1, &b2, &b3, &b4 );
```

```
            /* ^ scans for time, acceleration along x, y, and z
axis, orientation along the x, y and z axis, and buttons being
pressed ^ */

        /*printf("Echoing output: %d, %lf, %lf, %lf, %lf, %lf, %lf,
%d, %d, %d, %d \n", t, ax, ay, az, gx, gy, gz, b1, b2, b3, b4); */

            /* ^ prints raw output of time, acceleration xyz,
orientation xyz, and buttons being pressed ^ */

                  if ((close_to(0.012, ax, ay) == 1) &&
(orientation(0.15, 1, gx, gy, gz) == 't')) {
                        if (preVal != 't') {
                        printf("TOP\n");
                        preVal = 't';
                        }
                  }
                  if ((close_to(0.012, ax, ay) == 1) &&
(orientation(0.15, -1, gx, gy, gz) == 'v')) {
                        if (preVal != 'v') {
                        printf("FRONT\n");
                        preVal = 'v';
                        }
                  }
                  if ((close_to(0.012, ax, ay) == 1) &&
(orientation(0.15, 1, gx, gy, gz) == 'r')) {
                        if (preVal != 'r') {
                        printf("BACK\n");
                        preVal = 'r';
                        }
                  }
                  if ((close_to(0.012, ax, ay) == 1) &&
(orientation(0.15, -1, gx, gy, gz) == 'l')) {
                        if (preVal != 'l') {
                        printf("LEFT\n");
                        preVal = 'l';
                        }
                  }
                  if ((close_to(0.012, ax, ay) == 1) &&
(orientation(0.15, 1, gx, gy, gz) == 'f')) {
                        if (preVal != 'f') {
                        printf("RIGHT\n");
                        preVal = 'f';
                        }
                  }
                  if ((close_to(0.012, ax, ay) == 1) &&
(orientation(0.15, -1, gx, gy, gz) == 'a')) {
                        if (preVal != 'a') {
```

```
                        printf("BOTTOM\n");
                        preVal = 'a';
                        }
                  }
                  if (trianglePress(b1) == 1) { /* <--- stops the
program if the triangle button, assigned to b1, is pressed */
                        break;
                  }
    }

    return 0;
}

/* Put your functions here */
double mag(double x, double y, double z){ /* <--- calculates
magnitude of acceleration along x, y, z */
double magg = sqrt((x*x)+(y*y)+(z*z));
      return magg;
}
int close_to(double tolerance, double point, double value) { /* <--
checks to see if (point - value) is less than the tolerance, true or
false */
      if (fabs(point-value) < tolerance) {
            return 1;
      }
      return 0;
}
int trianglePress(int press) {
      if (press == 1) {
            return 1;
      }
      return 0;
}
char orientation(double tolerance, int target, double gx, double gy,
double gz) {
      if (target == 1) {
            if (close_to(tolerance, target, gy) == 1) { /* top +y */
                  return 't';
            }
            if (close_to(tolerance, target, gx) == 1) {
                  return 'f';
            }
            if (close_to(tolerance, target, gz) == 1) { /* +z facing
down, BACK */
                  return 'r';
            }
      }
      if (target == -1) {
            if (close_to(tolerance, target, gy) == 1) {
                  return 'a';
            }
            if (close_to(tolerance, target, gx) == 1) {
                  return 'l';
            }
            if (close_to(tolerance, target, gz) == 1) { /* -z facing
front */
                  return 'v';
            }
      }
}
```