**Lab 8: The A-Mazing DS4 Race - Part 1**

# **Objectives:**

* Practice Top-Down Program Design, Problem Solving in C
* Develop skills in handling events in a loop
* Introduction to ncurses
* Understand how to process input data into a smoother output (moving averages)

# **Starting Point:**

* [**lab8\_1.c**](https://drive.google.com/open?id=1u1wFCabtLa6_KRsmJwV8gc6DuQqOXW-V)
* [**lab8\_2.c**](https://drive.google.com/open?id=16yox4JvjT2SDGdXzxnOFvuACUIVCC5aX)
* Compiling the lab this week requires the use -lncurses flag

Ex. **gcc -o lab8 lab8.c -lncurses**

# **Turn-in:**

This is a two week lab. **You will submit** **one lab report** at the end of Lab 8 part 2 that includes answers to the all questions in for this week's lab, all your source code that has been checked by your undergraduate TA.

# **Process:**

## Problem

In this and the following lab, you will develop a simple, real-time game controlled by the DualShock 4. The game will be a maze navigation game where you control an avatar using the gyroscope of the DS4. The DS4 will be used to move the avatar left and right on the maze. The avatar will start at the top of the maze, the screen, and at predetermined time intervals, the avatar will move down the maze. Walls in the maze will stop the avatar from moving. The game is won if the avatar reaches the bottom of the maze.

## Goals

This week you will be developing code to put an avatar (your character) on the screen and move around on the screen. Next week, you will work on creating the actual walls of the maze and having your character properly interact with the maze. Week 2 will also include developing game logic.

## Criteria:

* 1. Start your avatar (a single character) at the top center of the screen. A function ( **draw\_character()** ) that lets you write a character to any character location in the window is included in the starting source code for **lab08\_2.c**.  **DO NOT CHANGE THIS FUNCTION.**
  2. At a certain time increment, the avatar will fall one line down the screen. Finding a reasonable interval of time is part of your task. Ensure that the avatar is erased when it moves to a new spot. Your avatar should not leave a trail of old avatar characters as it moves around.
  3. Character movement
     1. If the DualShock 4 is tilted right, the avatar will move one spot to the right.
     2. If the DualShock 4 is titled left, the avatar will move one spot to the left.
     3. This portion of the code will require using a **moving average** (explained below)
  4. The avatar may not move into locations occupied by the maze or off of the screen.
  5. The avatar wins if it makes it to the bottom of the screen without getting stuck. The program will terminate when this condition is met.

## Design:

### Part A Moving Averages:

When dealing with real-time data sources such as the DualShock 4, it is common to want to smooth out rough data. One way to do that is to apply a moving average to the data. We will develop a function that reads from **ds4rd.exe** the x, y, and z values from the gyroscope and computes a moving average on the data in real time.

A moving average of **length n** computes the average **of the last n inputs**. For instance, a moving average of length 2 of using (1, 3, 5, 6, 3) as the input stream is (2, 4, 5.5, 4.5). A moving average of length 3 of the same data would be (3, 4.666, 4.666). Note that the moving average has to accumulate n inputs before it can output something.



For this lab, start with the **lab8\_1.c** source code linked above. This program will provide you with skeleton code for pulling in gyroscope values from the DS4. It will contain a **m\_avg()** function that you have to implement. The function takes an array of doubles, an integer that contains the number of values to average, and a double value to insert into the array. The function will return a double that is the computed average over the updated set of values.

Function Prototype:

**double m\_avg(double buffer[], int avg\_size, double new\_item);**

To implement the function, you will have to shift the values in the array, insert the new value into the array, and perform the computation of the moving average.

The length of the moving average shall be given on the command line to your program (code is included in the sample to help you read this input). Once enough data has been read to output a moving average, a new output should be generated for each input line.

When you are convinced that you have implemented the moving average correctly, run the program with a circular movement of the controller, such as spinning the controller. Capture the output of the program into a CSV file and create three graphs in excel for each axis. **Show these results to your TA before continuing (DEMO)**.

### Part B: Character Movement

Now that you have the moving average completed, you can begin on printing the avatar and moving it around on the screen. Download and open **lab8\_2.c**. Copy and paste the function for the moving average into the sample code. In the sample code, we have included a function that lets you write a character to a specific (x, y) position on the screen. Be sure to use this function (**draw\_character()**). **DO NOT MODIFY THIS FUNCTION**. In a loop, you will make the character begin at the top center of the screen and move downward every so often. Note that if you do this without a delay, the program will complete so fast that you can barely see the avatar move.

To add delay, remember that if the **-t** option is given to **ds4rd.exe**, the first argument is the time in milliseconds since the executable began. Use this data to wait some number of milliseconds (a delay for you to determine), and then implement the avatar moving down the screen with the delay. Keep track of a moving average of the gyroscope as well here so that you can use them to control the avatar.

For the left and right movement, you will need to use the moving averages function along with the gyroscope data to determine if the avatar moves left or right. **No credit will be given if your solution does not make use of the moving average in a meaningful way.** Please note that you need to fill the array with input data before you calculate moving average data, otherwise your moving average will be skewed.

When you have completed the downwards and horizontal mov

nt of the avatar, **demo your functioning program to your TA.**

The program should end when the avatar hits the bottom of the screen.

**Your demo for Part A and Part B must be completed before next week’s lab.**

**Your lab report is not due until after Lab 8 Part 2.**

**Questions and Experiments:**

1. Explain the differences between the raw data and the averaged data in your graph for part A.
2. Explain the delay you used to ensure character movement is not erratic.