**Conditionals**

**LAB #4**

**SECTION 5**

**SUBMITTED BY:**

**Jackson Collalti**

**SUBMISSION DATE:**

**9-26-2022**

**Problem**

Using a DualShock 4 controller, create a program that can read the controller's orientation when it’s not moving. This will require understanding normalized acceleration (a) and raw output of acceleration (g). A while loop, if statements, similar statements, and functions will be used when creating the solution. The output will be the direction the face of the controller is facing (Up, Down, Left, Right, Towards, Away). Requirements Program outputs orientation of Dualshock when not moving and includes and uses three or more of your functions. Then for features, the loop will end, and the function will stop when the triangle button is pressed. Feature 2 the program only outputs a new line when its orientation changes, meaning no repeat printing. A working program must be demonstrated to a TA when finished.

**Analysis**

Criteria must include three or more functions in the final program that are your own. When the triangle button is pressed, the program must stop running. The orientation must only print out once if the direction hasn’t changed. The orientation should be displayed when the controller isn’t moving. For inputs, gx, gy, and gz are all needed for orientation inputs. To identify if the controller is moving, inputs must be ax, ay, and az. For the while loop, I’ll need the triangle button to be an input. For the outputs, I’ll need print statements for all the directions. Then for functions that don’t display, I’ll need magnitude and a tolerance output for each direction. The only formula needed is for magnitude, which was created in the last lab using the square root of (ax^2 + ay^2 + az^2). I plan on making functions for the different axes (left and right, up and down, forward and away). Then a function for tolerance and a function for magnitude.

**Design**

First, I need to test the control outputs to see how each direction outputs in the variables. Then I create a tolerance function. Using the data I collect, I can decide on tolerance values and what direction is positive and negative. After that, I can begin to write code that identifies the direction the controller is facing. Luckily I can use the same magnitude code from the previous lab, so I just need to copy and paste it. At this point, I’ll need to test the code to see if it works. If it passes, then I’ll add the conditional that if the triangle button is pressed, everything stops. I that I can move on to the final problem, being that a direction is only printed once. To do this, I’ll use a statement saying that repeat values aren’t allowed. I’ll need a variable that says what the previous value was and compare that to the new value.

**Testing**

To test if the program works, I’ll need to test a series of orientations multiple times to see if the output is correct. Most of the testing will be in the orientation output since this is the most likely thing to be wrong. Testing the stoppage using the triangle button is easy to test. It’s either a yes or a no. Where orientation could work but might have some issues regarding outputting some positions wrong, the other focus in testing is the single print feature. I plan on doing multiple tests with it and even varying timings.



**Comments**

In this lab, I learned how to create and use functions. This made me understand inputs and outputs for functions. I learn how to turn data into a program that can complete a task. I also improved my ability to create test code. My original code only had two functions, close\_to and mag. Originally I had a bunch of if statements to print the orientation. I got a working program, and from there, I made functions using the if statements I created to fill the three function criteria. I had issues with my Right function tolerance. It’s the reason I have a separate tolerance called tolerancePass2, and it has a very large value.

1. How did you approach the design?

I approached the design by going in order of complexity. I started off by adding the mag function, then if statements for direction. From then I added tolerance values, and after that, I made sure the print function wouldn’t repeat. Finally, I implemented the loop statement.

2. What data did you have to read in?

The inputs I used were gx, gy, and gy for orientation, then ax, ay, az for magnitude. Close\_to I used a tolerance variable, value, and point variable.

3. What functions did you choose to implement and why?

It was obvious that close\_to would need to be a function because values will constantly be compared. The mag function was already a function in the last lab so that was another obvious choice. Then I turned each axis orientation into a function, this made the most sense to meet the three function requirements.

4. What tolerance values did you pick and how did you decide on them?

I originally overheard the value .02 being discussed, this worked for most cases. I had some inconsistency, so I moved up the tolerance to .03 and I had success with it so I kept it. I had problems with the right orientation so I have a very high tolerance for that orientation.

**Source Code Given**

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

- SE/CprE 185 Lab 04

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

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

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

- Includes

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

#include <stdio.h>

#include <math.h>

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

- Defines

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

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

- Prototypes

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

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

- Implementation

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

int main(void) {

int t, b1, b2, b3, b4;

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

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 )

/\* 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);

/\* It would be wise (mainly save time) if you copy your code to calculate

the magnitude from last week

(lab 3). You will also need to copy your prototypes and functions to the

appropriate sections

in this program. \*/

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

az));

}

return 0;

}

/\* Put your functions here \*/

**Test code with only two functions**

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

- SE/CprE 185 Lab 04

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

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

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

- Includes

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

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

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

- Defines

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

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

- Prototypes

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

int close\_to (double tolerance, double point, double value);

double mag(double ax, double ay, double az);

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

- Implementation

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

int main(void) {

int t, b1, b2, b3, b4;

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

double tolerance, point, value;

int Value;

/\*

Data

Up gy=1

Down gy=-1

Left gx =1

Right gx =-1

Away gz =1

Forward gz =-1

\*/

double mag(double ax, double ay, double az);

while (b1 != 1) {

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

double tolerancePass = 0.03;

double tolerancePass2 = .5;

double toleranceFail = 0.70;

double pointN = -1.0;

double pointP = 1.0;

double pointZ = 0.0;

if(close\_to (toleranceFail, pointZ, gx) == 1){

}

else if((close\_to (tolerancePass, pointP, gx) == 1) && (value != 1)){

printf("Left\n");

value = 1;

}

else if((close\_to (tolerancePass2, pointN, gx) == 1) && (value != 2)){

printf("Right\n");

value = 2;

}

if(close\_to (toleranceFail, pointZ, gy) == 1){

}

else if((close\_to (tolerancePass, pointP, gy) == 1) && (value != 3)){

printf("Up\n");

value = 3;

}

else if((close\_to (tolerancePass, pointN, gy) == 1) && (value != 4)){

printf("Down\n");

value = 4;

}

if(close\_to (toleranceFail, pointZ, gz) == 1){

}

else if((close\_to (tolerancePass2, pointP, gz) == 1) && (value != 5)){

printf("Away\n");

value = 5;

}

else if((close\_to (tolerancePass, pointN, gz) == 1) && (value != 6)){

printf("Towards\n");

value = 6;

}

// 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);

/\*

if(mag(ax, ay, az) >= 0.05){

printf("DualShock4 is moving\n");

}

else{

printf("DualShock4 isn't moving\n");

}

\*/

/\* It would be wise (mainly save time) if you copy your code to calculate the magnitude from last week

(lab 3). You will also need to copy your prototypes and functions to the appropriate sections

in this program. \*/

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

fflush(stdout);

}

return 0;

}

/\* Put your functions here \*/

int close\_to (double tolerance, double point, double value){

if(abs(value - point) >= tolerance){

return 0;

}

if(abs(value - point) <= tolerance){

return 1;

}

}

double mag(double ax, double ay, double az){

double Mag;

Mag = sqrt(ax \* ax + ay \* ay + az \* az);

}

**Working Final Code with 5 functions**

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

- SE/CprE 185 Lab 04

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

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

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

- Includes

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

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

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

- Defines

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

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

- Prototypes

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

int close\_to (double tolerance, double point, double value);

double mag(double ax, double ay, double az);

int directionLR(double gx, double gy, double gz, double value);

int directionUD (double gx, double gy, double gz, double value);

int directionTF (double gx, double gy, double gz, double value);

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

- Implementation

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

int main(void) {

int t, b1, b2, b3, b4;

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

double tolerance, point, value;

int Value;

/\*

Data

Up gy=1

Down gy=-1

Left gx =1

Right gx =-1

Away gz =1

Foward gz =-1

\*/

double mag(double ax, double ay, double az);

while (b1 != 1) {

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

if(directionLR(gx, gy, gz, value)==1&& (value != 1)){

printf("Left\n");

value = 1;

}

else if (directionLR(gx, gy, gz, value)==2 && (value != 2)) {

printf("Right\n");

value = 2;

}

if (directionUD(gx, gy, gz, value)==3 && (value != 3)) {

printf("Up\n");

value =3;

}

else if (directionUD(gx, gy, gz, value)==4 && (value != 4)) {

printf("Down\n");

value= 4;

}

if (directionTF(gx, gy, gz, value)==5 && (value != 5)) {

printf("Away\n");

value = 5;

}

else if (directionTF(gx, gy, gz, value)==6 && (value != 6)) {

printf("Towards\n");

value = 6;

}

// 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);

/\*

if(mag(ax, ay, az) >= 0.05){

printf("DualShock4 is moving\n");

}

else{

printf("DualShock4 isn't moving\n");

}

\*/

/\* It would be wise (mainly save time) if you copy your code to calculate the magnitude from last week

(lab 3). You will also need to copy your prototypes and functions to the appropriate sections

in this program. \*/

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

fflush(stdout);

}

return 0;

}

/\* Put your functions here \*/

int close\_to (double tolerance, double point, double value){

if(abs(value - point) >= tolerance){

return 0;

}

if(abs(value - point) <= tolerance){

return 1;

}

}

double mag(double ax, double ay, double az){

double Mag;

Mag = sqrt(ax \* ax + ay \* ay + az \* az);

}

int directionLR (double gx, double gy, double gz, double value){

double tolerancePass = 0.03;

double tolerancePass2 = .5;

double toleranceFail = 0.70;

double pointN = -1.0;

double pointP = 1.0;

double pointZ = 0.0;

if(close\_to (toleranceFail, pointZ, gx) == 1){

return 0;

}

else if((close\_to (tolerancePass, pointP, gx) == 1) ){

return 1;

}

else if((close\_to (tolerancePass2, pointN, gx) == 1) ){

return 2;

}

}

int directionUD (double gx, double gy, double gz, double value){

double tolerancePass = 0.03;

double tolerancePass2 = .5;

double toleranceFail = 0.70;

double pointN = -1.0;

double pointP = 1.0;

double pointZ = 0.0;

if(close\_to (toleranceFail, pointZ, gy) == 1){

return 0;

}

else if((close\_to (tolerancePass, pointP, gy) == 1) ){

return 3;

}

else if((close\_to (tolerancePass, pointN, gy) == 1)){

return 4;

}

}

int directionTF (double gx, double gy, double gz, double value){

double tolerancePass = 0.03;

double tolerancePass2 = .5;

double toleranceFail = 0.70;

double pointN = -1.0;

double pointP = 1.0;

double pointZ = 0.0;

if(close\_to (toleranceFail, pointZ, gz) == 1){

return 0;

}

else if((close\_to (tolerancePass2, pointP, gz) == 1) ){

return 5;

}

else if((close\_to (tolerancePass, pointN, gz) == 1)){

return 6;

}

}

//Pre-Lab Page