Maze Game in Python using PYGAME

Evuri Mohana Sreedhara Reddy
CS-108 Project

Spring 2023–24

Abstract

This report outlines the development process of me making the game: Lost in the Maze: A PYGAME Adventure, a 2D maze navigation game. It covers the game's concept, design, implementation, and the challenges encountered during development. The report aims to provide a comprehensive overview that enables understanding and playing the game without direct access to its source code.

Contents

1 Introduction to my Game:

The aim of this game is to complete the mazes generated as quick as possible. The high scores equivalent - *Least Time Taken* is also based on this.

2 Modules

The external modules used are:

- `pygame-ce` - The frequently updated pygame community edition version of pygame, which is a set of Python modules designed for writing video games.
- `Random` - A module that implements pseudo-random number generators for various distributions.
- `Sys` - A module that provides access to some variables used or maintained by the interpreter and to functions that interact with the interpreter.
- `Time` - A module that provides various time-related functions.
- `os` - A module that provides a portable way of using operating system-dependent functionality.
- `heapq` - A module that implements heap queues. I used this module to implement the priority queue for the A* algorithm.

3 Directory Structure

The project directory is as follows:

```
.
├── Modules
│   ├── MainMenu.py
│   ├── PlayGame.py
│   ├── Preferences.py
│   └── Scores.py
├── media
│   ├── fonts
│   ├── images
│   ├── sounds
│   └── videos
├── data
│   ├── path.txt
│   └── LeastTimes.txt
├── game.py
└── settings.py
```

- **game.py** - The main game loop.
- **settings.py** - Has all the global variables and modules necessary for the game to function smoothly.
- **Modules** - The programs which manage various parts of the game.
- **media** - Contain all the images, sounds, and fonts used in the game.
- **data** - Contains the path of the maze and the High-Score Card (Least Time Taken).

4 Running Instructions

4.1 Prerequisites

Note: I am assuming that python is already installed :) The file can be run by using the suitable command of the two:

```
python game.py > /dev/null
python3 game.py > /dev/null
```

If the project is opened in pycharm, a run configuration named Game should be shown, where the program can be run by running that config.
Ensure that `pygame-ce` and only `pygame-ce` are only installed, not the traditional pygame. If you have the traditional pygame installed, run:

```
pip uninstall pygame
pip install pygame-ce
```

4.2 Game Navigation and Gameplay

Note: To ensure easy navigation between various screens, I have introduced the back button which smoothly takes you to the previous screen.

4.2.1 Intro Screen

The game starts with an Intro Screen[1]:

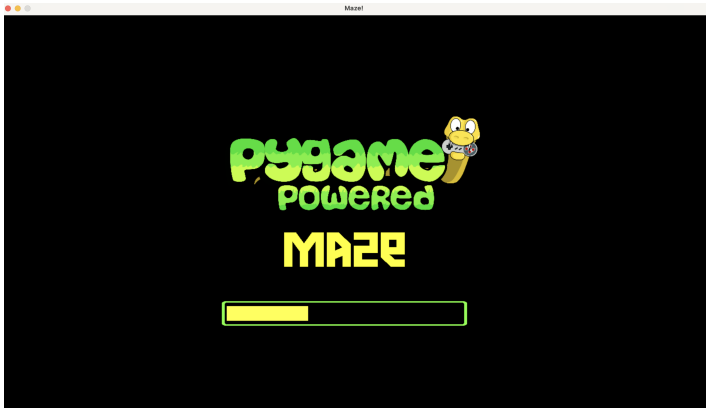

Figure 1: Intro Screen

4.2.2 Main Menu

After Loading, we will be greeted with a Main Menu, from which we can choose to:

- Play
- See the Fastest Solves in each Level
- Customise the Game: Mute or Unmute
- Quit

We can select any of these by pressing on these buttons[2].

4.2.3 Game Level Selection

We have 3 levels of mazes, we can choose anyone[3].

4.2.4 The Game!

The game starts, waiting for you to give the input of navigation using the arrow keys or [W A S D]. Your aim is to go to the diagonally opposite corner, which has another door, waiting for you to open it. The *Score* is measured in terms of the time taken to reach the opposite end: the Lower, the Better!

We also have various themes which can be changed using the *Change Theme* button. The music can be turned off by pressing the Music button.

Some examples of the game screens[4][5].

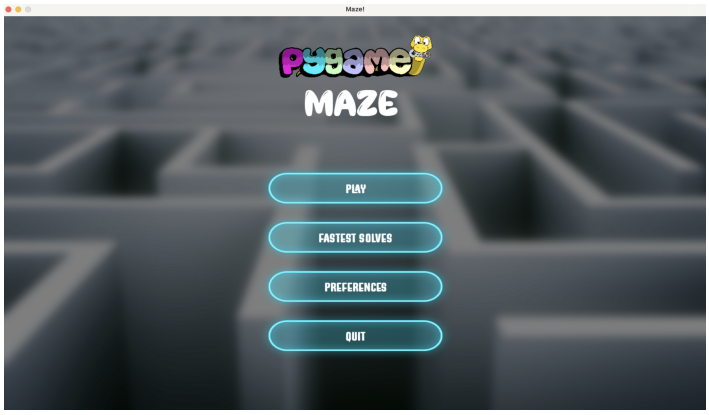

Figure 2: Main Menu

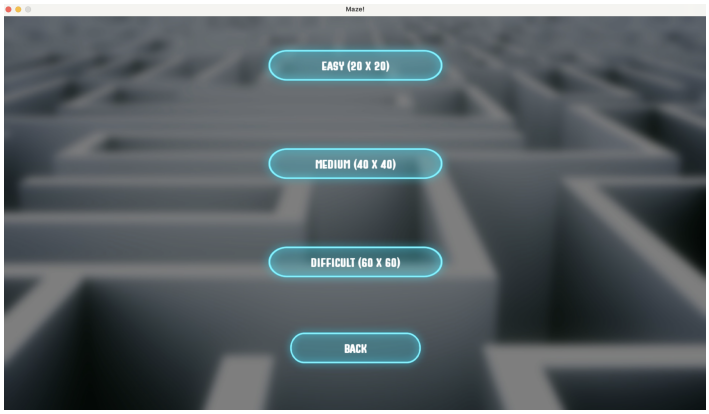

Figure 3: Game Level Selection

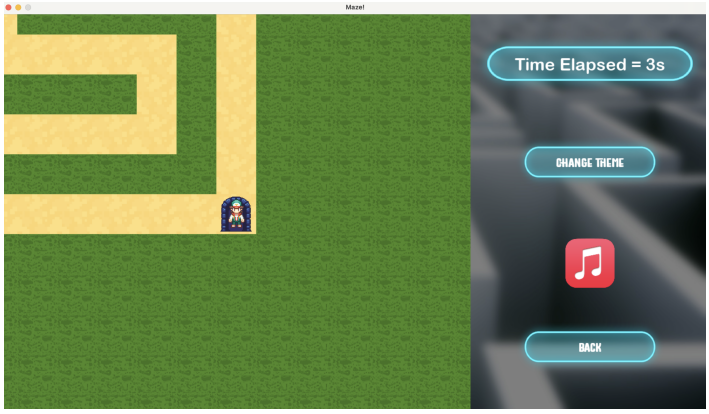

Figure 4: Game Starts!

4.2.5 Game Over

On reaching the opposite end, the game ends, and the time taken is displayed. The screen looks like this[6]:

4.2.6 Fastest Solves

On clicking the Fastest Solves button on the Main Menu, you will see the Least Time taken to solve the various levels of the maze. An example screen looks like this[7]:

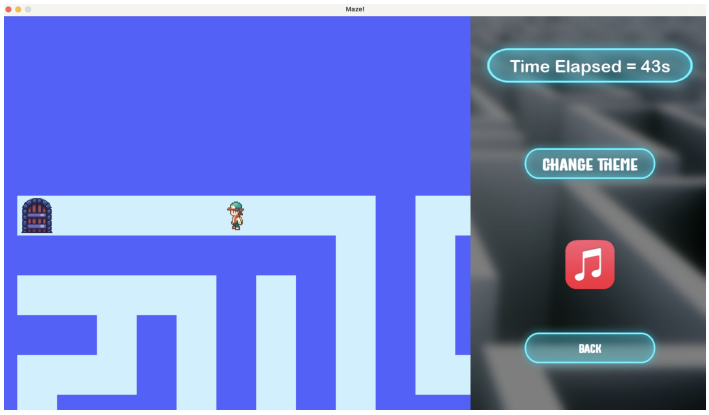

Figure 5: Game Starts!

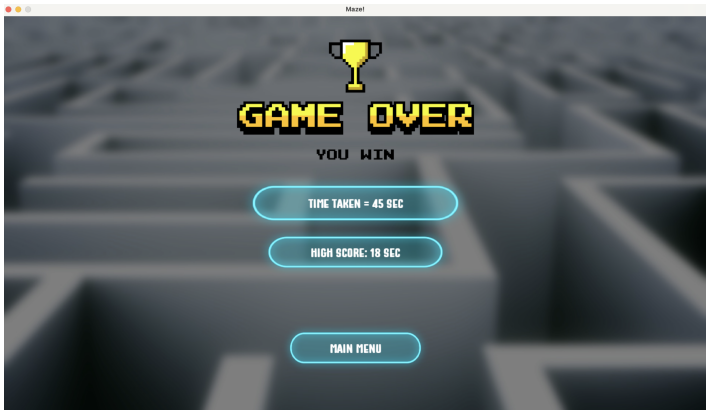

Figure 6: Game Over

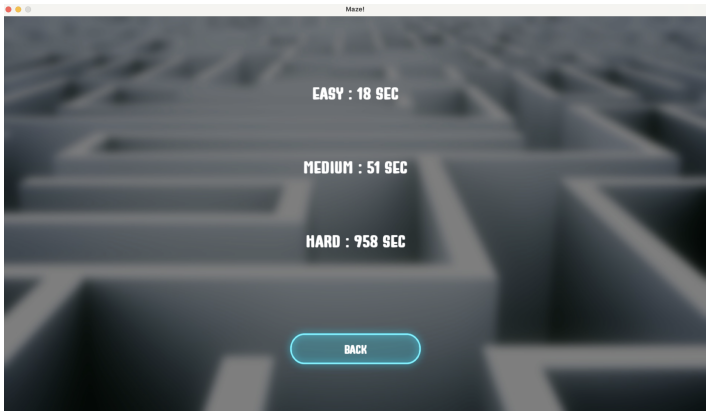

Figure 7: Fastest Solves

4.2.7 Preferences

This window enables you to mute the music part of the game. You can do this by clicking on the red music button. If you want The screen looks like this[8][9]:

4.2.8 Quit

On clicking this button, the Game ends and the program terminates.

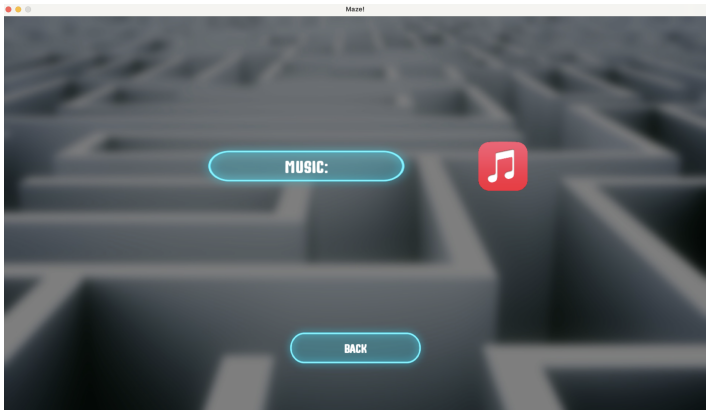

Figure 8: Music On

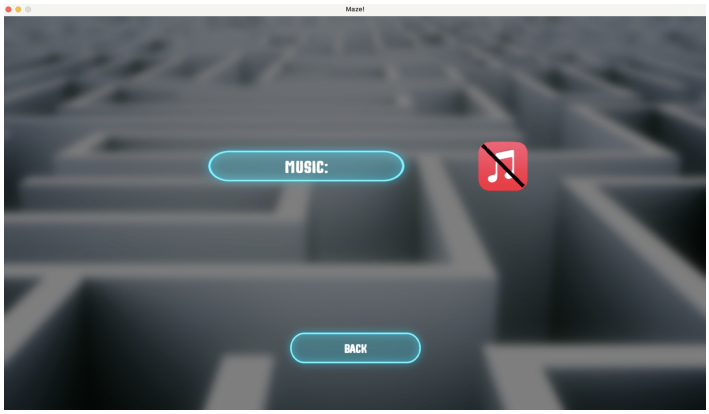

Figure 9: Music Off

5 Various Implementations in the code

For the maze generation, I used the *Recursive Backtracking* algorithm. This algorithm is a randomized version of the depth-first search algorithm. The algorithm starts at a random cell and chooses a random neighboring cell that has not been visited, creates a path between the two cells, and moves to the neighboring cell. The algorithm continues until it has visited every cell in the grid. I have modified this algorithm slightly to make the wall size and the path size the same, which makes the maze look more appealing. [3]
For the pathfinding, I used the *A** algorithm. The *A** algorithm is a pathfinding algorithm that uses a heuristic to determine the next node to visit in a graph. The algorithm uses a priority queue to determine the next node to visit based on the cost of the path to that node and the heuristic value of the node. The algorithm continues until it reaches the goal node or there are no more nodes to visit. I have used the `heapq` module to implement the priority queue for the A* algorithm. [4] For the pygame functions, I referred to the official documentation of pygame[1] and pygame[2], mostly the latter for the updated functions and methods.

5.1 Customization in the Game

A list of all the special customization implemented in the game:

- Animation when the player moves.
- Dynamic Background of the Main Menu.
- Music and Sound Effects.
- Themes for the Game.
- Music and Sound.

- High Scores.
- Preferences.
- Back Button for easy navigation.
- Customized Fonts.
- Responsive Buttons.

References

[1] Pygame Official Documentation https://www.pygame.org/docs/.

[2] Pygame CE Official Documentation https://pyga.me/docs/.

[3] Maze Generation Algorithms (by professor-l) https://professor-l.github.io/mazes/.

[4] A* Algorithm https://github.com/rennaMAhcuS/Maze/blob/main/OtherResources/A*.md