ROLL & PITCH

LAB 7

SECTION 2

Gavin Monroe

SUBMISSION DATE:

10/21/17

Problem

Problem 1: How did you scale your values? Write an equation and justify it.

rad >= -PI/2 && rad <= PI/2, with this equation we could easily determine if we needed to times the number(78/PI) by the rad to scale said rad. We needed to check if the rad was over or under the specific value to justify multiplying said number. Since there is 80 spots together but 0 takes up one and 0 counts as one we use the number 78. The other equations listed on the part 1 document help with calculating the roll and pitch but doesn't help scale it.

Problem 2: How many degrees does each letter in your graph represent? This is the precision of your graph. As your experiment with the roll and pitch, what do you notice about the graph's behavior near the limits of its values?

After Doing simplistic math and finding limits per right degree which should be same for the left, I found the top limit rad to be 1.5 and I found the lower limit to be 0.001 per increment. 1.5 / 0.001 = 1500, then I divide it by 90 degrees since that's the top limit of rotation we can do. Finding each letter to be 16.66 repeating degrees. I found it to stop at 40 after it gets past the limit of rotation for the controller

Analysis

After figuring out the functions and how it displayed the left and right characters I figured out that –t, -b, -g is very important for the command.

Design

Design was mostly already there I just had to piece it altogether with the code that I had ro place in it. No global vars like it mentioned in the comments

Testing

While testing my code I had a hard time getting the equations to work for the scaling, but after I figured out the equation to do it, it became straight forward. I also ran into problems with greater than and less then since I had them mixed up.

Comments

I had fun it was way more challenging than I thought it was would be.

Source Code

```
// SE 185 lab7.c

//

// This is 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.


#include <stdio.h>

#include <math.h>

#include <string.h>

#define PI 3.141592653589


//NO GLOBAL VARIABLES ALLOWED



//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_line(double* g_x, double* g_y, double* g_z, int* time, int* Button_T, int* Button_X, int*
Button_S, int* Button_C);


// PRE: -1.0 <= x_mag <= 1.0
// This function computes the roll of the DS4 in radians

// if x_mag outside of -1 to 1, treat it as if it were 1 or -1

// POST: -PI/2 <= return value <= PI/2

double roll(double x_mag);


// PRE: -1.0 <= y_mag <= 1.0

// This function computes the pitch of the DS4 in radians

// if y_mag outside of -1 to 1, treat it as if it were 1 or -1

// POST: -PI/2 <= return value <= PI/2

double pitch(double y_mag);



// PRE: -PI/2 <= rad <= PI/2

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

// POST: -39 <= return value <= 39

int scaleRadsForScreen(double rad);


// 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);


int main()
```

```
{

	double x, y, z;				// magnitude values of x, y, and z
	int b_Up, b_Down, b_Left, b_Right, t;	// variables to hold the button statuses
	double roll_rad, pitch_rad;			// value of the roll measured in radians
	int scaled_value;		// value of the roll adjusted to fit screen display

	//insert any beginning needed code here

	do
	{
		fflush(stdout);
		// Get line of input
		scanf("%d, %lf, %lf, %lf, %d, %d, %d, %d", &t, &x, &y, &z, &b_Up, &b_Right, &b_Down,
&b_Left );//Grab Input
		// calculate roll and pitch.  Use the buttons to set the condition for roll and pitch
		roll_rad = roll(x);
		pitch_rad = pitch(y);
		// switch between roll and pitch(up vs. down button)
		if (b_Up == 1){
			scaled_value = scaleRadsForScreen(roll_rad);
		}else if(b_Down == 1){
			scaled_value = scaleRadsForScreen(pitch_rad);
		}
		// Scale your output value
// Output your graph line

if (scaled_value < 0){

        scaled_value = scaled_value * -1;

        int i;

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

                printf(" ");

        }

        print_chars(scaled_value, 'r');

}else if (scaled_value == 0){

        int i;

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

                printf(" ");

        }

        printf("0\n");

}else{

        int i;

        for(i=0; i<=40 - scaled_value; i++){

                printf(" ");

        }

        print_chars(scaled_value, 'l');

}

fflush(stdout);
```

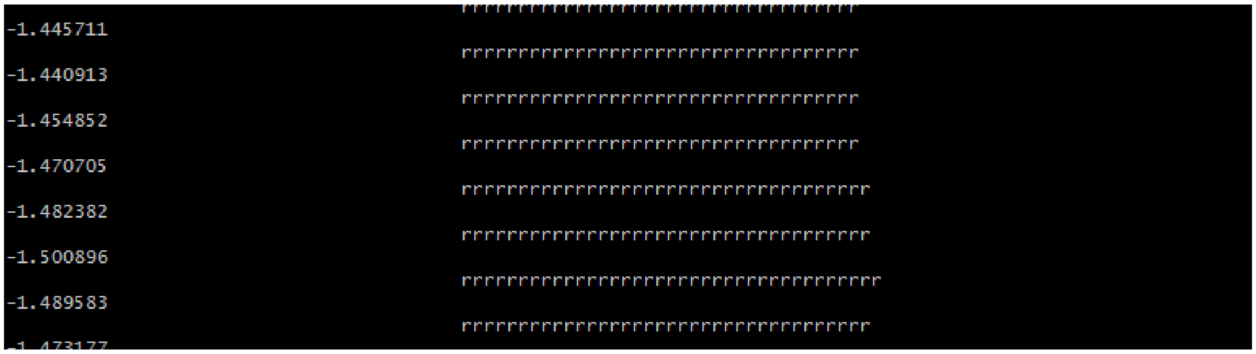
```
	} while (read_line(&x, &y, &z, &t, &b_Up, &b_Down, &b_Left, &b_Right) == 0); // Modify to stop
when left button is pressed

	return 0;

}

int read_line(double* g_x, double* g_y, double* g_z, int* time, int* Button_T, int* Button_X, int*
Button_S, int* Button_C){



	if (*Button_S == 1){

		return 1;

	}else{

		return 0;

	}

}



double roll(double x_mag){

	if (x_mag > 1){

		x_mag = 1;

	}else if(x_mag < -1.0){

		x_mag = -1.0;

	}

	return asin(x_mag);

}

double pitch(double y_mag){

	if (y_mag > 1){

		y_mag = 1;
```

```
        }else if(y_mag < -1.0){

                y_mag = -1.0;

        }

        return asin(y_mag);

}

int scaleRadsForScreen(double rad){

        int num = 78/PI;

        if (rad >= -PI/2 && rad <= PI/2){

                num = num*rad;

        }

        return num;

}

void print_chars(int num, char use){

        int i;

        if (num > 39){

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

                        printf("%c", use);

                }

        }else{

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

                        printf("%c", use);

                }

        }

        printf("\n");
```

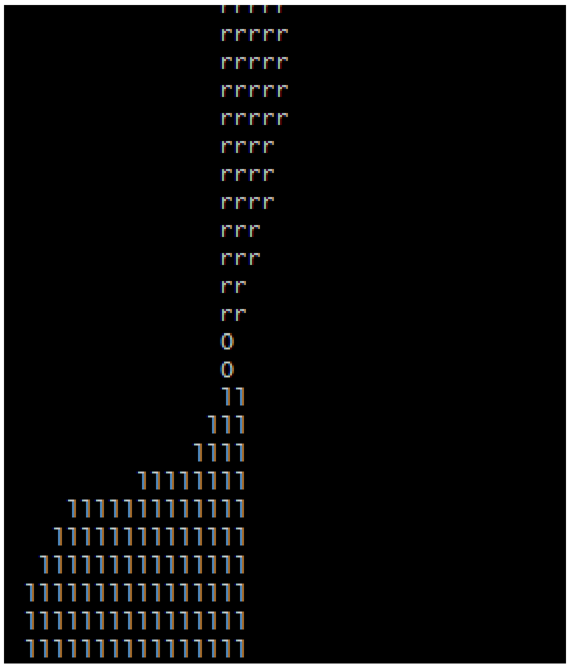
}

Screen Shots

I outputted the rad for roll to find the limits, so I can then find the degrees



```
                                        rrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr
-1.445711
                                        rrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr
-1.440913
                                        rrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr
-1.454852
                                        rrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr
-1.470705
                                        rrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr
-1.482382
                                        rrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr
-1.500896
                                        rrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr
-1.489583
                                        rrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr
-1.473177
```



```
              rrrrr
              rrrrr
              rrrrr
              rrrrr
              rrrrr
              rrrr
              rrrr
              rrrr
              rrr
              rrr
              rr
              rr
              0
              0
              11
             111
            1111
        11111111
   1111111111111
  11111111111111
 111111111111111
1111111111111111
1111111111111111
1111111111111111
```

^ this was the output of my code.