A Random Maze Project

Surya Prakash V

April 28, 2024

1 Maze Making Documentation

1.1 Introduction

This documentation provides an overview of how I prepared a random maze game in roughly 15 days of constant work and determination. This maze game supports customizing audio, images, sprite objects by replacing them with the original name. It features an intuitive UI allowing users to mute, pause, and interact with buttons that change color. However, due to time constraints, the sprite functionality remains incomplete.

1.2 Efficient Usage of Sprites

There are four types of sprites in my game:

- Player sprite: Contains animations and detects collisions with other sprites.
- Wall sprite: Contains positions of all wall positions and detects collisions with the Player sprite to prevent passing through the maze.
- Explosion sprite: Contains animations, and when a player hits it, lives are reduced by one.
- Villain sprite: Similar to the Explosion sprite but with different animations.

2 Modules Utilized

- *pygame*: To run the game[1]
- *random*, *shuffle*: To put some sprites in random locations
- *os*: To run a python program from main.py
- *math*: To use π

[1] Pygame Documentation: https://www.pygame.org/project/733

- YouTube tutorials were also referenced for certain aspects of the game development process.

3 Directories of the Game

- main.py: is in the parent folder
- button.py: is in the parent folder contains UI for main.py
- 1.py, 2.py, 3.py: are in the daughter directory of name difficulties
- Timer.py: in the daughter directory of name difficulties used to show remaining Time in a simple, gorgeous way
- heart.py: is in the daughter directory of name difficulties used to show remaining lives in a gorgeous way
- images, music, fonts directories: are in the parent folder and used to make more UI
- maze.py: used to make maze image and being utilized by Timer.py

4 Instructions

Here are the rules for playing the game:

4.1 Running the game

- Open the submission folder in VS Code or any other similar IDE.
- Run the Python file `main.py` and choose to Play.
- On clicking the Play button, the program will provide you with three levels to choose from, each one harder than the previous.
- To play, click on any level using your mouse.
- The game will start running.
- You can interact with buttons that change color when hovered upon.

4.2 High Scores

- Clicking the HighScores button will display the top 3 high scores. By default, these scores are 0.
- Playing the game will update these scores accordingly.

4.3 Controls in the Game

The controls when playing `1.py`, `2.py`, `3.py` (the main game files) are as follows:

- Left arrow: Moves the player to the left
- Right arrow: Moves the player to the right
- Up arrow: Moves the player upward
- Down arrow: Moves the player downward
- Space bar: Pauses the game

4.4 Game Over

The game ends and shows the after screen under the following conditions:

- If Timer = 0
- If lives = 0

4.5 To Win Game

- If you find the Dragon Ball before the timer completes or you exhaust your lives, you win the game.

4.6 Score

The score is calculated by the following formulas:

4.6.1 If Won

- Score = timer $\times$ 10 + lives $\times$ 100 + 1000

4.6.2 If Lost

- Score = timer $\times$ 10 + lives $\times$ 100

5 Development Process

The journey of developing the game encompassed several phases, each marked by unique challenges and creative solutions.

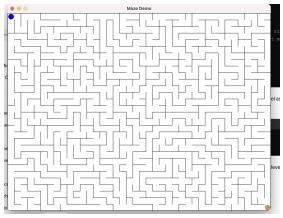
5.1 Initial Research and Prototyping

At the onset, thorough research was conducted to understand the fundamentals of game development and to gather inspiration from existing projects. YouTube emerged as a valuable resource, where tutorials on creating a main menu provided essential insights. By leveraging code snippets and tutorials, I acquired the necessary foundations to kickstart the development process. However, mere replication was not sufficient; extensive modifications and customizations were made to ensure alignment with the envisioned game concept.

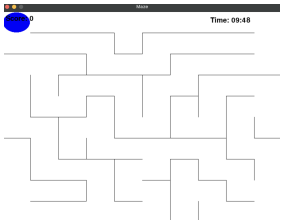
5.2 Maze Design and Implementation

The maze aspect of the game presented intriguing challenges. Referencing resources from the Pygame website, I explored various maze generation algorithms to create dynamic and engaging levels. Initial attempts with the first maze type encountered hurdles in maintaining consistent wall thickness, despite exhaustive troubleshooting efforts over several days. Faced with this setback, I transitioned to an alternative maze design. Although lacking the desired zoom functionality, it served as a suitable foundation for further development.

Collaboration played a pivotal role in overcoming obstacles encountered during maze implementation. Consulting with peers and exchanging ideas led to breakthrough solutions, enhancing both the efficiency and creativity of the development process. Through iterative experimentation and feedback integration, the maze system gradually evolved, culminating in a visually appealing and seamlessly navigable environment.



(a) Maze before



(b) Maze After

5.3 Integration of Advanced Features

As development progressed, attention shifted towards implementing advanced features to enrich the gameplay experience. One significant challenge involved optimizing maze collisions to ensure fluid interaction between the player character and the environment. By employing sprite animations and fine-tuning collision detection algorithms, I achieved a harmonious balance between realism and responsiveness, enhancing immersion for players.

Furthermore, the addition of end screens, high score tracking, and dynamic music integration further elevated the game's depth and replay value. Iterative

refinement based on user feedback and playtesting sessions ensured that each feature contributed meaningfully to the overall experience, fostering engagement and enjoyment.

5.4 Continuous Improvement and Iteration

The development process was characterized by a commitment to continuous improvement and iteration. Regular playtesting sessions and feedback gathering enabled me to identify areas for enhancement and refinement. Through diligent iteration and fine-tuning, the game gradually evolved from a concept to a polished and immersive experience, ready for release to the wider audience.

Overall, the development journey was a testament to perseverance, creativity, and collaboration. By embracing challenges as opportunities for growth and learning, I was able to overcome obstacles and realize the vision of creating a captivating and enjoyable game.

6 Difference in levels

These are the values in each level:

6.1 Lvl.1

- lives = 3
- maze = 10×10
- timer = 140
- explosions, villains = 2,2

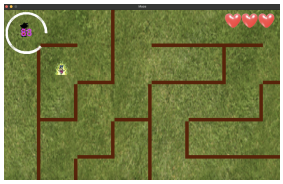
6.2 Lvl.2

- lives = 3
- maze = 15×15
- timer = 120
- explosions, villains = 3,3

6.3 Lvl.3

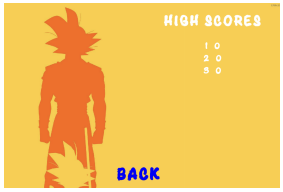
- lives = 3
- maze = 15×15
- timer = 100
- explosions, villains = 3,3

7 Final Game

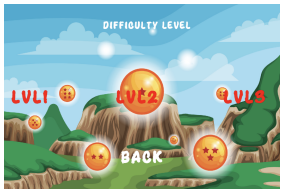


(a) Game Screen

(b) Pause Screen



(a) Scores



(b) Pause Screen

(c) Home screen

8 Conclusion

In conclusion, the development of the random maze game was a journey marked by dedication, collaboration, and continuous improvement. Through meticulous planning and iterative refinement, the game evolved into an immersive experience, offering players engaging challenges and dynamic gameplay. The integration of advanced features and attention to detail underscores the commitment to delivering a polished product. Overall, the project showcases the power of perseverance and creativity in realizing a captivating gaming experience.

References

[1] Pygame. https://www.pygame.org/project/733

[2] baraltech.HOW TO MAKE A MENU SCREEN IN PYGAME!.2022. https://www.youtube.com/watchv=GMBqjxcKogAt=20spp=ygULcHlnYW1lIG1lbnU