DS4 Equalizer Lab

LAB #7

SECTION #2

Jaden Burke

SUBMISSION DATE:10/18/2022

10/18/2022

Problem

I need to make a program in C given a skeleton program that first read input from a DS4 controller. It will then display a moving bar graph that will print the orientation of the controller based on different variables. The input to the graph function will need to be the scaled value of some input from the controller. The graph will display the roll, or the right and left turn movement, if the triangle button is pressed. It will display the pitch, or the front and back movement, if the cross button is pressed. And I chose to display the left and right movement of the right joystick if the circle button is pressed. It also needs to stop the program if the square button is pressed

Analysis

The first and easiest step is to scan the input for the function. This can be done by finishing the given read_input function. It functions exactly the same as previous labs way of reading the controller data, however I will have to edit the scanf function to both have more values than usual as well as to be in the correct format for pointer input of variables. The next step is displaying the graph. This can be done by finishing the given prototype functions of graph_line and print_chars. The graph function will need to call the print function with some logic, and the print function will just need a simple loop. The scaling can be done pretty easily through some multiplication and division. Which graph to display is as easy as reading the input for which button is pressed and switching a variable value which is then used to call the graph function with the right input. The square button ending the while loop is as easy as setting the condition for the while loop to continue to the output of the read input function, which will be the opposite of the state of the square button.

Design

As stated above, I started off by copy pasting some code from the previous lab's scan function and just adding more variables that needed to be scanned as well as removing the &symbol since the variables that the function receives are already in pointer form. I also had the function return the state of the square button for later use. The next thing I did was create the scaling functions. The magnitude was as easy as just multiplying the value from the controller by 39 and casting it to an integer. The joystick scaling was basically the same thing. However, since it both took and integer input and was not already between -1 and 1 for its base values, I had to cast the input to double, divide by 128, assign that to a holder value, then return that holder value times 39. After that came the print function which was super easy. I just used a for loop to print the given character the right amount of times. The graph function was a little trickier. If the number was positive, I would first print the difference of 39 and the number of spaces, and then the right number of L's. If the number was negative I would print 40 spaces and then the absolute value of the number of R's. Finally if the number was zero I would print 39 spaces and then a 0. The next step was also pretty easy, I just defined a variable and initialized it to 0 so that I could start the program with the same graph every time. I then changed the variable value depending on what button was pressed and ran the right graph accordingly using if else statements. For roll I used x and for pitch I used z for the input into the scale functions. The last part of ending the function when the

square button was pressed was as easy as creating another variable, setting it to the condition of the while loop, and setting it to the opposite of the square button state.

Testing

Testing the function ran mostly as intended. I ran into a small problem near the very beginning stages because I simply called the function in my console as lab7 instead of lab07 however that issue was quickly fixed. I also originally forgot to add a newline after every correct line was printed which caused a lot of formatting issues, but that was also a quick and easy fix. Another problem I had was that originally my joystick scaling just divided the integer by 128 which almost always resulted in 0. This is why I chose to cast it to a double and have a holder variable. The last problem I ran into was because I for some reason did not set the condition for printing 0 as when the inputted value is zero for my graphing function.

Comments

Question 1: I chose to scale my magnitude values by just multiplying by 39. Since I based my rotation on the x value which was already between -1 and 1, I just had to multiply by 39 since 39 * x will always be between 0- 80. The joy stick I first divided by 128 as a double and then did the same thing in order to get it to a value between -1 and 1 first, and then just multiplying by 39.

Question 2: As the graph got towards the limits of its values turns into a curve since if you turn the controller to far, the value the is outputted by the controller start going down from -1 or 1 back towards 0.

Screen Shots

Screenshot 1 of Code:

```c
/*-------------------------------------------------------------------------------
-                              SE 185 Lab 07 - The DS4 Equalizer
-                Developed for 185-Rursch by T.Tran and K.Wang
-   Name:Jaden Burke
-   Section:2
-   NetID:jadenb
-   Date:10/18/22
-
-   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>
#include<stdlib.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);
```

Screenshot 2 of Code:

```c

/*-------------------------------------------------------------------------------
    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 d
    int scaled_joy;                     /* Value of joystick adjusted to fit screen display
    int running;                        //Keeps track of whether while loop should continue
    int whichGraph = 0;                 //keeps track of what graph to run
    /* Put pre-loop preparation code here */

    do
    {
        /* Scan a line of input */
        running = !read_input( &t, &x, &y, &z, &b_Up, &b_Right, &b_Down, &b_Left, &j_LX, &j

        /* Calculate and scale for pitch AND roll AND joystick */
        scaled_roll = scaleMagForScreen(x);
        scaled_pitch = scaleMagForScreen(z);
        scaled_joy = scaleJoyForScreen(j_RX);
        /* Switch between roll, pitch, and joystick with the up, down, and right button, re

        /* Output your graph line */
        if(b_Up){
            whichGraph = 0;
        } else if(b_Down){
            whichGraph = 1;
        } else if(b_Right){
            whichGraph = 2;
        }
        if(whichGraph == 0){
            graph_line(scaled_roll);
        } else if(whichGraph == 1){
            graph_line(scaled_pitch);
        } else if(whichGraph == 2){
            graph_line(scaled_joy);
        }
        fflush(stdout);

    } while (running); /* Modify to stop when left button is pressed */

    return 0;

}
```

Screenshot 3 of Code:

```
126  int read_input( int* time,
127                  double* g_x, double* g_y, double* g_z,
128                  int* button_T, int* button_C, int* button_X, int* button_S,
129                  int* l_joy_x, int* l_joy_y, int* r_joy_x, int* r_joy_y ){
130      scanf("%d, %lf, %lf, %lf, %d, %d, %d, %d, %d, %d, %d, %d", time, g_x, g_y, g_
131      button_C, button_X, button_S, l_joy_x, l_joy_y, r_joy_x,r_joy_y);
132      return *button_S;
133      }
134
135  int scaleMagForScreen(double rad){
136      if(rad >= -1 && rad <= 1){
137          return (int)(rad * 39);
138      } else{
139          return 0;
140      }
141  }
142  int scaleJoyForScreen(int rad){
143      double holder = (double)rad / 128.0;
144      if(holder >= -1 && holder <= 1){
145          return holder * 39;
146      } else{
147          return 0;
148      }
149  }
150  void graph_line(int number){
151
152      if(80 - number > 90){
153          print_chars(40, ' ');
154          print_chars(abs(number), 'R');
155          print_chars(1, '\n');
156      } else if(80 + number == 80 || 80 - number == 80){
157          print_chars(39, ' ');
158          print_chars(1, '0');
159          print_chars(1, '\n');
160      } else{
161          print_chars(39 - number, ' ');
162          print_chars(number, 'L');
163          print_chars(1, '\n');
164      }
165  }
166  void print_chars(int num, char use){
167
168      for(int i = 0; i < num; i++){
169          printf(" %c", use);
170      }
171
172  }
```

Screen Shot of Output: