TITLE

The A-Mazing DS4 Race

LAB #08

SECTION #08

FULL NAME

Adam Jennissen

SUBMISSION DATE:

11/10/2022

DATE

11/08/2022

Problem

This was a 2 part lab where I had to create a function that would calculate the moving average of the controller, and then apply that to moving a character around a maze. For the maze, I had to come up with a way to move the character around, and a way to randomly generate the maze. I also had to make sure that the character could not move off screen or through any walls.

Analysis

For the moving average I realized that I needed a way to shift the array over by one value, which I used a for loop for. The difference between the raw data and the averaged data is that the average data is a lot smoother, the raw data jumps around a lot, and the averaged data helps keep a smooth line on the graph. I also realized that I would need two nested for loops in order to generate the maze and print it to the screen. For moving the character I realized that I would need to create a timer and then create a blank character in the current location of the character and then create a new character in the new location. I used a delay of 70 ms to move left and right, and a delay of 250 ms to move down. I used these because I wanted the character to be able to move left and right faster than it moved down, and I felt like these were good delays for both.

Design

When I started this lab, I created the moving average function first. I just shifted the array over by one with a for loop, added the new number in, and then averaged the values in the array. I used that function to determine if the character should move left or right. After making the moving average function, I created the timer to move the character down the screen and move it left and right. Once I had the character moving around properly, it was time to create the maze. I started by generating the maze array with two nested for loops and a rand() % 100 – difficulty to determine if there should be a wall at a given spot in the array. After making the array, I created the print function that used two nested for loops to go through the maze array and print a wall anywhere the array had one. After the maze was correctly printing, all I had left was to make sure the player could not move through the walls. I decided to use an if, else if to see if the location the character was trying to move to was going to be a wall or not, by comparing that location to the array. In order to see if the player has lost the game, I had to check if the character had walls on both sides and below them, if they did then they lost.

Testing

When testing this function, I had to run in many times. The first few times I ran it was before the maze was added, and I had to make sure the character moved correctly. I had a small bug that prevented the character from moving sometimes, but after fixing that I moved on to making the maze. Once I had the maze, I had to do a lot of testing to make sure that my difficulty was working correctly, and that my character could not move through walls. One I had all of that working I had to test a handful of times in order to get a maze that I could lose on and after testing my loosing condition a few times, I knew that my program worked correctly.

Comments

In this lab I got a lot more comfortable with making entire programs in C and understanding how all of my functions work together. I also got better at working off of skeleton code, and understanding all of the different functions in C that I can use. This lab was also the first time I used ncurses, so I learned a lot about that. Finally I also learned how to take input during the run command in C.