Lab 2: Solving Simple Problems in C

LAB 2

SECTION C

James Mechikoff

SUBMISSION DATE: 10/23/2018

Date

10/22/2018

Problem

The problem of this lab is to take values given to us from the controller and out put them in real time onto the screen. The values we are seeking are the roll to the right and the left, the pitch front and back, and one axis of the left or right joy stick. It used our understanding of pointers, functions, and print statements during this time.

Analysis

This lab was first going to involve figuring out what needs to be done in what order. Thankfully that order and what needs to be done are given to us by the skeleton code. Next was knowing what to put where. That was pretty simple logical reasoning. The final part as giving the functions that we just placed in order things to do. Scaling the values of pitch and role should be fairly easy however scaling the value of the joy sticks should be a bit more difficult. After that is was simply filling in those scaled values too our graph line function we developed.

Design

The design for this lab was pretty much filling in the functions that were provided in the skeleton code we were supposed to use. Scaling values for the joy sticks, plugging those into a graph line function we were supposed to develop. The scaling for the magnitude was as simple as taking the values times 39. The scaling value for the joy sticks was a bit more complex, which involve running it through a formula we had to develop. The Graph line was simple and was just printing the right amount of characters.

Testing

This problem was much easier to test than past labs. It gave us the formats to use, and it was just up to us to fill in the blanks and develop the functions needed. It involved making one function, testing that function, moving onto the next, testing it, and moving on to the last one, testing it, and in-between all of this, fixing mistakes found.

Comments

This lab was a fun show case of what we have learned so far and leaves things out in the open for what they can do. With more fine tuning, and other applications this could easily be taken much further and possibly applied to things like flight simulations. I love flight simulations, so it gives me a lot of ideas on how to apply these things in the future.

Questions

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

   The scaling for my pitch and roll was fairly simple. I knew the values given were based between 1 and – 1. So I took any input times 39 to get my scaling.

   The scaling for my joy sticks was a bit more complex. I took my value +128 / 255 (The max range for our input values) * 79 (The max range for our output values) – 39.

2. As your experiment with the roll and pitch, what do you notice about the graph's behavior near the limits of its values?

   Things tended to get a bit funny. I noticed I never actually managed to reach the peak no matter how far I pushed it.

Source Code

<Use NPP Exporter to PASTE source code>

```
/*-----------------------------------------------------------------------------
--
-                                       SE 185 Lab 07
-             Developed for 185-Rursch by T.Tran and K.Wang
-     Name:        James Mechikoff
-     Section:     C
-     NetID:             726219551
-     Date:        10/22/2018
-
-   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 roll;
      int tilt;
      int toggle = 0;
      int count=  0;
      int joy;

      int timedifferance =0;

    /* Put pre-loop preparation code here */

    do
    {
        /* Scan a line of input */
            scanf("%d, %lf, %lf, %lf, %d, %d, %d, %d, %d, %d, %d, %d",
                  &t, &x, &y, &z, &b_Up, &b_Right, &b_Down, &b_Left, &j_LX,
&j_LY, &j_RX, &j_RY);

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

            /*if(b_Up == 1){

                  roll = 1;
                  tilt = 0;
                  joy = 0;
```

```
      }else if(b_Down == 1){

            roll = 0;
            tilt = 1;
            joy = 0;

      }else if(b_Right == 1){

            roll = 0;
            tilt = 0;
            joy = 1;

      }*/

if (b_Right==1){

      if(count == 0) {
                  count++;
            toggle++;

      }

}else if(b_Right == 0){

            count = 0;

      }

if (toggle == 1) {

      roll =1;
      tilt =0;
      joy = 0;
      }

      else if (toggle == 2) {

            tilt = 1;
            roll = 0;
            joy = 0;

      }

      else if(toggle == 3) {

            joy = 1;
            roll =0;
            tilt = 0;

      }

      else if(toggle > 3) {

            toggle = 1;
      }
```

```
        /* Output your graph line */

            if( roll == 1){

                  graph_line(scaled_roll);
                  fflush(stdout);

            }else if( tilt == 1){

                  graph_line(scaled_pitch);
                  fflush(stdout);

            }else if( joy == 1){

                  graph_line(scaled_joy);
                  fflush(stdout);

            }


        fflush(stdout);

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

    return 0;

}

int scaleMagForScreen(double rad) {

      return (int)(rad * 39);

}

int scaleJoyForScreen(int rad) {

      if(-128 <= rad && rad < 127){

            if(rad == -128 ){

                  return -39;

            }else if(rad == 0){

                  return 0;

            }else if(rad == 127){

                  return 39;

            }else {

                  return ((rad + 128.0) / (255.0)) * (79.0) - 39.0;

            }
      }
}
```

```
void print_chars(int num, char use) {
      for (int i = 0; i < num; ++i) {
            printf("%c", use);
      }
}

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 );

                        }


void graph_line(int number) {

                              if (number < 0) {

                                    print_chars(39, ' ');
                                    print_chars(-number, 'r');
                                    print_chars(39 + number, ' ');
                                    print_chars(1, '\n');

                              }

                              else if (number > 0) {

                                    print_chars((39 - number), ' ');
                                    print_chars(number, 'l');
                                    print_chars(39, ' ');
                                    print_chars(1, '\n');

                              }
                              else {

                                    print_chars(39, ' ');
                                    print_chars(1, '0');
                                    print_chars(39, ' ');
                                    print_chars(1, '\n');

                              }
                        }
```

Screen Shots