Lost in the Maze : A Pygame Adventure
(Author & TA : Guramrit Singh, CS-108, Spring 2023-24)

Major updates:

April 9: You can now use any search algorithms from the internet like BFS, DFS, etc with proper citation in the report for constructing random maze and its solution path. You are free to directly copy the algorithm from the internet and you won't be asked any questions pertaining to the algorithm in the viva.

April 9: Customization (5 marks) for everyone will be relatively graded to some extent.

April 9: A new resource has been added.

General Description:

I believe all of us have played maze games at some point in our lives. I personally have played them a lot, usually in *newspapers* and *magazines*. The objective of the game is to navigate through a maze from the starting point to the ending 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 to complete each level successfully.

The **aim** of this project is to design and implement a variant of this **2D maze game** using **Pygame**, a popular Python library for game development. The game will consist of a single player-controlled character navigating through a maze with a restrictive 2D top view from the starting point to the end point while

avoiding obstacles and traps. The project will involve creating various levels of increasing difficulty, implementing collision detection, designing intuitive user interfaces, and integrating sound effects and visual elements to enhance the gaming experience. There will also be marks for creativity and innovation in game design and implementation. Furthermore, the project will also require you to give a **LaTeX report** and **viva** on the project.

Key Objectives (Minimum Requirements for Basic Completion):

1. Designing an intuitive user interface for the game menu i.e opening screen, gameplay screen, and game over screen.

2. Game menu should have options for starting a new game, selecting difficulty levels, and exiting the game.

3. Generating random mazes for each level, you need to have at least 3 levels and ensure that each maze is solvable i.e there is a path from the starting point to the end point. Just to help you and TAs out with this solvability part, generate a path from the starting point to the end point and save it in a file 'path.txt', use 'U' for up, 'D' for down, 'L' for left, 'R' for right. Note that by following this path, you should be able to reach the end point from the starting point. Upon initiating the game, it is imperative that a fresh maze be generated dynamically every time. This means that the game should not rely on pre-existing hard-coded mazes but rather create a new random maze at the start of each session. (**Hint:** You can first generate a random path from start point to end point and later place walls, traps etc such that they do not affect the solution path i.e. you compute the solution path first and then later generate the maze corresponding to the solution path. Another method is to use any search algorithm from the internet like BFS, DFS, etc with proper citation in the report for constructing random maze and its solution path.)

4. Players should only be able to see some part of the maze in their vicinity, the rest of the maze should be hidden. This can be thought of as having a camera that follows the player and shows only the part of the

maze that is visible to the player. *For example*, you can show a 5x5 grid around the player and hide the rest of the maze.

5. Note that your maze size should be at least twice the size of the visible grid in terms of rows and columns. *For example,* if you are showing a 5x5 grid around the player, then your maze should be atleast 10x10.

6. Difficulty levels should affect the maze complexity, number of obstacles, less time to complete the maze, etc.

7. Implementing player controls for navigating the maze using keyboard inputs or mouse clicks.

8. Incorporating collision detection to prevent the player from passing through walls or obstacles.

9. Introducing features such as score, timer, and lives to enhance gameplay dynamics. You can define what the score means. For example, it might be the difference between initial distance between start point and end point and corresponding distance at current point in time or if you are considering addition of collectables (*see point 4 in additional requirements*), then your score may be defined as the number of items collected multiplied by their respective weights. Timer is a reverse counter, where the game ends if time left becomes 0.

10. Testing the game thoroughly to ensure smooth gameplay, bug fixing, and optimization for performance.

11. Commenting the code to explain the logic behind each function, class, module etc. You don't need to write comments for each line. Just give a short description before definition of each function and/or class. See this for reference.

12. Providing a LaTeX project report.

LaTeX report

The report must detail the development process, including the modules utilized, implementation methods for the game, added features, differentiation among levels, gameplay instructions, and more. It's advisable to incorporate images and/or videos showcasing different stages of the game to make the report attractive. Additionally, include a references section with links to sources such as websites, articles, books, etc., from which coding concepts were used in your project.

[This is important so that you don't get reported for plagiarism].

Additional Objectives (You can add as many features as you want, there is no upper bound)

The sky is not the limit. Your mind is.

Marilyn Monroe
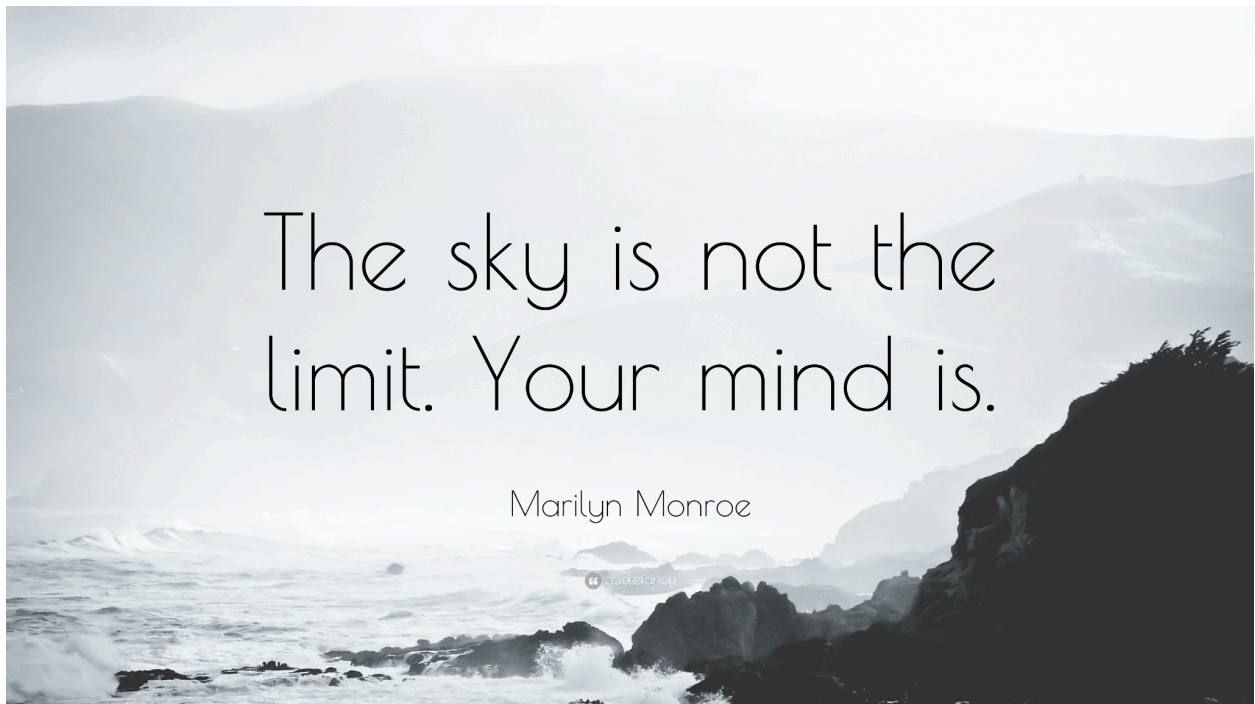
[EXAMPLES, you can come up with your own ideas as long as they are reasonable, innovative and creative]:

1. Maintaining a high score leaderboard to track players' performances across multiple game sessions. You may use a text file to store the high scores against the user name. Display the top 5 high scores or so on the game over screen. Give a menu option to view the high scores.

2. Integrating sound effects for player actions, collisions, and level completion. You may use the Pygame mixer module to play sound effects. You can also add background music to enhance the gaming experience. Give an option to mute the sound. Maybe you can also give an option to change the background music, this can be done by taking the path of the music file as input from the user as a command line argument.

3. Implementing visual enhancements such as animations, particle effects, and background elements. You can use Pygame's sprite module to create animated sprites for the player character and obstacles. You can also add visual effects like particle explosions when the player collides with an obstacle or reaches the end point. You can integrate vents into the environment, drawing inspiration from "Among Us" . These vents would feature animations and sound effects for both opening and closing. Additionally, you should depict a smooth transition from the current frame to the frame on the other side of the vent. You can also color the maze walls and paths differently to make the maze more visually appealing. You can as well color the timer and score according to their values, for example, if the timer is running out, you can color it red, if the score is increasing, you can color it green, etc.

4. Adding extra features like power-ups, collectibles, and special abilities to make the gameplay more engaging. You can introduce power-ups that grant temporary invincibility, speed boosts, or extra lives to the player. You can also add collectibles like coins, keys, or gems that provide bonus points or unlock secret areas or shortcuts in the maze. A key may be used to unlock a door that connects two parts of the maze which were previously disconnected i.e. there was no way to get from one part of the maze to the other. A gem may be used to temporarily reveal a larger portion of the maze.

Modules allowed:

You are required to generate the maze on your own using the pygame module. You cannot use any external module to generate a solvable maze for you.

Here's a general list of allowed modules: pygame, sys, os, argparse, random, numpy, time, queue, and math.

If you want to use or include any other useful module in the above list, then kindly talk to the TA incharge of this project.
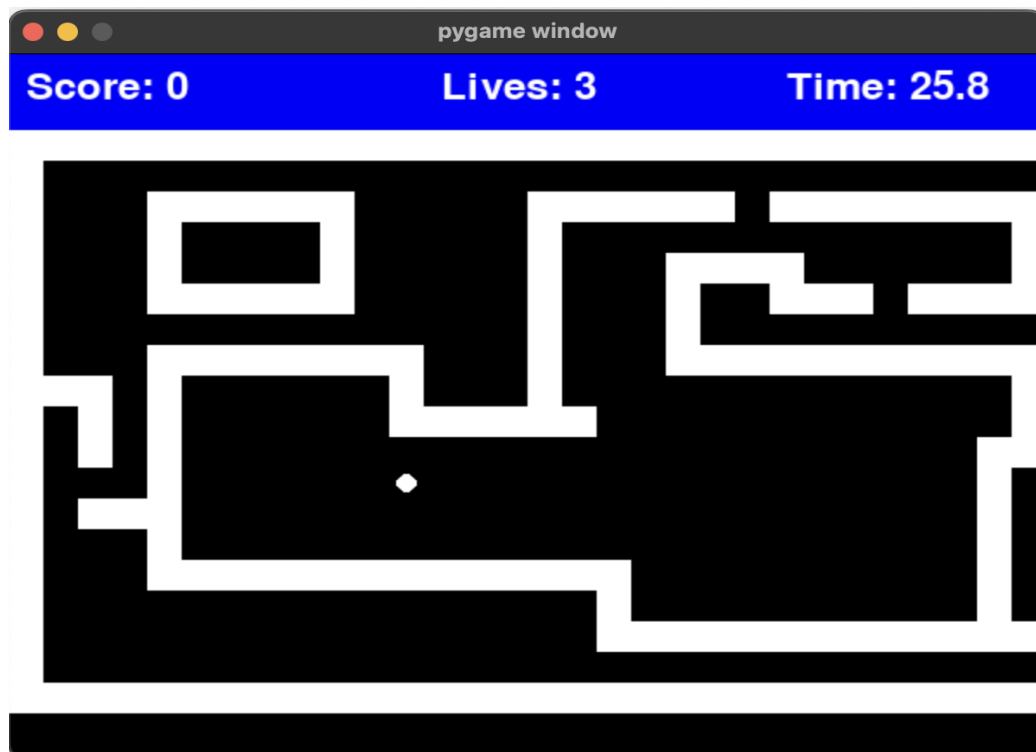
Some example images illustrating how various screens of the game appear after basic completion

These designs are rudimentary, feel at liberty to enhance the screen display as you see fit.

1. Opening screen
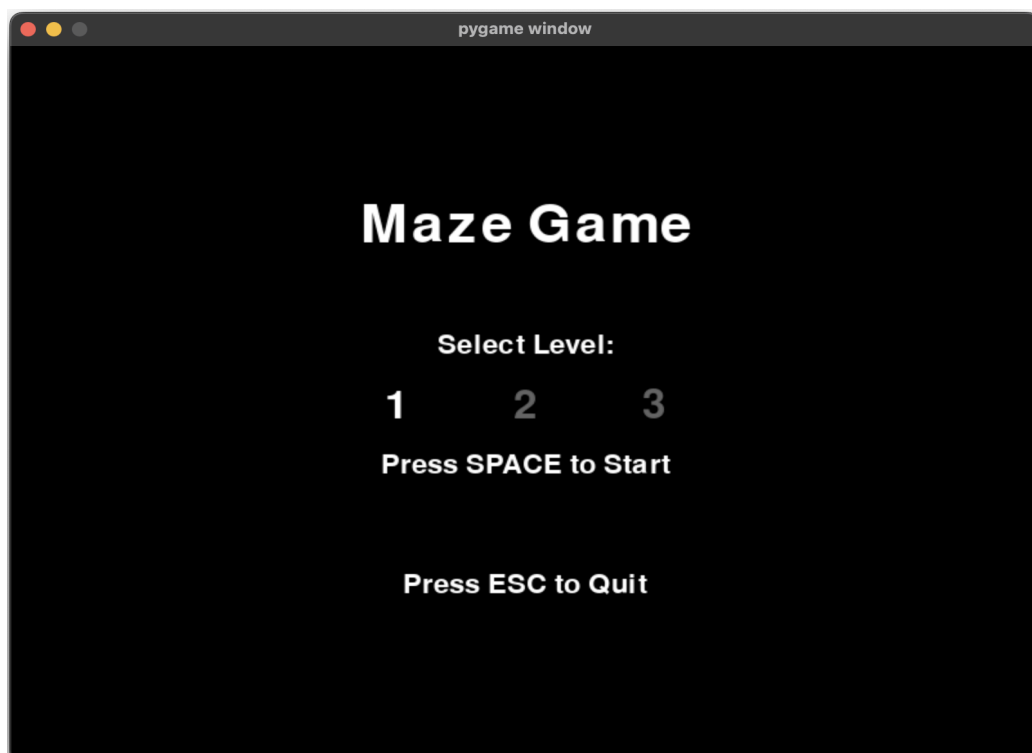
2. Gameplay screen

3. Game over screen

References:

1. [Pygame documentation](#)

2. [Project 1 - Alien Invasion](#)

3. [Snake game](#)

Evaluation Criteria and Marks Distribution (Total 15 marks):

8 marks for the basic requirements.

5 marks for the additional requirements i.e. graded based on your creativity and innovation in the game's features.

2 marks for the LaTeX report and comments in the code.

Viva evaluation does not have separate marks allocated. Instead, your performance across the aforementioned tasks will be assessed for a total of 15 marks.

Naming convention for the files:

1. 'game.py' : Main python file implementing the maze game using Pygame. TA will run this program to test your implementation.

2. 'report.pdf' : LaTeX project report detailing the points mentioned above.

3. Any other files that you may have used in the project like other python files, images, sounds, etc can be named as per your wish but give a brief description about them in the report.

Submission instructions: (Tentative)

1. Create a folder named 'submission'

2. Place all your files and folders inside it

3. Tar the 'submission' folder and name the tar file as 'submission.tar.gz'

4. Submit the 'submission.tar.gz' file

5. The submission platform is yet to be decided and will be announced later

IMPORTANT INSTRUCTIONS:

1. **PLAGIARISM/COPYING FROM EACH-OTHER WILL LEAD TO SEVERE PENALTIES.**

2. Use of AI tools are not prohibited but also not recommended. You should be able to explain anything the TA asks during Viva to get marks for that part. It is recommended to refer to the official documentation of pygame and any other modules you intend to utilize, rather than relying solely on inquiries to GPT.

3. Include all references in the report so that you don't get a tough time later 🙂.

4. Try to modularize your code as much as possible. You can create separate modules for generating mazes, player controls, collision detection, etc. This will make your code more readable, maintainable, and reusable. Use variable names that are descriptive and meaningful to convey the purpose of the variable. This is good programming practice and will help you in the long run.

5. Please feel free to ask any doubts related to the project to *Guramrit Singh* (WhatsApp number : *+91 8837565814*)

"Wisdom is not a product of schooling but of the lifelong attempt to acquire it."

Albert Einstein

BEST OF LUCK!