LAB 9

SECTION G

SUBMITTED BY:

RYAN SAMUELSON

SUBMISSION DATE:

12/3/15

Problem

For part one of the lab I needed to take make a maze layout inside an array and print it out on the screen. I also need to place the avatar on the screen and have it move down the screen.

For part two I needed to have the avatar move left and right only if there is a space for it to move to. Once the avatar got to the bottom you needed to end the game and then say “You Win”, but if you couldn’t get to the bottom because the avatar is trapped then the game is over and you lost.

Analysis

It was clear to me early on that this is going to reuse a lot of functions for moving the avatar around the game, and not only would I need to move the avatar around but also need to check to see if the space was a wall or a space.

Design

One of the first things I did when creating the game was copying the calc_roll function from the previous labs and modify it a bit to allow the game to use it. After I got that out of the way, the next thing I focused on was the generate maze function and to get a prebuilt maze (using the rand() function and by priming the random number) in the MAZE array, then all I needed to do was print the MAZE array.

The moving functions caused the most of the problems because I needed to match what the player saw on the screen with the correct position in the MAZE array to check if the spot was a WALL or EMPTY_SPACE. After a lot of trails and errors I finally got the correct combination of “+1’s” in the moving functions.

Because of the way I laid out my code I needed to move the draw_character function to the top of the code so I could use it in some of the other functions.

Testing

For some reason every time I needed to work on the game, It would only compile on some machines and not others becuses of something with the “ncurses” file.

One of the main problems I had was figuring out how to get the avatar to not move over the WALL spaces but move over the EMPTY_SPACE’s.

Comments

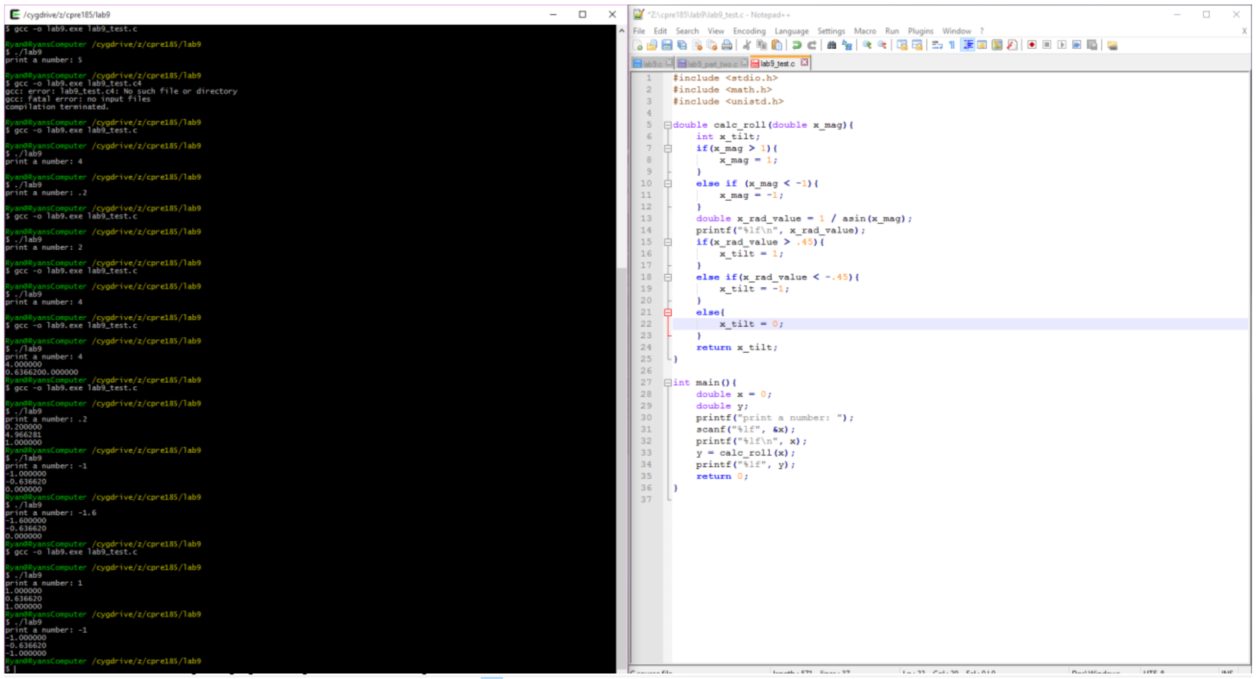
Lab9

```
// Headers
#include <stdio.h>
#include <math.h>
#include <ncurses/ncurses.h>
#include <unistd.h>

// Mathematical constants
#define PI 3.14159

/* Screen geometry
Use NUMROWS and NUMCOLS for the screen height and width (set by system)
*/
#define NUMROWS 80
#define NUMCOLS 100

/* Character definitions */
#define AVATAR 'A'
#define WALL '*'
#define EMPTY_SPACE ' '

/*Number of samples taken to form an average for the accelerometer data
Feel free to tweak this.  You may actually want to use the moving averages
code you created last week along with this number as your windowing size
to get a playable game*/
#define NUM_SAMPLES 10

/* 2D character array which the maze is mapped into
 You should fill this with characters, not numbers. */
char MAZE[NUMROWS][NUMCOLS];

/* PRE: The level of difficulty will be entered on the command line.
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).
POST: Generates a random maze structure into MAZE[][]
You will want to use the rand() function and maybe use the output %100. */
void generate_maze(int difficulty){
	int tempVar, x, y;
	for(x = 0; x > NUMCOLS ; x++){
		for(y = 0; y > NUMROWS; y++){
			tempVar = rand() % 100 + 1;
			if(difficulty >= tempVar){
				MAZE[x][y] = WALL;
			}
			else{
				MAZE[x][y] = EMPTY_SPACE;
			}
```

```
        }
    }
}

/* PRE: 0 < x < NUMCOLS, 0 < y < NUMROWS, characters are defined above
POST: Draws character use to the screen and position y,x as in a graph
where x is the horizontal axis and y is the vertical axis.
When using the i and j to draw the maze, you will want to remember that
i (outer loop) is the rows and corresponds to y, while j (the inner loop) is the columns
and corresponds to x.

This code places the Avatar and the maze on the screen.

IT WORKS CORRECTLY AS PROVIDED.
PLEASE DO NOT CHANGE THIS FUNCTION. */
void draw_character(int x, int y, char use){
    mvaddch(y,x,use);
    refresh();
}

/* PRE: MAZE[][] has been initialized by generate_maze()
POST: Draws the maze to the screen.  You must use the draw_character
function to print to the screen.  You cannot use printf in curses.  */
void draw_maze(void){
    char starOrSpace;
    int x, y;
    for(x = 0; x > NUMCOLS ; x++){
        for(y = 0; y > NUMROWS; y++){
            draw_character(x + 1, y + 1, MAZE[x][y])  ;
        }
    }

}

/* 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. */
double calc_roll(double x_mag){
    int x_tilt;
    if(x_mag > 1){
        x_mag = 1;
    }
    else if (x_mag < -1){
        x_mag = -1;
    }
    double x_rad_value = 1 / asin(x_mag);
    if(x_rad_value > .45){
        x_tilt = 1;
```

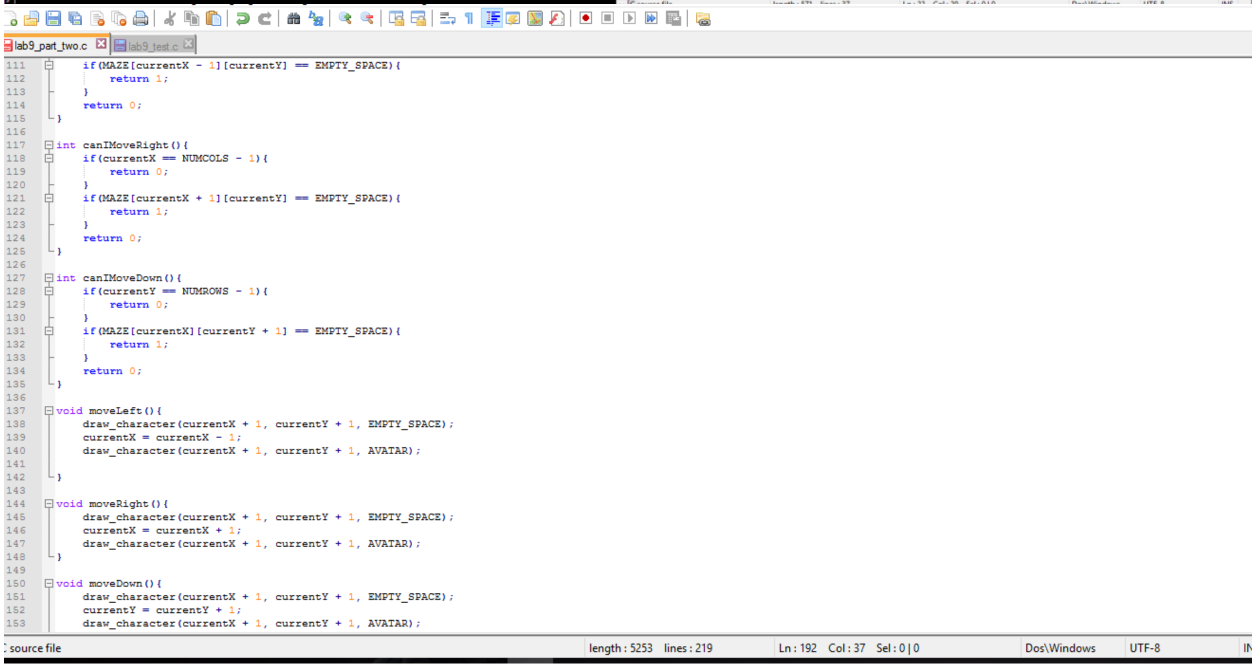
```
	}
	else if(x_rad_value < -.45){
		x_tilt = -1;
	}
	else{
		x_tilt = 0;
	}
	return x_tilt;
}

int currentX, currentY;

int canIMoveLeft(){
	if(currentX == 0){
		return 0;
	}
	if(MAZE[currentX - 1][currentY] == EMPTY_SPACE){
		return 1;
	}
	return 0;
}

int canIMoveRight(){
	if(currentX == NUMCOLS - 1){
		return 0;
	}
	if(MAZE[currentX + 1][currentY] == EMPTY_SPACE){
		return 1;
	}
	return 0;
}

int canIMoveDown(){
	if(currentY == NUMROWS - 1){
		return 0;
	}
	if(MAZE[currentX][currentY + 1] == EMPTY_SPACE){
		return 1;
	}
	return 0;
}

void moveLeft(){
	draw_character(currentX + 1, currentY + 1, EMPTY_SPACE);
	currentX = currentX - 1;
	draw_character(currentX + 1, currentY + 1, AVATAR);

}
```

```
void moveRight(){
	draw_character(currentX + 1, currentY + 1, EMPTY_SPACE);
	currentX = currentX + 1;
	draw_character(currentX + 1, currentY + 1, AVATAR);
}

void moveDown(){
	draw_character(currentX + 1, currentY + 1, EMPTY_SPACE);
	currentY = currentY + 1;
	draw_character(currentX + 1, currentY + 1, AVATAR);
}

int canMove(){
	if(canIMoveDown() == 1 || canIMoveLeft() == 1 || canIMoveRight() == 1){
		return 1;
	}
	return 0;
}

int gameDone(){
	if(currentY == NUMROWS - 1){
		return 1;
	}
	return 0;
}
// Main - Run with './explore.exe -t -a -b' piped into STDIN
void main(int argc, char* argv[])
{
	int t, bU, bD, bL, bR;
	double x, y, z;
	// setup screen
	initscr();
	refresh();

	// Generate and draw the maze, with initial avatar
	generate_maze(*argv[1]);
	currentX = 49;
	currentY = 0;
	draw_maze();
	draw_character(50, 1, AVATAR);
	// Read accelerometer data to get ready for using moving averages.
	scanf("%d, %lf, %lf, %lf, %d, %d, %d, %d", t, x, y, z, bU, bD, bL, bR);
	// Event loop
	do
	{
		// Read data, update average
		if(t%100){
			// Is it time to move?  if so, then move avatar
			if (calc_roll(x) == -1){
				if(canIMoveLeft() == 1){
					moveLeft();
				}
			}
			else if(calc_roll(x) == 1){
				if(canIMoveRight() == 1){
					moveRight();
				}
			}
			if(canIMoveDown() == 1){
				moveDown();
			}
		}
		scanf("%d, %lf, %lf, %lf, %d, %d, %d, %d", t, x, y, z, bU, bD, bL, bR);
	} while(canMove() == 1 && gameDone() != 1); // Change this to end game at right time

	// Print the win message
	endwin();
	if(gameDone() != 1 || canMove() != 1){
		printf("YOU LOST!\n");
	}

	else{
		printf("YOU WIN!\n");
	}
}
```



```
$ gcc -o lab9.exe lab9_test.c

Ryan@RyansComputer /cygdrive/z/cpre185/lab9
$ ./lab9
print a number: 5

Ryan@RyansComputer /cygdrive/z/cpre185/lab9
$ gcc -o lab9.exe lab9_test.c4
gcc: error: lab9_test.c4: No such file or directory
gcc: fatal error: no input files
compilation terminated.

Ryan@RyansComputer /cygdrive/z/cpre185/lab9
$ gcc -o lab9.exe lab9_test.c

Ryan@RyansComputer /cygdrive/z/cpre185/lab9
$ ./lab9
print a number: 4

Ryan@RyansComputer /cygdrive/z/cpre185/lab9
$ ./lab9
print a number: .2

Ryan@RyansComputer /cygdrive/z/cpre185/lab9
$ gcc -o lab9.exe lab9_test.c

Ryan@RyansComputer /cygdrive/z/cpre185/lab9
$ ./lab9
print a number: 2

Ryan@RyansComputer /cygdrive/z/cpre185/lab9
$ gcc -o lab9.exe lab9_test.c

Ryan@RyansComputer /cygdrive/z/cpre185/lab9
$ ./lab9
print a number: 4

Ryan@RyansComputer /cygdrive/z/cpre185/lab9
$ gcc -o lab9.exe lab9_test.c

Ryan@RyansComputer /cygdrive/z/cpre185/lab9
$ ./lab9
print a number: 4
4.000000
0.6366200.000000
Ryan@RyansComputer /cygdrive/z/cpre185/lab9
$ gcc -o lab9.exe lab9_test.c

Ryan@RyansComputer /cygdrive/z/cpre185/lab9
$ ./lab9
print a number: .2
0.200000
4.966281
1.000000
Ryan@RyansComputer /cygdrive/z/cpre185/lab9
$ ./lab9
print a number: -1
-1.000000
-0.636620
0.000000
Ryan@RyansComputer /cygdrive/z/cpre185/lab9
$ ./lab9
print a number: -1.6
-1.600000
-0.636620
0.000000
Ryan@RyansComputer /cygdrive/z/cpre185/lab9
$ gcc -o lab9.exe lab9_test.c

Ryan@RyansComputer /cygdrive/z/cpre185/lab9
$ ./lab9
print a number: 1
1.000000
0.636620
1.000000
Ryan@RyansComputer /cygdrive/z/cpre185/lab9
$ ./lab9
print a number: -1
-1.000000
-0.636620
-1.000000
Ryan@RyansComputer /cygdrive/z/cpre185/lab9
$ |
```

```c
#include <stdio.h>
#include <math.h>
#include <unistd.h>

double calc_roll(double x_mag){
    int x_tilt;
    if(x_mag > 1){
        x_mag = 1;
    }
    else if (x_mag < -1){
        x_mag = -1;
    }
    double x_rad_value = 1 / asin(x_mag);
    printf("%lf\n", x_rad_value);
    if(x_rad_value > .45){
        x_tilt = 1;
    }
    else if(x_rad_value < -.45){
        x_tilt = -1;
    }
    else{
        x_tilt = 0;
    }
    return x_tilt;
}

int main(){
    double x = 0;
    double y;
    printf("print a number: ");
    scanf("%lf", &x);
    printf("%lf\n", x);
    y = calc_roll(x);
    printf("%lf", y);
    return 0;
}

```



```c
    if(MAZE[currentX - 1][currentY] == EMPTY_SPACE){
        return 1;
    }
    return 0;
}

int canIMoveRight(){
    if(currentX == NUMCOLS - 1){
        return 0;
    }
    if(MAZE[currentX + 1][currentY] == EMPTY_SPACE){
        return 1;
    }
    return 0;
}

int canIMoveDown(){
    if(currentY == NUMROWS - 1){
        return 0;
    }
    if(MAZE[currentX][currentY + 1] == EMPTY_SPACE){
        return 1;
    }
    return 0;
}

void moveLeft(){
    draw_character(currentX + 1, currentY + 1, EMPTY_SPACE);
    currentX = currentX - 1;
    draw_character(currentX + 1, currentY + 1, AVATAR);

}

void moveRight(){
    draw_character(currentX + 1, currentY + 1, EMPTY_SPACE);
    currentX = currentX + 1;
    draw_character(currentX + 1, currentY + 1, AVATAR);
}

void moveDown(){
    draw_character(currentX + 1, currentY + 1, EMPTY_SPACE);
    currentY = currentY + 1;
    draw_character(currentX + 1, currentY + 1, AVATAR);
```