**The DS4 Equalizer**

**LAB # 7**

**SECTION #3**

**James Gaul**

**SUBMISSION DATE: 3/29/24**

**LAB DATE: 3/22/24**

# Problem

For this lab, we were assigned to use pointers to return function values, and to use the terminal display to create a simple visualizer of the DualShock controller’s inputs.

# Analysis

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

* As the gyroscope input returns a floating point value between 1.000 and -1.000, I simply multiplied this value by the max number of characters that could be displayed (39). The equation was scaledValue = value \* 39;
* As the joystick outputs an integer value between positive and negative 127, I needed to divide this input value by 127 in addition to multiplying it by 39 as before. Thus, I used the equation: scaledValue = inputValue \* (39.0 / 127.0);

# Design

I used a simple int variable and switch statement to cycle between input modes when the upper (triangle) button was pressed. To ensure that holding down the button didn’t change the mode continuously, I used a “prevUp” value to record this state, only allowing the mode to change when the up button was pressed and prevUp was also zero.

# Testing

Question 2: As you experiment with the roll and pitch, what do you notice about the graph's

behavior near the limits of its values?

As the graph reaches its limit, it “rolls over” to the opposite extreme, with the line suddenly shifting from “far right” to “far left” and vice-versa.

# Comments

By far, the most difficult task for me was simply getting the scanf statement to properly read the controller data. The large number of variables involved (combined with my own inexperience with using pointers) meant that the largest portion of my lab time was on the read\_input function alone.

# Screen Shots

1. Output - Pitch (forward/backwards):

A screenshot of a computer screen

Description automatically generated

1. Output - Roll (Right/Left):

A screen shot of a computer screen

Description automatically generated

1. Output – Joystick (Right/Left):

A screenshot of a computer

Description automatically generated

**Source Code:**

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

- SE 185 Lab 07 - The DS4 Equalizer

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

- Name: James Gaul

- Section: 3

- NetID: 947125207

- Date: 3/24/24

-

- 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**);**

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

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**,** char indicatorA**,** char indicatorB**);**

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

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\_rx**;** /\* Value of joystick adjusted to fit screen display \*/

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

int quitProg**;**

int inputMode **=** 0**;**

int prevUp**;**

printf**(**"Press square Button to Quit\n"**);**

**do**

**{**

/\* Scan a line of input \*/

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

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

scaled\_pitch **=** scaleMagForScreen**(**y**);**

scaled\_roll **=** scaleMagForScreen**(**x**);**

scaled\_joy\_rx **=** scaleJoyForScreen**(**j\_RX**);**

/\* Switch between roll, pitch, and joystick\*/

**if** **(**b\_Up **==** 1**){**

**if(**prevUp **==** 0**){**

**if(**inputMode **<** 2**){**

inputMode**++;**

**}**

**else{**

inputMode **=** 0**;**

**}**

**}**

prevUp **=** 1**;**

**}**

**else{**

prevUp **=** 0**;**

**}**

/\* Output your graph line \*/

**switch(**inputMode**){**

**case** 0**:**

graph\_line**(**scaled\_pitch**,** 'F'**,** 'B'**);**

**break;**

**case** 1**:**

graph\_line**(**scaled\_roll**,** 'L'**,** 'R'**);**

**break;**

**case** 2**:**

graph\_line**(**scaled\_joy\_rx**,** 'L'**,** 'R'**);**

**break;**

**}**

fflush**(**stdout**);**

**}** **while** **(**quitProg **==** 0 **);**

printf**(**"Goodbye!"**);**

**return** 0**;**

**}**

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\n"**,** 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**;**

**}**

**}**

int scaleMagForScreen**(**double rad**){**

int scale **=** 39**;**

double cappedSize**;**

**if** **(**rad **>** 1.0**){**

cappedSize **=** 1.0**;**

**}**

**else** **if(**rad **<** **-**1.0**){**

cappedSize **=** **-**1.0**;**

**}**

**else{**

cappedSize **=** rad**;**

**}**

double scaledSize **=** cappedSize **\*** scale**;**

**return** round**(**scaledSize**);**

**}**

int scaleJoyForScreen**(**int rad**){**

double scale**;**

**if(**rad **>=** 0**){**

scale **=** **(**39.0 **/** 127.0**);**

**}**

**else** **{**

scale **=** **(**39.0**/**128.0**);**

**}**

**return** rad **\*** scale**;**

**}**

void graph\_line**(**int number**,** char indicatorA**,** char indicatorB**){**

**if(**number **>=** 0**){**

print\_chars**(**40**,** ' '**);**

**if(**number **>** 0**){**

print\_chars**(**number**,** indicatorA**);**

**}**

**else{**

print\_chars**(**1**,** '0'**);**

**}**

**}**

**else{**

print\_chars**(**39 **+** number**,** ' '**);**

print\_chars**(**number **\*** **-**1**,** indicatorB**);**

**}**

print\_chars**(**1**,** '\n'**);**

**}**

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

int i**;**

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

printf**(**"%c"**,** use**);**

**}**

**}**