Mazes and Men

CS 240 Team #3

technical support manual

Dhiraj Ranjit

Austin Bower

DAVE buchanand

2015 Edition

Table of Contents

[Introduction](#Introduction) [3](#_Toc400051163)

[Function Definitions 3](#_Toc400051163)

[Introduction](#Introduction)

**Mazes and Men:**

As one of the group projects for the CS 240 class, Austin, Dhiraj and Dave decided to make a maze game “Mazes and Men” in C programming language using curses library. In this game, a player needs to solve the maze within a certain timeframe to face an enemy with a different power. The player will roll three six-sided dice to fight against the enemy. If the sum of those three dice is greater than or equal to the level of the enemy, the player wins that round and advance to the next level. The player needs to win ten levels to win the game. This game is basically made using structures and 2D arrays. The curses library is really what allows this game to work. The IDE used for this game is Codeblocks and Nano on Raspberry Pi.

Function Definitions

[PrintMaze 4](#_Toc400051158)

[StoreMaze 5](#_Toc400051159)

[PrintEnemy 6](#_Toc400051160)

[StoreEnemy 7](#_Toc400051160)

[MazeTraversal 8](#_Toc400051160)

[Random 9](#_Toc400051160)

[DiceRoll 9](#_Toc400051160)

[Mazecall 10](#_Toc400051160)

[EnemyCall 11](#_Toc400051160)

[StartMenu 12](#_Toc400051158)

PrintMaze

**Prototype:**

void PrintMaze(char array[ ][22], int rows, int cols, WINDOW \*window);

**Parameters:**

Char array[ ][ 22]: A char array where the data had been store from reading the text file

int rows: No. of rows of the array

int cols: No. of the columns of the array

Window \* window A pointer to point to the curses window where the data will be printed

**Description:**

The PrintMaze function is used for printing the maze in the screen. The maze file is read into a 2d array from a text file. The PrintMaze function reads the char array one element at a time and prints the data on curses window. Only 10 maps have been added and implemented since it is just a demo.

StoreMaze

**Prototype:**

void StoreMaze(char array[ ][22], int rows, int cols, FILE \* file);

**Parameters:**

Char array[ ][ 22]: A empty char array for storing data from reading the text file

int rows: No. of rows of the array

int cols: No. of the columns of the array

File \* file A pointer to point to the text file from which data will be read

**Description:**

StoreMaze function is used for storing the maze in the array from the text file. Using counters, the text file is read from top to bottom one character at a time and the data found is inserted into the array. Only 10 maps have been added and implemented since it is just a demo.

PrintEnemy

**Prototype:**

void PrintEnemy(struct Enemy in, int rows, int cols, WINDOW \*window)

**Parameters:**

struct Enemy in: Using a structure for data stored in order to allow multiple values to be stored

int rows: No. of rows of the array

int cols: No. of the columns of the array

WINDOW \*window A pointer to point to the display window created by curses

**Description:**

PrintEnemy function reads the data from the structure which have enemy’s name, power level and graphics array. Then it ouputs on the window created by curses. The function is implemented using three while loops.

StoreEnemy

**Prototype:**

Void StoreEnemy(struct Enemy\* in, int rows, int cols, FILE \*file)

**Parameters:**

struct Enemy in: Using a structure for data stored in named x

int rows: No. of rows of the array

int cols: No. of the columns of the array

File \* file A pointer to point to the text file from which data will be read

**Description:**

StoreEnemy function reads the data from the text file which have enemy’s name, power level and graphics array. Then it inputs all the data in a structure to its respective block. For increasing level of difficulty, the 5 power is added to the enemy.

MazeTraversal

**Prototype:**

int MazeTraversal(char array[][22], int rows, int cols, int timeLimit, WINDOW \*window, int userLevel)

**Parameters:**

Char array[ ][ 22]: A char array where the data had been store from reading the text file.

int rows: No. of rows of the array

int cols: No. of the columns of the array

int timeLimit: Time limit set for a gameplay based user level

WINDOW \*window A pointer to point to the display window created by curses

Int userLevel Int value for the user’s current level

**Description:**

The MazeTraversal function is used to allow the user to traverse through the mazes. It allows the user to press arrow keys and move an ‘X’ through the maze until it reaches the end. A timer keeps track of how long it takes to complete the traversal.

Random

**Prototype:**

int Random()

**Description:**

Random() function is used to generate random values (1-6) for three 6 sided dice.

DiceRoll

**Prototype:**

int DiceRoll(WINDOW \*window, struct Enemy in)

**Parameters:**

WINDOW \*window A pointer to point to the display window created by curses.

struct Enemy in: Using a structure for data stored in named in.

**Description:**

The DiceRoll function uses Random() function to get three different values for the dice and output them in the screen. Also, it checks if the sum of the three values is greater than the enemy level. In case of greater, it prompts user a win message true (1) otherwise it prints the false (0). It returns the value (i.e. either 1 or 0) whenever it’s called in order to break out of the game loop if the user has failed.

MazeCall

**Prototype:**

int MazeCall (WINDOW \*mazeWindow, int mazeRows, int mazeCols, int timeLimit[], int userLevel, char mazeOne[][22], char mazeTwo[][22], ………………………, char mazeNine[][22], char mazeTen[][22])

**Parameters:**

WINDOW \*mazeWindow A pointer to point to the display window created by curses where the maze read from the maze array is been displayed.

int mazeRows: No. of rows of the maze array

int mazeCols: No. of the columns of the maze array

int timeLimit: Time limit set for a gameplay. Default: 20 sec

Int userLevel Int value for the user’s current level

char mazeOne[][22] A char array where the data had been store from reading maze1.txt

char mazeTwo[][22] A char array where the data had been store from reading maze2.txt

**.**

**.**

**.**

char mazeTen[][22] A char array where the data had been store from reading maze10.txt

**Description:**

The MazeCall function is used to determine, based on the user’s level, which maze that MazeTraversal should be called with.

EnemyCall

**Prototype:**

int EnemyCall ( WINDOW \*enemyWindow, int enemyRows, int enemyCols, int userLevel, struct Enemy enemyOne, struct Enemy enemyTwo, ………………….., struct Enemy enemyNine, struct Enemy enemyTen)

**Parameters:**

WINDOW \*enemyWindow A pointer to point to the display window created by curses where the graphics read from the enemy struct is been displayed.

int enemyRows: No. of rows of the enemy array

int enemyCols: No. of the columns of the enemy array

Int userLevel Int value for the user’s current level

struct enemyOne A enemy struct where the data had been store from reading enemy1.txt

struct enemyTwo A enemy struct where the data had been store from reading enemy2.txt

**.**

**.**

**.**

struct enemyTen A enemy struct where the data had been store from reading enemy10.txt

**Description:**

The EnemyCall function is used to determine, based on the user’s level, which enemy should be used to call the DiceRoll function.

StartMenu

**Prototype:**

**void StartMenu(WINDOW \*window)**

**Parameters:**

WINDOW \*window A pointer to point to the display window created by curses.

**Description:**

StartMenu function runs an infinite loop to display the start menu until the user wants to start the game. If the user presses the ‘s’ key, the game starts and if they press the ‘i’ key, the game instructions are shown. After displaying the instructions any key can be pressed to return to the main menu.