DS4Maze P1 & P2

LAB 9

SECTION 2

Gavin Monroe

SUBMISSION DATE:

11/15/2017

Problem

The problem was to detect the walls and to stop the player from moving is the walls was blocking said player. I ran into issues of the player replacing the walls but fixed it but moving the code before the movement.

There was no questions for Part 1 So Part 2:

In the "Safe to Go RIGHT/LEFT" conditions and the "Can I fall" conditions, document what is checked and how?

I used the close_to function with findDirection function from my Lab5 Code, I then used the double array MAZE variable to check my left and right before I can actually move. If there was an EMPTY_SPACE char there in the array location then I could move to that location. This is also true for Falling.

Describe what would be necessary to check for the player losing the game and how you would add it to the state machine.

We need to check if the player can go down, left, and right if he can do at least one of these things then we are in the clear. Next if he can do these things and is still stuck by measuring the length of the platform by when the player moves and still is stuck then we lose. I did this by adding the d3 character which tells if he is moving left or right. I change it to Y as yes the player is stuck. This will break the loop and present You lose.

Analysis

I had to make the maze by the percent of walls there were from user input, then set the player, then analysis the players position before moving the player. The goal was to get the player to the end without getting him/her stuck.

Design

Using previous code and modifying it to my needs I made it through this demo with ease I would say. I made sure to only use the variables I needed and that overall made my code smaller than I imagined. I was surprised on how fun this was.

Testing

I had to determine If I needed to use the avg function or just use the mag function with the close to. By the end of it I figured out the more simplistic approach was to use the close_to function to overall detect left and right. Then I just tested for placement of the player,

Comments

I had fun making this little game. I will most likely add more to this later on to help improve where I can.

Source Code

```
// WII-MAZE Skeleton code written by Jason Erbskorn 2007

// Edited for ncurses 2008 Tom Daniels

// Updated for Esplora 2013 TeamRursch185

// Updated for DualShock 4 2016 Rursch



// Headers

#include <stdio.h>

#include <math.h>

#include <ncurses/ncurses.h>

#include <unistd.h>

#include <stdlib.h>

// Mathematical constants

#define PI 3.14159


// Screen geometry

// Use ROWS and COLS for the screen height and width (set by system)

// MAXIMUMS

#define NUMCOLS 100

#define NUMROWS 60


// Character definitions taken from the ASCII table

#define AVATAR 'A'

#define WALL '*'

#define EMPTY_SPACE ' '


// Number of samples taken to form an average for the gyroscope data

// Feel free to tweak this.  You may actually want to use the moving averages
```

```
// code you created last week
#define NUM_SAMPLES 10
char d3 = 'N';
double gStore;
int done = 0;
void findDirection(double gxInput, double gyInput, double gzInput, double tole);
int close_to(double tole, double point, double val);
// 2D character array which the maze is mapped into
char MAZE[NUMROWS][NUMCOLS];



// POST: Generates a random maze structure into MAZE[][]
//You will want to use the rand() function and maybe use the output %100.
//You will have to use the argument to the command line to determine how
//difficult the maze is (how many maze characters are on the screen).
void generate_maze(int difficulty);

// PRE: MAZE[][] has been initialized by generate_maze()
// POST: Draws the maze to the screen
void draw_maze(void);

// PRE: 0 < x < COLS, 0 < y < ROWS, 0 < use < 255
// POST: Draws character use to the screen and position x,y
void draw_character(int x, int y, char use);

// PRE: -1.0 < x_mag < 1.0
// POST: Returns tilt magnitude scaled to -1.0 -> 1.0
// You may want to reuse the roll function written in previous labs.
float calc_roll(float x_mag);
```

```
// Main - Run with './ds4rd.exe -t -g -b' piped into STDIN
void main(int argc, char* argv[])
{
	double x[10000], y[10000], z[10000];
	double gx, gy, gz;// magnitude values of x, y, and z
	int px = (NUMCOLS -2) / 2;
	int py = 1;
	int b_Up, b_Down, b_Left, b_Right, t;
	// setup screen
	initscr();
	refresh();

	// Generate and draw the maze, with initial avatar
	int diff = 0;
	if (argc>1) {
		sscanf(argv[1], "%d", &diff );
	}else {
	}
	generate_maze(diff);
	draw_maze();
	draw_character(px, py, AVATAR);
	draw_character(NUMCOLS -2 , NUMROWS -2, 'F');
	// Read gyroscope data to get ready for using moving averages.
	//printf("%d, %lf, %lf, %lf, %d, %d, %d, %d", &t, &x, &y, &z, &b_Up, &b_Right, &b_Down,
&b_Left);
	// Event loop
	do
	{
```

```
		// Read data, update average
		scanf("%d, %lf, %lf, %lf, %d, %d, %d, %d", &t, &gx, &gy, &gz, &b_Up, &b_Right,
&b_Down, &b_Left);
		findDirection(gx, gy, gz, 0);//Find Direction which the controller is facing.
		if ((t % 100) == 0){
			if (d3 == 'L' && MAZE[py][px - 1] == EMPTY_SPACE){
				px -= 1;
				draw_character(px, py, AVATAR);
				draw_character(px + 1, py, EMPTY_SPACE);
			}else if (d3 == 'R' && MAZE[py][px + 1] == EMPTY_SPACE){
				px += 1;
				draw_character(px, py, AVATAR);
				draw_character(px - 1, py, EMPTY_SPACE);
			}
			if (MAZE[py + 1][px] == EMPTY_SPACE){
				py += 1;
				draw_character(px, py - 1, EMPTY_SPACE);
				draw_character(px, py, AVATAR);
			}else{
				if (MAZE[py][px + 1] != EMPTY_SPACE){
					if (MAZE[py][px - 1] != EMPTY_SPACE){
						done = 1;
						d3 = 'Y';
					}
				}
			}
		}
		// Is it time to move?  if so, then move avatar
```

```
        } while((py != NUMROWS -2) && (done != 1)); // Change this to end game at right time


        // Print the win message
        endwin();
        if (d3 == 'Y'){
                printf("YOU LOSE!\n");
        }else{
                printf("YOU WIN!\n");
        }
}



// PRE: 0 < x < COLS, 0 < y < ROWS, 0 < use < 255
// POST: Draws character use to the screen and position x,y
//THIS CODE FUNCTIONS FOR PLACING THE AVATAR AS PROVIDED.
//
//   >>>>DO NOT CHANGE THIS FUNCTION.<<<<
void draw_character(int x, int y, char use)
{
        mvaddch(y,x,use);
        refresh();
}
void findDirection(double gxInput, double gyInput, double gzInput, double tole){
        if ((close_to(.06, 1, gxInput) == 1) && (close_to(.4, gStore, gxInput) != 1)){
                gStore = gxInput;
                d3 = 'L';
        }else if ((close_to(.06, -1, gxInput) == 1) && (close_to(.4, gStore,  gxInput) != 1)){
                gStore = gxInput;
```

```
                d3 = 'R';
        }
}
int close_to(double tole, double point, double val){
        if ((point - tole < val) && (point + tole >val)){
                return 1;
        }else{
                return 0;
        }
}
void generate_maze(int difficulty){
        int i;
        int row;
        int col;

        for(row=0; row<NUMROWS;row++){
                for(col=0; col<NUMCOLS;col++){
                        MAZE[row][col] = EMPTY_SPACE;
                }
        }
        for(row=0; row<NUMROWS;row++){
                for(col=0; col<NUMCOLS;col++){
                        if (difficulty == 0){
                        }else if(difficulty > 0 && difficulty <= 100){
                                int random = rand()%100;
                                if (random > difficulty){
                                        MAZE[row][col] = EMPTY_SPACE;
                                }else{
                                        MAZE[row][col] = WALL;
```

```
                        }
                    }else{
                        MAZE[row][col] = WALL;
                    }
                }
            }
            for(i=0; i<NUMCOLS; i++){
                MAZE[0][i] = '-';
            }
            for(i=0; i<NUMROWS;i++){
                MAZE[i][0] = '|';
            }
            for(i=0; i<NUMCOLS;i++){
                MAZE[NUMROWS -1][i] = '-';
            }
            for(i=0; i<NUMROWS;i++){
                MAZE[i][NUMCOLS-1] = '|';
            }
}
void draw_maze(void){
            int row;
            int col;
            for(row=0; row<NUMROWS;row++){
                for(col=0; col<NUMCOLS;col++){
                    char p = MAZE[row][col];
                    draw_character(col, row, p);
                }
            }
}
```

Screen Shots