THE A-MAZING DS4 RACE

LAB 8 A and 8 B REPORT

ARYAN RAO

SECTION 5

DATE- 11/5/2021

SUBMISSION DATE- 11/9/2021

Problem

Create a real-time game that would be controlled by the DS4 and the game consists of a maze and a character which is to be controlled using the same.

Analysis

The problem consists of various functions defined by us. We must implement mechanisms of ensuring when the character is to be moved left, right, or down.

Design

- Implement the functions
- Check for space to go left, right, or down
- Print the winning and losing messages accordingly

Testing

Made sure that the character moves when it is supposed to, and it doesn’t go beyond the boundaries of the maze. It goes left, right, and down when it is supposed to.

Comments

Scanning the correct information from the DS4 is necessary. It might have an impact on the whole code if not done properly.

Questions

1. Describe how you checked if the avatar could safely move down and go left/right.

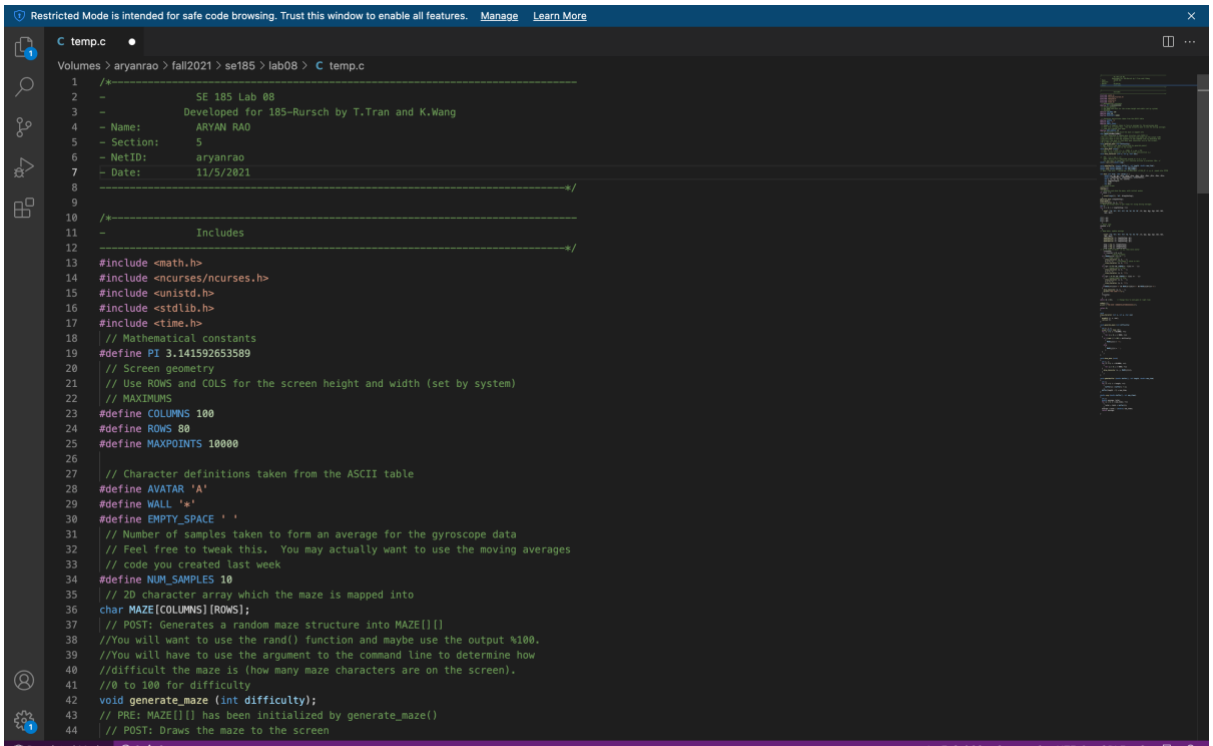
I used the If statements and applied to proper conditions which were if the character encounters the maze or a space, it moves accordingly.

2. Describe what was necessary to check for the player losing the game.

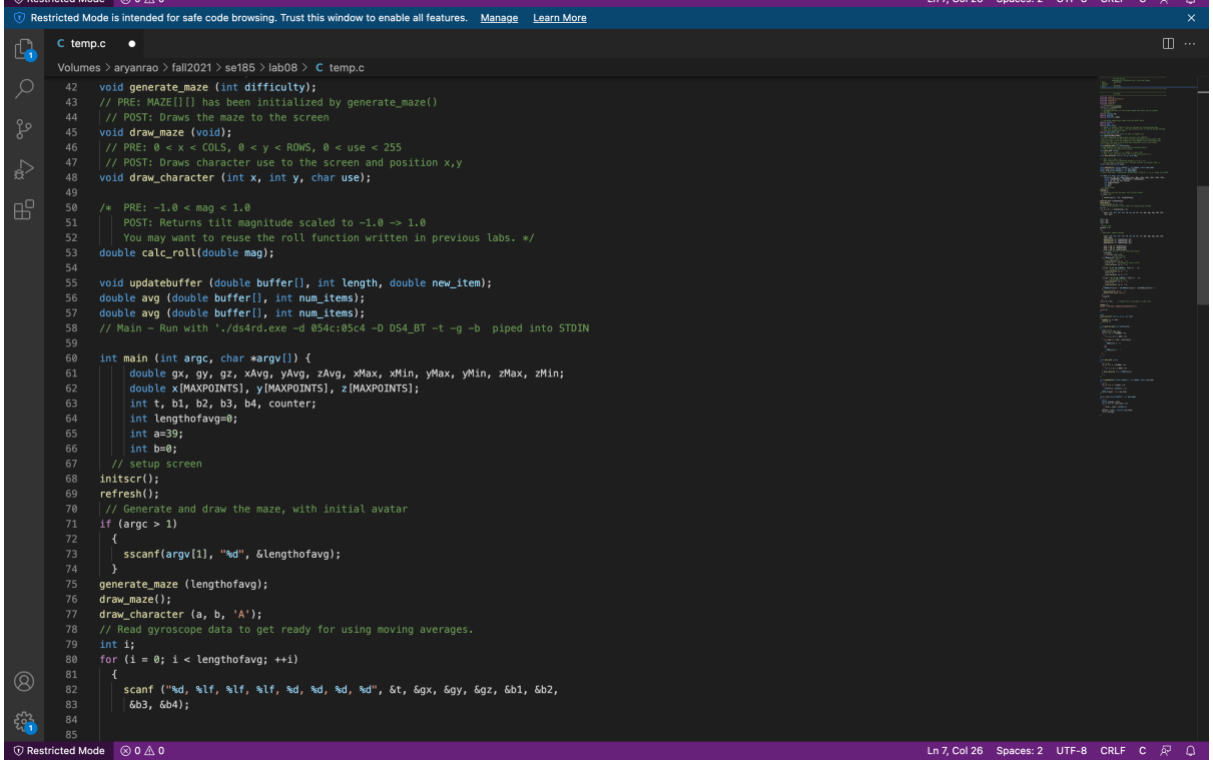
In the If statement, I checked if the avatar encountered a wall, it won’t move hence the user loses the game.

SCREENSHOTS

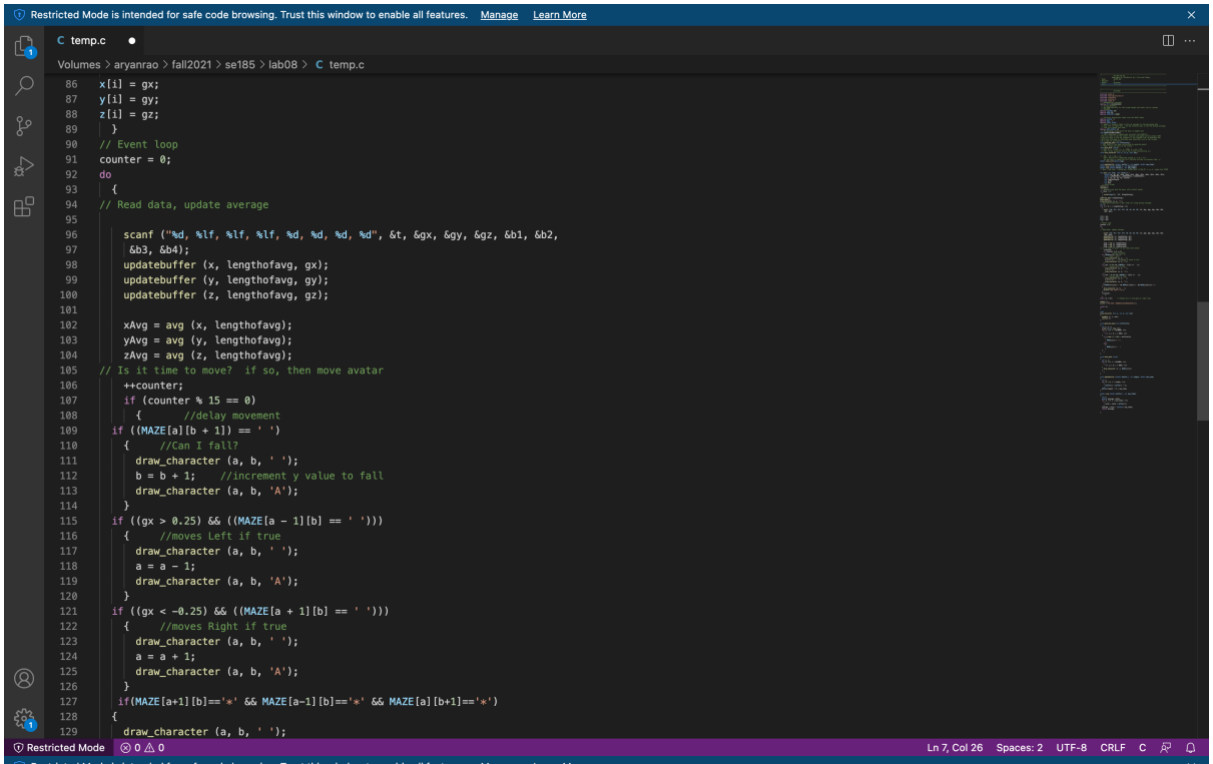
SOURCE CODE:

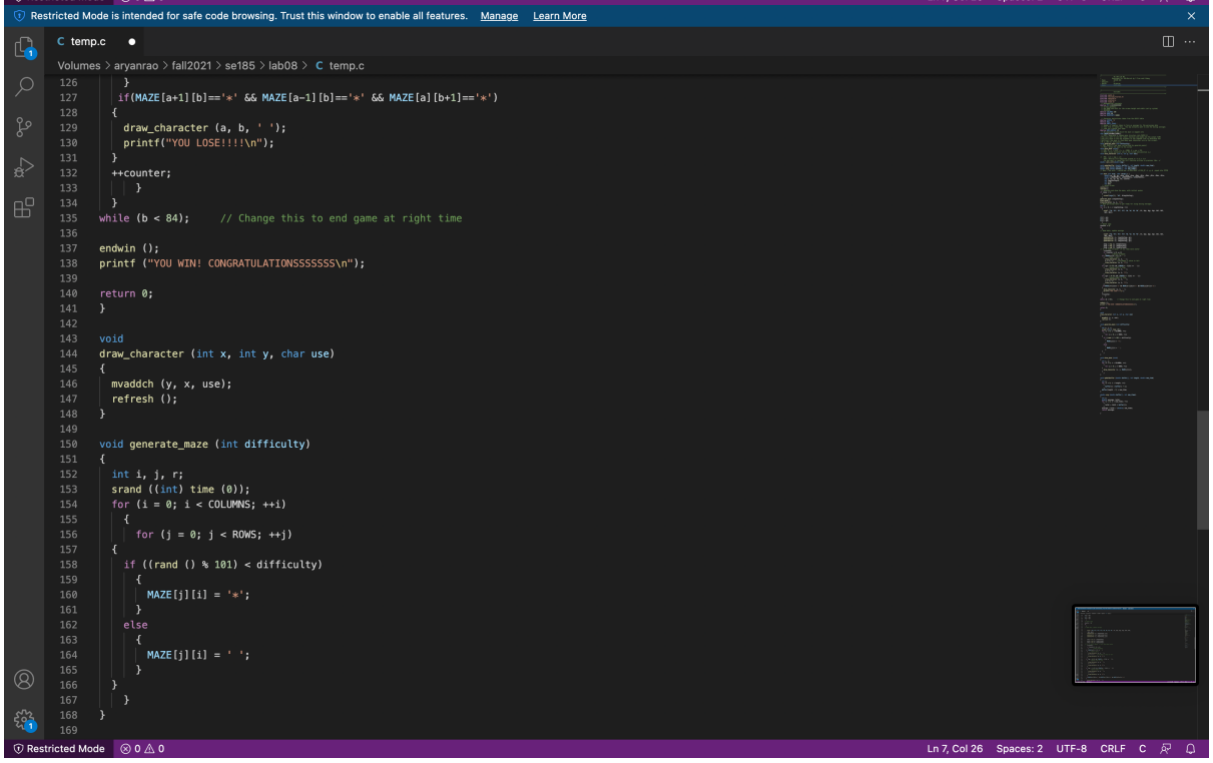


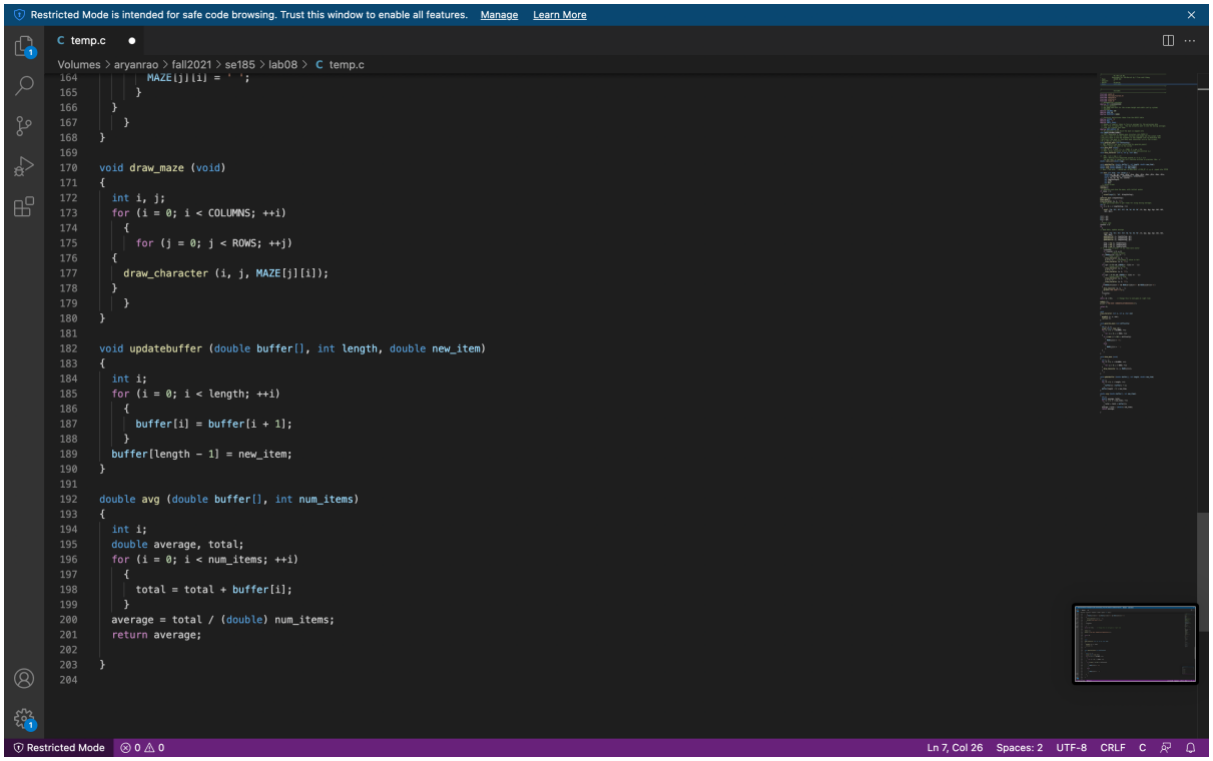
```
/*------------------------------------------------------------------------
-              SE 185 Lab 08
-            Developed for 185-Rursch by T.Tran and K.Wang
- Name:         ARYAN RAO
- Section:      5
- NetID:        aryanrao
- Date:         11/5/2021
------------------------------------------------------------------------*/

/*------------------------------------------------------------------------
-              Includes
------------------------------------------------------------------------*/
#include <math.h>
#include <ncurses/ncurses.h>
#include <unistd.h>
#include <stdlib.h>
#include <time.h>
 // Mathematical constants
#define PI 3.141592653589
 // Screen geometry
 // Use ROWS and COLS for the screen height and width (set by system)
 // MAXIMUMS
#define COLUMNS 100
#define ROWS 80
#define MAXPOINTS 10000

 // 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
 // 2D character array which the maze is mapped into
char MAZE[COLUMNS][ROWS];
 // 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).
//0 to 100 for difficulty
void generate_maze (int difficulty);
// PRE: MAZE[][] has been initialized by generate_maze()
 // POST: Draws the maze to 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 < 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 mag);

void updatebuffer (double buffer[], int length, double new_item);
double avg (double buffer[], int num_items);
double avg (double buffer[], int num_items);
// Main - Run with './ds4rd.exe -d 054c:05c4 -D DS4_BT -t -g -b  piped into STDIN

int main (int argc, char *argv[]) {
    double gx, gy, gz, xAvg, yAvg, zAvg, xMax, xMin, yMax, yMin, zMax, zMin;
    double x[MAXPOINTS], y[MAXPOINTS], z[MAXPOINTS];
    int t, b1, b2, b3, b4, counter;
    int lengthofavg=0;
    int a=39;
    int b=0;
  // setup screen
initscr();
refresh();
 // Generate and draw the maze, with initial avatar
if (argc > 1)
  {
    sscanf(argv[1], "%d", &lengthofavg);
  }
generate_maze (lengthofavg);
draw_maze();
draw_character (a, b, 'A');
// Read gyroscope data to get ready for using moving averages.
int i;
for (i = 0; i < lengthofavg; ++i)
  {
    scanf ("%d, %lf, %lf, %lf, %d, %d, %d, %d", &t, &gx, &gy, &gz, &b1, &b2,
     &b3, &b4);
```


Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
temp.c
Volumes > aryanrao > fall2021 > se185 > lab08 > temp.c
x[i] = gx;
y[i] = gy;
z[i] = gz;
}
// Event loop
counter = 0;
do
{
// Read data, update average
scanf ("%d, %lf, %lf, %lf, %d, %d, %d, %d", &t, &gx, &gy, &gz, &b1, &b2,
&b3, &b4);
updatebuffer (x, lengthofavg, gx);
updatebuffer (y, lengthofavg, gy);
updatebuffer (z, lengthofavg, gz);
xAvg = avg (x, lengthofavg);
yAvg = avg (y, lengthofavg);
zAvg = avg (z, lengthofavg);
// Is it time to move? if so, then move avatar
++counter;
if (counter % 15 == 0)
{ //delay movement
if ((MAZE[a][b + 1]) == ' ')
{ //Can I fall?
draw_character (a, b, ' ');
b = b + 1; //increment y value to fall
draw_character (a, b, 'A');
}
if ((gx > 0.25) && ((MAZE[a - 1][b] == ' ')))
{ //moves Left if true
draw_character (a, b, ' ');
a = a - 1;
draw_character (a, b, 'A');
}
if ((gx < -0.25) && ((MAZE[a + 1][b] == ' ')))
{ //moves Right if true
draw_character (a, b, ' ');
a = a + 1;
draw_character (a, b, 'A');
}
if(MAZE[a+1][b]=='*' && MAZE[a-1][b]=='*' && MAZE[a][b+1]=='*')
{
draw_character (a, b, ' ');
Restricted Mode
0 0
Ln 7, Col 26 Spaces: 2 UTF-8 CRLF C


Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
temp.c
Volumes > aryanrao > fall2021 > se185 > lab08 > temp.c
}
if(MAZE[a+1][b]=='*' && MAZE[a-1][b]=='*' && MAZE[a][b+1]=='*')
{
draw_character (a, b, ' ');
printf("YOU LOSE!!!!\n");
}
++counter;
}
}
while (b < 84); // Change this to end game at right time
endwin ();
printf ("YOU WIN! CONGRATULATIONSSSSSSS\n");
return 0;
}
void
draw_character (int x, int y, char use)
{
mvaddch (y, x, use);
refresh ();
}
void generate_maze (int difficulty)
{
int i, j, r;
srand ((int) time (0));
for (i = 0; i < COLUMNS; ++i)
{
for (j = 0; j < ROWS; ++j)
{
if ((rand () % 101) < difficulty)
{
MAZE[j][i] = '*';
}
else
{
MAZE[j][i] = ' ';
}
}
}
}
Restricted Mode
0 0
Ln 7, Col 26 Spaces: 2 UTF-8 CRLF C


Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
temp.c
Volumes > aryanrao > fall2021 > se185 > lab08 > temp.c
MAZE[j][i] = ' ';
}
}
}
}
void draw_maze (void)
{
int i, j;
for (i = 0; i < COLUMNS; ++i)
{
for (j = 0; j < ROWS; ++j)
{
draw_character (i, j, MAZE[j][i]);
}
}
}
void updatebuffer (double buffer[], int length, double new_item)
{
int i;
for (i = 0; i < length; ++i)
{
buffer[i] = buffer[i + 1];
}
buffer[length - 1] = new_item;
}
double avg (double buffer[], int num_items)
{
int i;
double average, total;
for (i = 0; i < num_items; ++i)
{
total = total + buffer[i];
}
average = total / (double) num_items;
return average;
}
Restricted Mode
0 0
Ln 7, Col 26
Spaces: 2
UTF-8
CRLF
C

OUTPUT: