**LAB 4 LAB REPORT**

**LAB 4**

**SECTION 3**

**SUBMITTED BY:**

**DANIEL RIPLEY-BETTS**

**SUBMISSION DATE:**

**2/23/2022**

**Part 1: Basics**

**Problem**

* Program outputs orientation of Dualshock when not moving
* Program includes and uses 3 or more of your own functions
* Write such a function with the following prototype: int close\_to (double tolerance, double point, double value);
* It should return true when value is within tolerance of point. For example, if point is 1.0 and tolerance is 0.25, then close\_to should return true when value is from 0.75 to 1.25, otherwise, it should return false.
* Use close\_to to implement your real number comparisons in your code. In C, 0 is evaluated as false, and any other number is true (Usually we use 1)

**Analysis**

**Design**

**Testing**

**Comments**

**Part 2: Feature 1**

**Problem**

Once your code does the above (displays which side is facing up), modify the while loop to stop and end the program when the user clicks the TRIANGLE BUTTON on the DualShock 4.

**Analysis**

**Design**

**Testing**

**Comments**

**Part 3: Feature 3**

**Problem**

Modify your code so that the program only outputs a new line when its orientation changes. (e.g. “RIGHT” shouldn’t appear multiple times in a row)

**Analysis**

**Design**

**Testing**

**Comments**

**Part 4: Questions**

1. How did you approach the design?
2. What data did you have to read in?
3. What functions did you choose to implement and why?
4. What tolerance values did you pick and how did you decide on them? Be sure to include your answers in the appropriate section of your lab report. If you don’t see where it should go, please put it in the comments section.

**Code**

/\* 185 Lab 3 Template \*/

#include <stdio.h>

#include <math.h>

/\* Put your function prototypes here \*/

int debug() {

return 1;

}

double mag(double x, double y, double z);

int seconds();

int minutes();

int millis();

int close\_enough(double tollerance, double point, double value);

int on\_off(int pressTriangle);

int checkPosition(int newPosition);

//todo test and try making this a conditional oldPosition = (oldPosition == newPosition) ? 1 : oldPosition = newPosition

int main(void) {

/\* DO NOT MODIFY THESE VARIABLE DECLARATIONS \*/

int t, X, square, triangle, circle;

double ax, ay, az, nx, ny, nz;

int position = 20;

/\* This while loop makes your code repeat. Don't get rid of it. \*/

while (on\_off(triangle)) {

scanf("%d,%lf,%lf,%lf,%lf,%lf,%lf,%d,%d,%d,%d", &t, &nx, &ny, &nz, &ax, &ay, &az, &triangle, &circle, &X, &square);

if (close\_enough(.25, 1, ay)) {

position = 1;

if (checkPosition(position)) {//try putting this in one if statement "if (close\_enough(.25, 1, ay) && (checkPosition(position))) {

printf("Top\n");

}

}

bottom: if (close\_enough(.25, -1, ay)) {

position = 2;

if (checkPosition(position)) {

printf("Bottom\n");

}

}

if (close\_enough(.25, 1, ax)) {

position = 3;

if (checkPosition(position)) {

printf("Left\n");

}

}

if (close\_enough(.25, -1, ax)) {

position = 4;

if (checkPosition(position)) {

printf("Right\n");

}

}

if (close\_enough(.25, -1, az)) {

position = 5;

if (checkPosition(position)) {

printf("Front\n");

}

}

if (close\_enough(.25, 1, az)) {

position = 6;

if (checkPosition(position)) {

printf("Back\n");

}

}

on\_off(triangle);

checkPosition(position);

fflush(stdout);

//printf("acceleration magnitude was: %.2lf\n",

// t, mag(ax, ay, az));

/\*double mstoSeconds = t/100.0;

printf("Echoing output: %.3lf, %.4lf, %.4lf, %.4lf\n", mstoSeconds, ax, ay, az); \*/

/\*printf("At %d minutes, %d seconds, and %d milliseconds for a total of %d mililiseconds overall, and it's magnitude of accel was: %lf\n",

minutes(t), seconds(t), millis(t), t, mag(ax,ay,az));\*/

}

return 0;

}

/\* Put your functions here \*/

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

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

}

int seconds(int timeS) {

timeS = timeS / 100;

return timeS % 60;

}

int minutes(int timeM) {

return timeM / 6000;

}

int millis(int time) {

time = time % 100;

return time;

}

int close\_enough(double tollerance, double point, double value) {

double upperLimit = point + tollerance;

double lowerLimit = point - tollerance;

if ((value <= upperLimit) && (value >= lowerLimit)) {

return 1;

}

else {

return 0;

}

}

int on\_off(int pressTriangle) {

return (1 - pressTriangle);

}

int oldPosition = 0;

int checkPosition(int newPosition) {

if (newPosition == oldPosition) {

return 0;

}

else if (newPosition != oldPosition) {

oldPosition = newPosition;

return 1;

}

}