**Lab4.Ogbondah**

**LAB 4**

**SECTION D**

**Chimzim Ogbondah**

**SUBMISSION DATE: 9.25.2018**

**DATE: 9.25.2018**

# Problem

1. The purpose of this lab was to be able to compute a program that would first print out the orientation of the controller based on the inputting values, and then once this was completed alter the coding, so it contained 3 functions, wouldn’t print the same orientation in a row, and wouldn’t print if the controller is moving.

# Analysis

1. The problem for this lab was to run the **./ds4rd.exe -d 054c:05c4 -D DS4\_BT -t -g -b -a command** to be able to see the different values scanned by the controller that define each orientation. Then create a close\_to function that will print out the orientation if the point is within the tolerance (tolerance is defined inside the function). Once this was done the mag function was taken from the previous lab and inserted inside the close to function, so the orientations will only print out when the controller is not moving. Finally, the last function was created to end the code which was done by setting the triangle button equal to one, and if this happens then the code breaks.

# Design

1. Our problem was to compute a program that will print the orientation of the controller, but only if the controller isn’t moving, and it cannot print out the same orientation twice in a row. Also if the triangle button is pressed the code will exit/quit. I used a step by step process by first allowing the code to work and then altering it to the specifications.
   1. Create the close\_to function from the instructions and define it below and above
   2. Run the ./ds4rd.exe command and grab the x, y, z functions for the orientations
   3. Write the close\_to function inside the main function
   4. Now insert the mag function inside the close\_to function and then nest all else if statements inside the if close\_to(mag) so the controller only prints when not moving.
   5. Create a variable called orientation = 0;
   6. And an && to all orientation prints and only allowing it to be true if it’s != that orientation
   7. Then set the orientation inside the call to the current
   8. Finally create a function buttonExt where if the triangle is pressed it quits the game and then insert it inside the main function

Using the basic outline above I met the criteria and correctly printed out the orientation of the controller. To ensure I did this correctly I looked over the instructions form the Lab.

# Testing

1. To make sure the results of the solution were correct, I compiled to make sure I didn’t get any errors, and then I went through the lab instructions to ensure that my code met all the criteria, and then I ran the program to make sure it worked effectively.

# Comments

In doing this Lab I learned about how important it is to understand hoe functions worked because I looked over my code the first time and thought I could simplify it by assigning point and tolerance to a double value and then just put a negative for the ones that were a negative value. This however leads to only the top if statement running. Know that I couldn’t do this would have save me time because in the end I had to revert my code back to the original. 1. I approached it by first defining the functions I wanted to work, then commenting them out and making sure my code worked first and then getting more specific and meeting all the criteria. (after it worked I uncommented my other functions). 2. The data from the ./ds4rd.exe and seeing which defining number inside the x, y, and z tell which one the orientation is. 3. The close\_to function, and that’s because the lab instructions told us to, the mag function and it would allow me to make sure it doesn’t print while the controller is moving. Finally, I added the buttonEXT function because when I typed the regular if statement inside the main function calling for it to quit it wouldn’t work. 4. 0.35 and I picked this value by looking at the highest and lowest points that the defining number reached and then I gave it a little bit of room just to make sure I always got the right orientation.

/\*-----------------------------------------------------------------------------

- SE 185 Lab 04

- Developed for 185-Rursch by T.Tran and K.Wang

- Name: Chimzim Ogbondah

- Section: D

- NetID: ogbondah

- Date: 9/24/ 2018

-----------------------------------------------------------------------------\*/

/\*-----------------------------------------------------------------------------

- Includes

-----------------------------------------------------------------------------\*/

#include <stdio.h>

#include <math.h>

/\*-----------------------------------------------------------------------------

- Defines

-----------------------------------------------------------------------------\*/

#define TRUE 1

#define FALSE 0

#define Up 2

#define BOTTOM 3

#define RIGHT 4

#define DOWN 5

#define TOP 6

#define LEFT 7

/\*-----------------------------------------------------------------------------

- Prototypes

-----------------------------------------------------------------------------\*/

/\*-----------------------------------------------------------------------------

- Implementation

-----------------------------------------------------------------------------\*/

int close\_to**(**double tolerance**,** double point**,** double value**);** /\* defining the functions for later use under the main function \*/

double mag**(**double x**,** double y**,** double z**);** /\* defining the functions for later use under the main function \*/

int buttonExt**(**b1**);** /\* defining the functions for later use under the main function \*/

int main**(**void**)** **{**

int t b1**,** b2**,** b3**,** b4**;**

int orientation **=** 0**;** // setting the orientation to an integer vaule as all of them correspond with a number

double ax**,** ay**,** az**,** gx**,** gy**,** gz**;**

**while** **(**TRUE**)** **{**

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

/\* printf for observing values scanned in from ds4rd.exe, be sure to comment or remove in final program \*/

/\*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); \*/

//printf("At %d ms, the acceleration's magnitude was: %f\n", t, mag(ax, ay, az));

// allows the user to exit the code if triangle is pressed

**if** buttonExt**(**b1**)** **{**

**break;**

**}**

**if** close\_to**(**0.05**,** 0 **,**mag**(**ax**,** ay**,** az**)** **)** **{**

**if** **(** close\_to\_neg**(**0.32**,** **-**1.0**,** gz**)** **&&** orientation **!=** 2 **{**

printf**(**"UP/n"**);**

orientation **=** 2**;** // makes the orientation the current(2) so it cannot print twice in a rown

**}**

**else** **if** **(** close\_to\_neg**(**0.35**,** **-**1.0**,** gy**)** **&&** **(**orientation **!=** 3**)** **)** **{**

printf**(**"BOTTOM\n"**);**

oreintation **=** 3**;** // makes the orientation the current(3) so it cannot print twice in a rown

**}**

**else** **if** **(**close\_to\_neg**(**0.35**,** **-**1.0**,** gx**)** **)** **&&** **(**orientation **!=**4**)** **)**

**{**

printf**(**"RIGHT/n"**);**

oreintation **=** 4**;** // makes the orientation the current(4) so it cannot print twice in a rown

**}**

**else** **if** **(**close\_to**(**0.35**,** 1.0**,** gz**)** **&&** **(**oreintation **!=** 5**)** **)(**5**)** **{**

pritnf**(**"DOWN/n"**);**

oreintation **=** 5**;** // makes the orientation the current(5) so it cannot print twice in a rown

**}**

**else** **if** **(**close\_to**(**0.35**,** 1.0**,** gy**)** **(**orientation **!=** 6 **)** **{**

printf**(**"TOP\n"**);**

oreintation **=** 6**;** // makes the orientation the current(6) so it cannot print twice in a rown

**}**

**else** **if(**close\_to**(**0.35**,** 1.0**,** gz**)** **&&** **(**oreintation **!=** 7**)** **)** **{**

printf**(**"LEFT\n"**);**

oreintation **=** 7**;** // makes the orientation the current(7) so it cannot print twice in a rown

**}**

**}**

**return** 0**;**

**}**

/\*CLose to function which checks to make sure the the point is within the tollerance for each point\*/

int close\_to**(**double toloerance**,** double point**,** double value**)** **{**

double lowerTol **=** point **-** value**;** // assigns the max the point can be

double UpperTol **=** point **+** value**;** //assigns the minimum that the point can be

**if** **(** **(**LowerTol **<=** point**)** **&&** **(**point **<=** UpperTol**)** **)** **{**

//if else checks to see if it is within or not

**return** TRUE**;**

**}**

**else** **{**

**return** FALSE**;**

**}**

/\*Function to calculate the accelerations' magnitude \*/

double mag**(**double x**,** double y**,** double z**)** **{**

//Step 8, uncomment and modify the next line

**return** **(**double**)(**sqrt**(**pow**(**x**,** 2**)** **+** pow**(**y**,** 2**)** **+** pow**(**z**,** 2**)));**

**}**

/\* Function that will allow the use to exit the program at will\*/

int buttonExt**(**int b1**)** **{**

**if** **(**b1 **==** 1**)** **{**

**return** TRUE**;** // if the TRIANGLe BUTTON is pressed the user can exit the program

**}**

**else** **{**

**return** FALSE**;**

**}**