CS108 Project
Maze Game

Gaurav Kumar
23B0989
Indian Institute of Technology, Bombay

April 28, 2024

Contents

1 Modules and File Structure

1.1 Pre-made modules

I have used these pre-made modules:

- os
- random
- pygame

1.2 My modules

Other than these I created several modules myself to aid my code development process. These are:

- **Player**: In the file *player.py*, the player class is implemented. It draws and animates player movements at a particular location.
- **Settings**: The file *settings.py* contains the game info like tile-size, window-size, timers, lives,etc.
- **Coin**: The file *coins.py* contains the Coin class which simply implements coin drawing and coin collection
- **Hammer**: The file *hammer.py* contains the Hammer class which implements hammer functions
- **Key**: The Key class in file *key.py* implements key.
- **Bomb**: The Bomb class in file *bomb.py* implements bombs.
- **Menu**: The Menu class in *menu.py* uses buttons and draws the menu screen
- **Levels**: The levels class in *levels.py* uses buttons and draws the level screen
- **Pause**: The Pause class in *paused_screen.py* implements the paused screen

1.3 Other files

The other files are

- .high_scores.txt: It stores the top 5 high scores of the player
- path.txt: It stores the solution path for the current maze.

1.4 Other folders

The folders include:

- fonts: To store the required fonts
- images: To store the images
- music: To store music files for backgound music and sound effects

2 Rules, instructions and gameplay

2.1 Rules and Instructions

The game primarily consists of collecting a key and then reaching to the treasure chest.
It has powerups and collectibles in between. The powerups include:

- 🔨 **Hammer**: It can break walls. Press space + arrow key in the direction to break the wall. Only one can be collected at a time ,i.e, no stacking
- 🔍 **Eye**: Displays the whole maze for 1.5s. It is only when and randomly put anywhere.

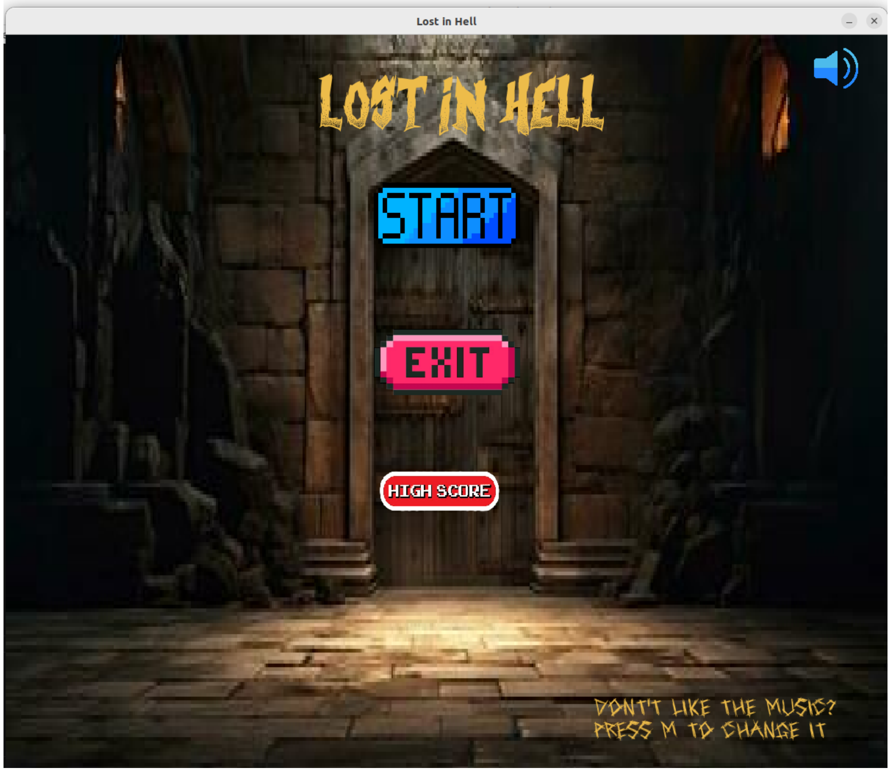
The collectibles are:

- 🪙 **Coins**: Simply collect coins to increase score
- 🗝 **Key**: Key is essential for clearing the maze. There is only one key. First, find the key and and then go to the end.

💣 **Bomb**: You have 3 lives in each state. You have to avoid these bombs as these reduce your lives by 1.

2.2 Gameplay

Upon opening the game, there is a menu screen having buttons for playing the game, quitting and high scores.



DON'T LIKE THE MUSIC?
PRESS M TO CHANGE IT

Figure 1: Menu Screen

Upon clicking start, you get options to choose the level which you want to play. There are 3 levels, in increasing order of difficulty and complexity.

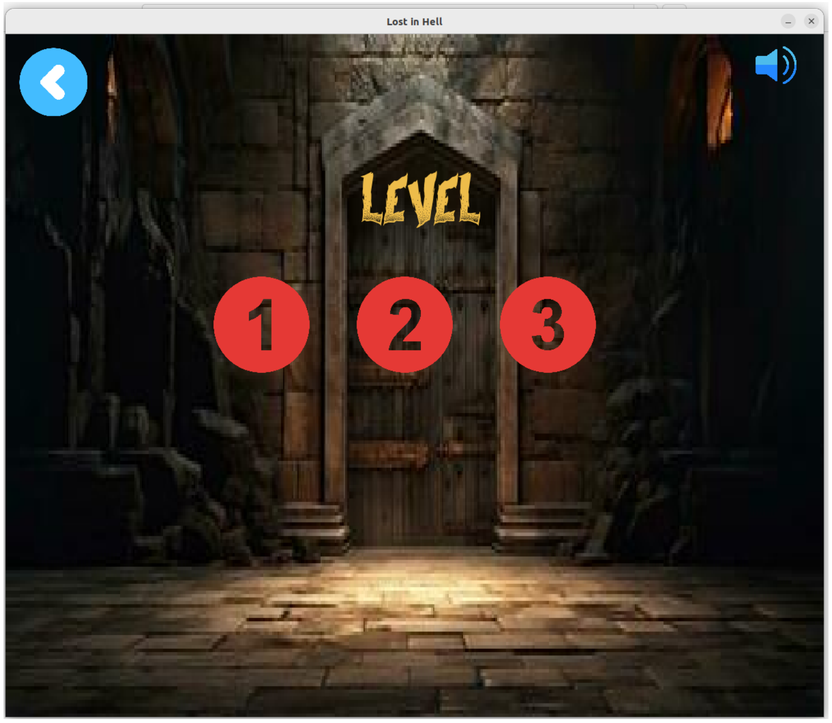


Figure 2: Level Screen

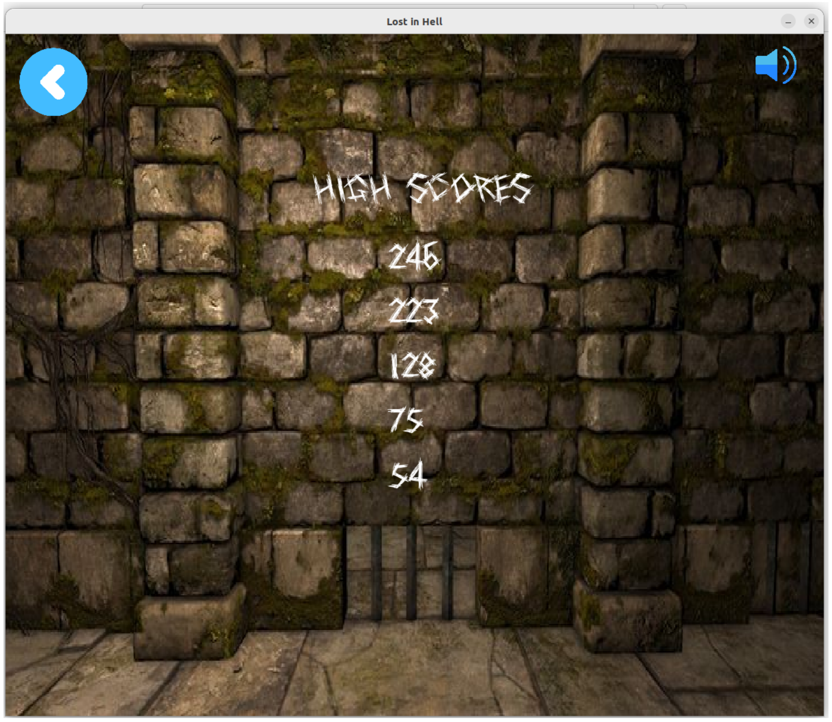


Figure 3: High Scores

The gameplays for 3 levels are shown:

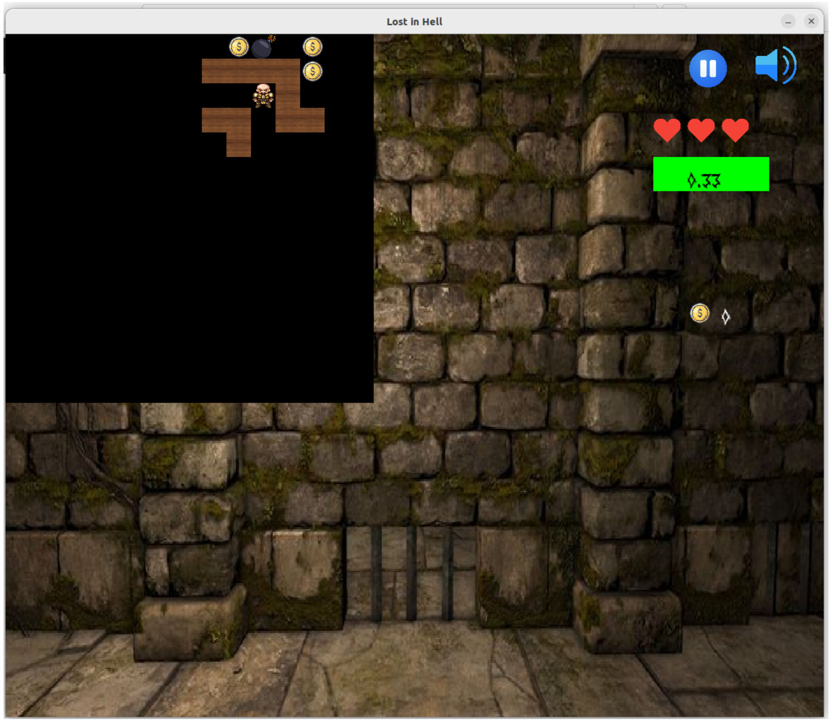


Figure 4: Level 1

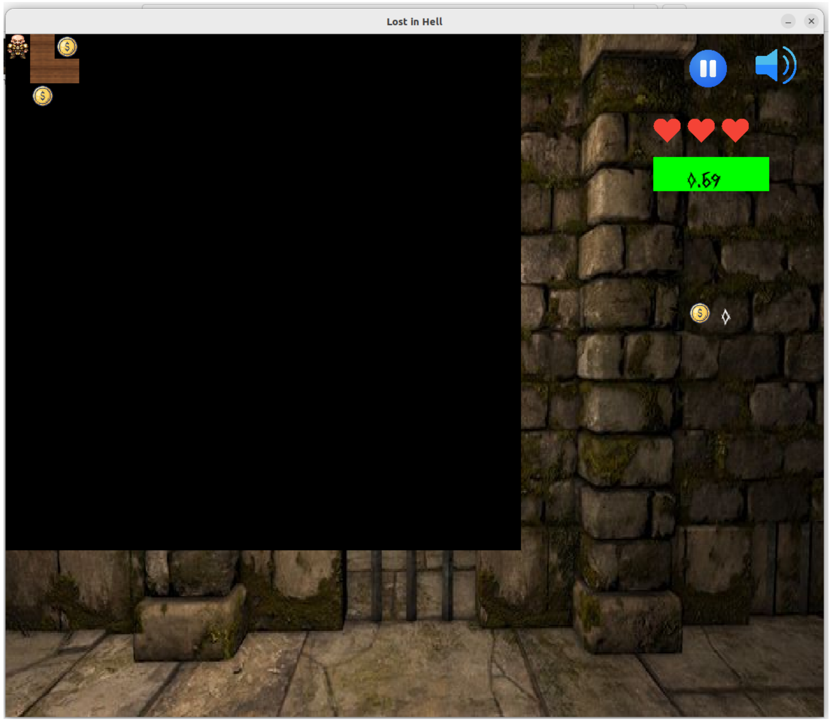


Figure 5: Level 2

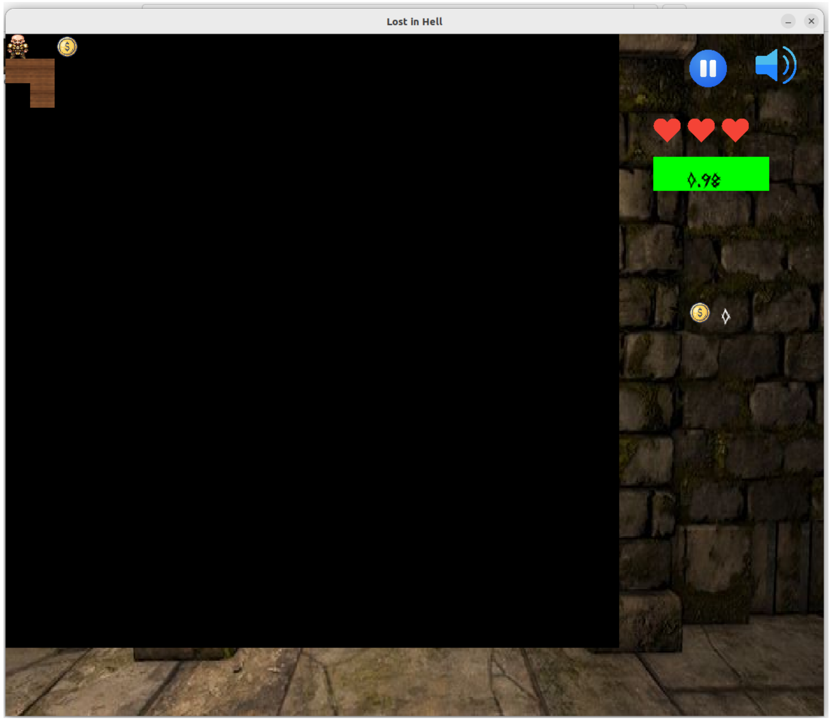
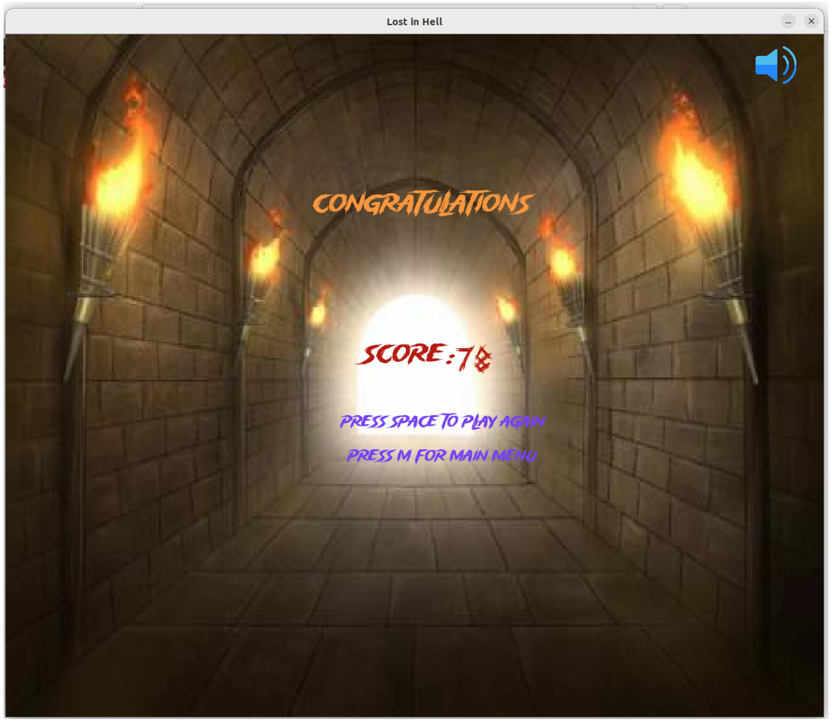


Figure 6: Level 3

The game winning screen



CONGRATULATIONS

SCORE:78

PRESS SPACE TO PLAY AGAIN

PRESS M FOR MAIN MENU

Figure 7: Game won

Game losing screen

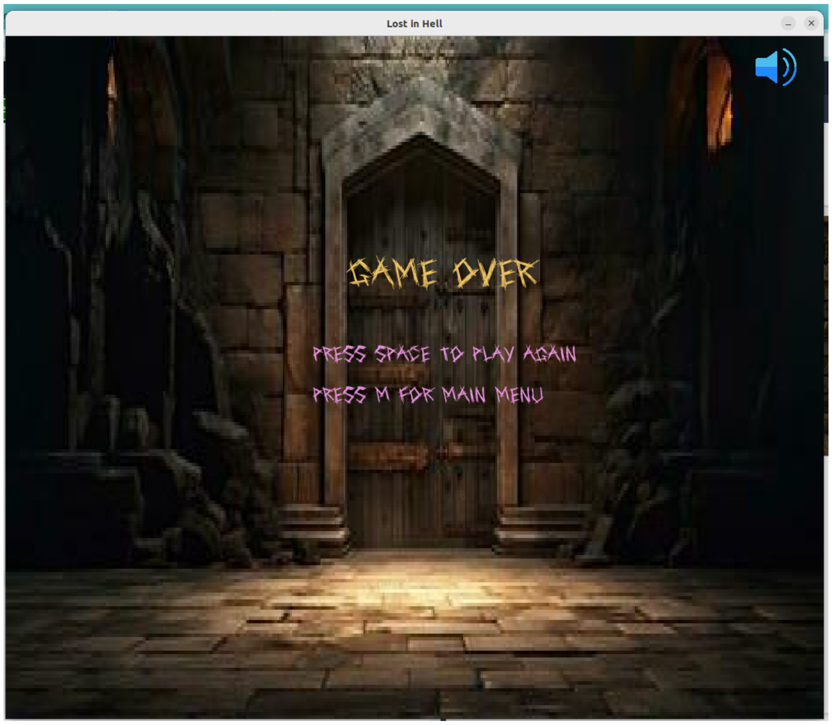


Figure 8: Game lost

3 Basic Completion

Random mazes are generated for 3 levels and solution is there in path.txt. There is an interactive user interface using buttons. The mazes are generated using dfs(depth-first-search) algorithm. This algorithm and its solution are taken from [1] with some input from my side and help from ChatGPT.

3.1 Maze Complexity

In different levels, the size of mazes increases from 15×15 in level 1 to 21×21 in level 2 and 25×25 in level 3.
Also, the probability of finding bombs increases with increasing level, and probability of powerups decreases.

3.2 Showing only some part

Only a 5×5 grid is visible around the player always, except when he collects an eye powerup to view whole maze for some time.

3.3 Miscellaneous

Collision detection and player controls are implemented properly. There is a reverse countdown timer, different for different levels. The formula for calculating score is

$$Score = totalCoinsCollected + lives \times 3 + countdown \times 2$$

4 Advanced Completion

I have added a few interesting additional features:

- There is a high scores leaderboard showing the top 5 high scores of the player upon clicking high score in menu. Also whenever a player makes a new high score, a message informing the same is shown to him. The high scores are stores in *./high_scores.txt*
- Player movements are animated by using sprites as displaying them continuously, also bomb explosion is also animated.
- Background music is present. There are sound effects for coins collection, bomb explosion, etc.
- Many powerups have been implemented as mentioned in 2.1
- There buttons to mute and unmute sound and to go back and forth between different screeens
- There is a key not in the original path, which needs to be collected to open the treasure chest in the end.
- There is an option to pause the game in between, and the pause screen has options for resume, restart and menu.
- You can also change the background music by pressing m on the menu screen and giving path to your favourite music file.

5 My project journey

5.1 Motivation

It was my first time making a game. I chose this project because I had done things similar to the other projects, but making a game using Pygame was the first experience for me.
In this amazing pygame project, I started by watching a few YouTube tutorials then I downloaded the required assets from [2] and [3]. When I was fluent with Pygame, I started on my own developing my own logic to handle different features.

5.2 Bugs

I encountered many bugs on the way like :

- System detecting multiple clicks and arrow key presses when I was using pygame.mouse.get_pressed, so I had to switch to pyagme events (like MOUSEBUTTONDOWN etc.)
- The path algorithm when directly taken from ChatGPT was not working, so I had to modify it by understanding the dfs algorithm.
- There were many minor bugs in logic, which had to be corrected manually by using print statements.

5.3 Learning Outcomes

I learned the basics of game development, how to animate objects, and how to change screens. I also became more fluent in python and now I am familiar with its scope, syntax, etc. In starting, it was looking hard, but as I progressed it became easier once the logic was clear.
I could have added many more features, but due to time constraints I was limited to the few, which I though are most relevant for a maze game.

References

[1] https://www.algosome.com/articles/maze-generation-depth-first.html.

[2] https://www.flaticon.com/.

[3] https://www.techwithtim.net/tutorials/game-development-with-python/pygame-tutorial/pygame-animation.