**DS4 Equalizer**

**LAB 7**

**SECTION C**

**Akash Patel**

**SUBMISSION DATE: 3/28/19**

**3/28/19**

# Problem

In this lab the objective was to print a graph of lefts and rights wave depending on the pitch and roll of the controller. The pitch and roll should switch depending on which button is pressed. If the controller or joystick was in the center then it should print 0s.

# Analysis

This program required a lot of case statements and formulas that would show where the controller was at. They kept track of characters and values that were in a certain range. The hardest part of the program was tracking the pitch and roll of the controller and the joystick.

# Design

This lab was skeleton coded and basically set up for us. I just copied and pasted the prototypes and turned them into functions. I had to get values and write equations to get the program to compile and work properly. I first created the functions in the main however I had a lot of difficulty and problems with that so I went with what I have now. I made a true or false statement with the square button to stop the program. I also created pitch and roll functions to set parameters for the pitch and roll. I used -1 and 1 as the parameters, then returned the sin of them. I also had to print 40 characters and blank spaces depending on the pitch and roll so I made a function for that where it would take the number and see whether its less then or greater than one. Then I created the graph line function so it would print chracters L or R depending on the value.

# Testing

At first I had a bunch of debug errors and I was also using the wrong flags so it wouldn’t even run properly. Once I took out the -a it started working.

# Comments

I was unable to correctly modify for it to print the joystick outputs. I tried many different methods but it just wouldn’t work.

Questions:

1. I scaled my values by setting the creating an equation that multiplied the value by 39 and the rad divided by (pi/2). This then gives a value that is a max 39 and min -39, however it gives a decimal value so I have to use int to truncate it to a int value. I did the same for joystick however I also multiplied it by 128 because the threshold is 128.

double xd = 39.0 \* (rad / (PI/2));

return (int)xd;

rad = -rad;

double xd = 39.0 \* ((double)rad / (double)128);

return (int)xd;

1. The graph behavior near its limits starts oscillating according to my program. It also shows the letters all the way to the end of each graph.

# Source Code

<Use NPP Exporter to PASTE source code>

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

- SE 185 Lab 07

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

- Name:

- Section:

- NetID:

- Date:

-

- This file provides the outline for your program

- Please implement the functions given by the prototypes below and

- complete the main function to make the program complete.

- You must implement the functions which are prototyped below exactly

- as they are requested.

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

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

- Includes

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

#include <stdio.h>

#include <math.h>

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

- Defines

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

#define PI 3.141592653589

/\* NO GLOBAL VARIABLES ALLOWED \*/

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

- Prototypes

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

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

PRE: Arguments must point to double variables or int variables as appropriate

This function scans a line of DS4 data, and returns

True when left button is pressed

False Otherwise

POST: it modifies its arguments to return values read from the input line.

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

int read\_input( int\* time,

double\* g\_x, double\* g\_y, double\* g\_z,

int\* button\_T, int\* button\_C, int\* button\_X, int\* button\_S,

int\* l\_joy\_x, int\* l\_joy\_y, int\* r\_joy\_x, int\* r\_joy\_y );

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

PRE: ~(-1.0) <= mag <= ~(1.0)

This function scales the roll/pitch value to fit on the screen.

Input should be capped at either -1.0 or 1.0 before the rest of your

conversion.

POST: -39 <= return value <= 39

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

int scaleMagForScreen(double rad);

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

PRE: -128 <= mag <= 127

This function scales the joystick value to fit on the screen.

POST: -39 <= return value <= 39

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

int scaleJoyForScreen(int rad);

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

PRE: -39 <= number <= 39

Uses print\_chars to graph a number from -39 to 39 on the screen.

You may assume that the screen is 80 characters wide.

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

void graph\_line(int number);

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

PRE: num >= 0

This function prints the character "use" to the screen "num" times

This function is the ONLY place printf is allowed to be used

POST: nothing is returned, but "use" has been printed "num" times

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

void print\_chars(int num, char use);

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

- Implementation

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

int main()

{

double x, y, z; /\* Values of x, y, and z axis\*/

int t; /\* Variable to hold the time value \*/

int b\_Up, b\_Down, b\_Left, b\_Right; /\* Variables to hold the button statuses \*/

int j\_LX, j\_LY, j\_RX, j\_RY; /\* Variables to hold the joystick statuses \*/

int scaled\_pitch, scaled\_roll; /\* Value of the roll/pitch adjusted to fit screen display \*/

int scaled\_joy; /\* Value of joystick adjusted to fit screen display \*/

int current = 0;

int d = 0;

int k = 0;

/\* Put pre-loop preparation code here \*/

int read\_input( int\* time,

double\* g\_x, double\* g\_y, double\* g\_z,

int\* button\_T, int\* button\_C, int\* button\_X, int\* button\_S,

int\* l\_joy\_x, int\* l\_joy\_y, int\* r\_joy\_x, int\* r\_joy\_y ){

scanf("%d, %lf, %lf, %lf, %d, %d, %d, %d, %d, %d, %d, %d", time, g\_x, g\_y, g\_z, button\_T, button\_C, button\_X, button\_S, l\_joy\_x, l\_joy\_y, r\_joy\_x, r\_joy\_y);

if(\*button\_S == 1){

return 1;

}

else{

return 0;

}

}

/\* Calculate and scale for pitch AND roll AND joystick \*/

int scaleMagForScreen(double rad){

double xd = 39.0 \* (rad / (PI/2));

return (int)xd;

}

int scaleJoyForScreen(int rad){

rad = -rad;

double xd = 39.0 \* ((double)rad / (double)128);

return (int)xd;

}

/\* Switch between roll, pitch, and joystick with the up, down, and right button, respectivly \*/

double roll(double g\_x)

{

if(g\_x > 1.0)

{

g\_x = 1.0;

}

else if(g\_x < -1.0)

{

g\_x = -1.0;

}

return asin(g\_x);

}

double pitch(double g\_y)

{

if(g\_y > 1.0)

{

g\_y = 1.0;

}

else if(g\_y < -1.0)

{

g\_y = -1.0;

}

return asin(g\_y);

}

void print\_chars(int num, char use){

int i = 0;

int numChars;

if(num < 0)

{

numChars = 40 + num;

for (i = 0; i < numChars; i++)

{

printf(" ");

}

for(i = num; i < 0; i++)

{

printf("l");

}

}

else if(num > 0)

{

for(i = 0; i < 40; i++)

{

printf(" ");

}

for(i = num; i > 0; i--)

{

printf("%c", use);

}

for(i = 0; i < (40 - num); i++)

{

printf(" ");

}

}

else if(num == 0)

{

for(i = 0; i < 39; i++)

{

printf(" ");

}

printf("0");

}

printf("\n");

}

/\* Output your graph line \*/

void graph\_line(int number, int num){

char r,l;

if(number < 0){

print\_chars(number, 'l');

}

else if(number > 0){

print\_chars(number, 'r');

}

else if(number == 0){

print\_chars(0, '0');

}

}

do

{

/\* Scan a line of input \*/

current = read\_input(&t, &x, &y, &z, &b\_Up, &b\_Right, &b\_Down, &b\_Left, &j\_LX, &j\_LY, &j\_RX, &j\_RY );

if(b\_Up == 1 && d > 200){

d = 0;

if(k==2){

k=0;

}

else{

k += 1;

}

}

d ++;

if (k == 0){

graph\_line(scaleMagForScreen(roll(x)),1);

}

if (k == 1){

graph\_line(scaleMagForScreen(pitch(y)),0);

}

if (k == 2){

graph\_line(scaleJoyForScreen(b\_Left),1);

}

fflush(stdout);

}

while (current == 0); /\* Modify to stop when left button is pressed \*/

return 0;

}

# Screen Shots

<Number the screenshots and paste here. The point of numbering the screenshots is so that you can refer to them during your discussion in the various parts above. Alternatively, you can include the screenshots in-line with the text above as part of your discussion.>

