CS 108 Project
Maze Game: The Labyrinth

Jigyasa Chouhan

23B1083
Indian Institute of Technology Bombay
April 27, 2024

Contents

1 Introduction to The Labyrinth

The Labyrinth is a classic Maze Game similar to the ones we have all played in newspapers and magazines. The objective of the game is to navigate through the maze from the start point to the end point. The maze is a complex network of paths and walls, and the player must find the correct path to reach the end point while avoiding obstacles and traps. The game requires strategic thinking, problem-solving skills, and quick reflexes.
The Labyrinth is an enhanced version of the above-described maze game. We are supposed to navigate through a maze using keyboard arrow keys to the end point. The twist is, at any given point, we can only view a fraction of the maze around the player, making it all more intriguing!

2 Game Play Overview

As soon as the game begins, the Main Menu of the game is visible which has the options to "Play", "Choose Level", "High Scores" and "Exit".

Figure 1: Main Menu

As the names suggest, clicking the Play button begins the game. Choose level button helps you choose level from Easy, Medium and Hard. High Scores button displays the top 5 High scores made in the game. Exit button will close the game window. The page is shown in 1.
The High Score Page is shown in 2.

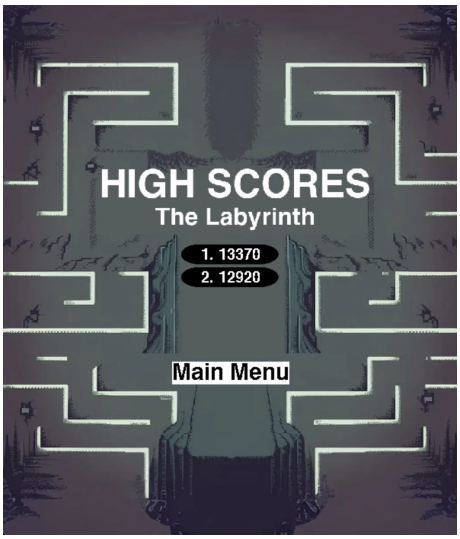


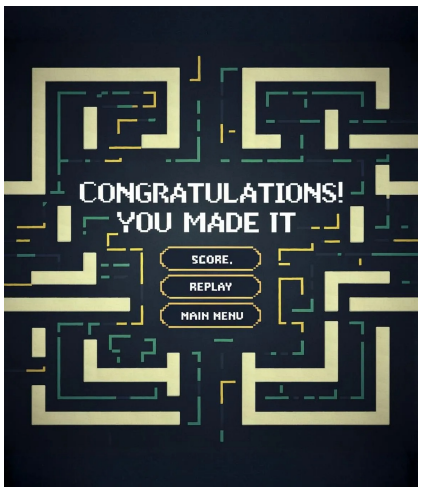
Figure 2: High Score Page

As you can see, in the actual game, the Maze is zoomed in to show a particular part of it on the screen. The player can be moved in the maze using the arrow keys. The player is not capable of walking over the walls of the maze. The gameplay screen is shown in 3 .

Figure 3: Game Play

Every game session has a time limit in which you need to find the way out the maze, the time left and score is displayed at the bottom of the game window.

When a game session ends, you can either win or lose the game i.e. find the way out or not. In both cases a "Game Over" screen appears which displays the result that are shown in 4a and 4b.



(a) Congratulations!

(b) Skill Issue?

3 Basic Idea of the Code

The modules used in developing this game are

- pygame
- random
- sys
- time
- math

The Depth First Search (DFS) algorithm has been used to generate and solve the maze in this game. A new random maze is generated in every new game session. The correct path of corresponding to every maze keeps getting stored in a file named `path.txt`. The file `maze_generator.py` contains the code to generate and solve a maze.
`Scores.py` keeps a track of the top 5 high scores in the game by storing them in the file `scores.txt`.
`collectibles.py` contains the classes of all collectibles available in the maze and methods to regenerate them or use them. `aliens.py` contains the Alien class which is the enemy class that kills the player.
`game.py` is the main function file which calls all the functions and uses pygame

to make the game window. It uses a bunch of booleans to decide which page has to be displayed in the window and which is not.
I used [1] and [2] to understand the basic functionalities and methods of pygame used in making a game.

4 Usage

The usage is only of the `game.py`. Open your terminal, navigate to the directory of the game. In the terminal type:
`python game.py <path_to_music>`

- **python:** This is a message to the terminal, to compile the python file. This may vary according to the operating system used by the user. Some people have to write python3 instead of python.
- **game.py:** This is the file name. The user has to ensure that the present working directory has the required file present.
- **<path_to_music>:** This is a path to a mp3 file to provide customised background music to the game. It is an optional argument, the default music will play if the path is not provided.

Now start playing!

5 Working of The Labyrinth

The detailed working and rules of the game is as follows:

- At the beginning of every game session, a new randomly generated maze is displayed. The dimensions of the maze are 15 x 15 while the area of the maze visible to the user is just 6 x 6.
- The player starts at the centre of the maze. The end point of the game can be at any corner of the maze, randomly.
- The player moves through the maze using keyboard arrow keys. The movement of the player is animated as you can see him running in the various directions as the corresponding arrow keys are pressed.
- Any collisions with the walls of the maze will lead to the player not being able to move in the particular direction.
- A timer starts as soon as a new game session begins, the time left is displayed at the bottom of the screen as soon as the game starts. If the player is not able to reach the end point before the timer goes off, they lose and the game session ends.

- The score is inversely proportional to the amount of time taken by the user to solve the maze, so you can see the score decreasing as the game begins. The score is also displayed at the bottom of the screen.
- There are different kinds of power ups present in the maze which can be collected by making the player go to the cell on which the power up is present.
- **Map:** At the beginning of the game, there will be a map generated which will contain the information about where the end point of the maze is. The map icon is visible partially or fully whenever a new game is started. If the player collects the map, an arrow will be displayed pointing in the direction where the end point of the particular maze is. The map icon is5.

Figure 5: Map icon

- **Boosters:**
  - **Time Booster:** There are time boosters spread throughout the maze collecting which will provide extra 5 seconds to solve the maze.
  - **Score Booster:** There are score boosters spread throughout the maze collecting which will provide 500 extra points to the score.

  The icons for both the boosters are shown in 6a and 6b.

(a) Time Booster

(b) Score Booster

- Whenever a Booster is collected, the user can see the "+5 sec!" or "+500!" being added to the time or score correspondingly, as shown in 7a and 7b

(a) Time Added

(b) Score Added

- **Aliens and Lives:** There are aliens randomly located in the maze that keep blinking throughout the game, the player needs to avoid colliding with the alien when the alien is present/visible. There are also a certain number of lives provided to the player at the start of the game. If the player collides with the alien then the player loses a life. The alien is shown in 8.

Figure 8: Alien

- The number of lives are displayed at the top right of the screen at all times during the game in the form of red hearts, every time a life is lost, a heart is broken and fades away as shown in 9

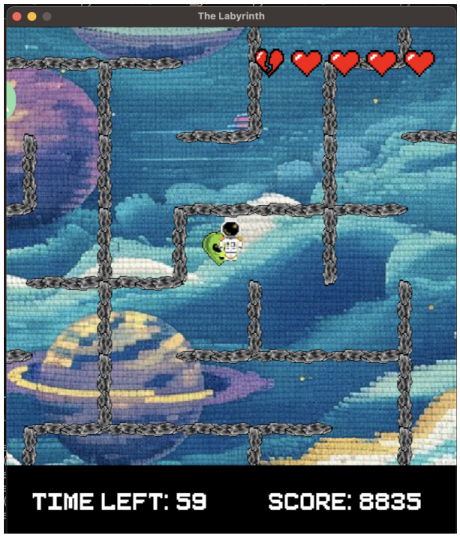


Figure 9: Lives

- If the player loses all the lives, the game ends with only the score of the collected score boosters.
- At the end of the maze, there will be a cell which does not have a wall, open from one side and a rocket at the outside. To win the game, the player needs to reach the rocket. The end point of the maze with the rocket will look like 10.

Figure 10: End point

- There is constant music in the background for a more soothing and exciting gameplay process. The music can be provided while executing the terminal command as well to customise your own game.
- There are three levels in the game which are Easy, Moderate and Hard. The levels can be chosen by clicking the "Choose level" button on the Main Menu which will direct you to the page 11. Click on the level you want to play to select it. By default, easy level is chosen.

Figure 11: Levels Page

- With increasing levels:
  - The time provided to solve the maze reduces.
  - The number of Boosters in the maze reduces.
  - The distance between the map and the player at the start of the game increases.
  - The number of aliens in the maze increase.
  - The number of lives provided reduces.
- After one game session ends, and the user is at either of the Game Over pages, the final score of the game is displayed and is stored in the High Scores if it is in the top 5 scores of the game.
- There is also a "Replay" button which generates a new game session, basically a new maze, of the same difficulty level as chosen before.
- To exit the game, the user can click on the "Exit" button on the Main Menu or just close the pygame window.

6 The Maze Generation Algorithm: Depth First Search Algorithm

The Depth-First Search (DFS) algorithm is a fundamental graph traversal technique used to explore a graph or tree data structure. I have implemented it on a grid of squares to generate a maze. I referred [3] to understand the algorithm

and [4] to understand the implementation using pygame. These are the steps of the algorithm:

- Start at a cell: Begin making paths from the top left cell of the grid.
- Explore Adjacent Cells: Visit one of the unvisited adjacent cells of the current cell. If there are multiple such cells, choose one arbitrarily.
- As you move from one cell to the adjacent, remove the wall between them creating a path.
- Mark Visited: Mark the current cell as visited to avoid revisiting it.
- Recursively Explore: Recursively apply the DFS algorithm to the chosen adjacent cell, repeating steps all adjacent cells are visited.
- Backtrack: If a cell has no unvisited adjacent cell, backtrack to the previous cell where there are unexplored paths.
- Repeat: Continue the process until all cells reachable from the starting cells are visited.

DFS is often implemented using a stack or recursion. It traverses deeply into a graph along one branch before backtracking to explore other branches. This makes it suitable for tasks like finding connected components, cycle detection, and topological sorting.

7 Deep Dive into the code

The code is a collection of a lot of python scripts and classes and functions. These are some prominent features used to make the Maze Game work.

- Booleans are used to direct every page to the other, by changing the values of the booleans, we decide what image is to be displayed on the screens. The booleans are names suggestively, 'levels', 'main_menu', 'game_active' and so on.
- All the images are loaded as soon as the python script is executed, since the pygame.image.load is a time taking function and will hamper the smoothness of the game if we keep uploading images throughout the game.
- The animation of the player while moving in all directions is achieved by using an array of images of the player in different positions of movement which are displayed in a suitable order and speed to create the illusion of continuous movement.
- The `maze_generator.py` using DFS to generate the maze, also has the definition of a class Cell. The class Cell is used to make the entire maze as a list of cells.

- The function generate_maze(corner) generates a maze given a random int as an argument encoding the supposed end point of the maze. It uses DFS and the attributes and methods of the cells in the grid like 'visited' and 'check_neighbours'.
- The function generate_maze(corner) also solves the maze while making it, by keeping a track of its path when it reaches the centre of the maze or the end point of the maze and then to the other end of the path. Then the path is stored accordingly with directions in `path.txt` using the save_path(path) function.
- All the collectibles are a defined as a class with various properties according to their job in the game in `collectibles.py`. All of them have the attribute of their locations to display them at the correct location constantly.
- The boosters only have a draw method to draw them at the correct locations in the maze. The score and time are made functions of the number of respective boosters collected though to serve their purpose in the game.
- The map has a attribute to store the end point of the maze to point in the correct direction when it is collected. The map.point() function uses the information and math module to calculate the angle at which the arrow must be displayed.
- The Alien class is defined in aliens.py and only really stores the alien's locations and a method to draw them on a given surface.
- The blinking of the Alien is achieved by using an array of bools, alien_visibility, that controls if the alien is to be drawn or not for each frame in a second.
- In `Scores.py` there are two functions defined to read the original high scores from scores.txt into an array. And the other function, add_score(score) adds the current score to that array, sorts it in reverse order and pops the last element of the array and writes it back into scores.txt.

8 Precautions and Error Handling

It is advised to not provide an invalid music path in the command line while executing the python command. Though, if no music path is provided, there is a default path provided to not let the inavailability of any audio hamper the user's game experience. If an invalid path is provided, the game will not begin and just end with a message "Invalid music file provided".
To win the game, the player needs to reach somewhere around the centre of the rocket, a little too off might end up the game not being won and the player can end up wandering around in open space. So it is advised to hit the centre of the rocket.

References

[1] Pygame Intro: URL: https://github.com/clear-code-projects/UltimatePygameIntro/blob/main/runner_video.py.

[2] Pygame Official Tutorials: URL: https://www.pygame.org/docs/#tutorials-reference-label.

[3] Maze generation: URL: https://www.youtube.com/watch?v=Ez7U6jU0q5k.

[4] Code Implementation: URL: https://github.com/StanislavPetrovV/Maze_Game/blob/main/main.py.