Documentation: MazeSolverProject

Name: K. Pranvith Chowdary

Roll no: SE22UCSE133

1. Project Overview

This project is a **Maze Solver** application that allows the user to generate and solve a maze using various algorithms. The user can interact with the maze by controlling a player who needs to find a path from a starting point to the goal.

The following algorithms are implemented to solve the maze:

- **Breadth-First Search (BFS)**
- **Depth-First Search (DFS)**

2. Features

- **Maze Generation**: Generates a random maze with walls and paths, ensuring the player can reach the goal.
- **Player Movement**: The player can move through the maze using arrow keys.
- **Timer**: A timer tracks how long the user takes to solve the maze.
- **Solve with Algorithms**: The user can choose to solve the maze using BFS, DFS.
- **Dynamic Difficulty**: The difficulty of the maze can be adjusted by selecting from Easy, Medium, or Hard levels.

3. Technologies Used

List the technologies and tools used in the project.

Example:

- **Java**: The project is written in Java.
- **Swing**: The GUI is built using Swing for creating interactive panels, buttons, and labels.
- **Java Collections Framework**: Data structures like Queue and LinkedList are used for BFS.
- **Point Class**: Used to represent positions in the maze.

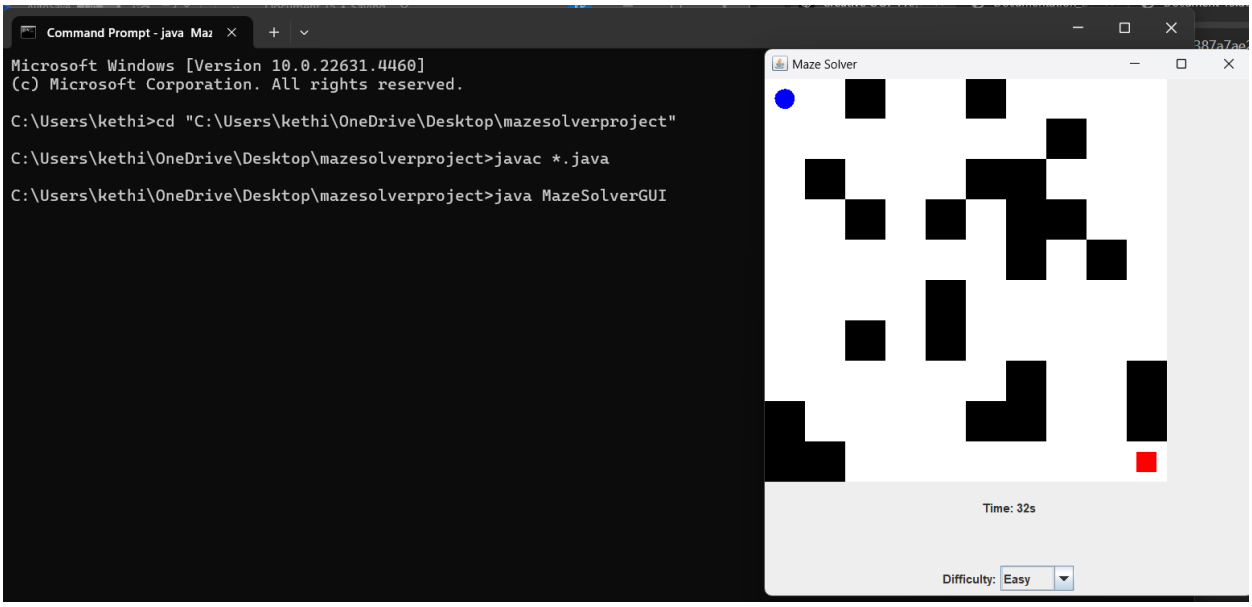
4. Installation Instructions

1. **Install Java**: Make sure you have JDK 8 or higher installed.
2. **Run the Application**: Compile and run the project from your preferred IDE (such as IntelliJ IDEA or Eclipse) or the command line:

   Code:

   javac MazeSolverGUI.java

   java MazeSolverGUI



5. Usage Instructions

1. **Starting the Game**: When you run the program, a window will appear showing a maze with walls, a starting point (player), and an ending point (goal).
2. **Player Movement**: Use the arrow keys to move the player through the maze. The player will move one step at a time.
3. **Solve the Maze**: To solve the maze, you can use the "Solve with BFS," "Solve with DFS," or "Solve with A*" buttons. Each button will trigger a corresponding algorithm to solve the maze:
   - **BFS**: Finds the shortest path from the start to the goal.
   - **DFS**: Searches the maze depth-first to find a path.

- **A***: (Currently not implemented) will eventually use heuristics to find the shortest path.

4. **Timer**: The timer will display the time taken to solve the maze. If the player reaches the goal, the timer stops and displays a congratulatory message.
5. **Changing Difficulty**: You can change the difficulty of the maze by selecting an option from the "Difficulty" dropdown. The maze will regenerate with a different density of walls based on the selected difficulty.

6. Code Structure and Explanation

Maze Class

- Responsible for generating the maze grid, placing walls, and ensuring the maze has a solvable path.
- Contains methods to get the size of the maze, the start and goal positions, and to check if a move is valid.

Solver Class

- Contains the logic for solving the maze using different algorithms (BFS, DFS, and).
- Includes methods for each algorithm, where BFS and DFS are implemented.

Player Class

- Represents the player in the maze, keeping track of the player's current position and whether they have won by reaching the goal.
- Includes methods for moving the player and resetting the position.

MazePanel Class

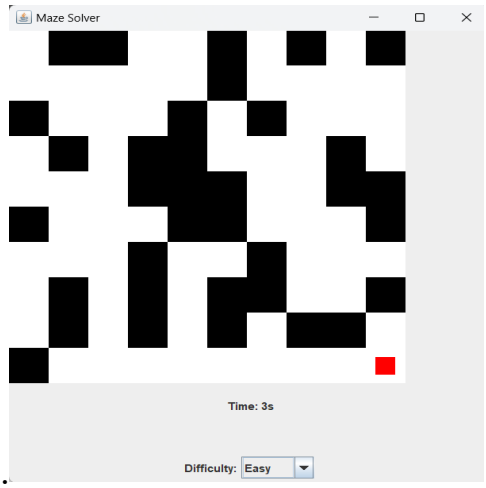
- Responsible for rendering the maze and displaying the player's current position.
- Uses paint Component to draw walls, paths, and the player on the screen.

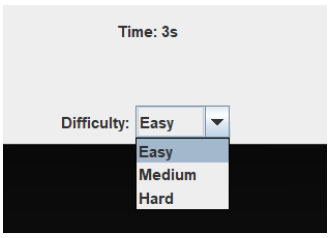
MazeSolverGUI Class

- The main GUI class that sets up the interface, adds buttons for solving the maze and allows interaction with the player and algorithms.
- Includes a timer to track how long it takes to solve the maze and a difficulty selector to change the maze's complexity.

User Interface

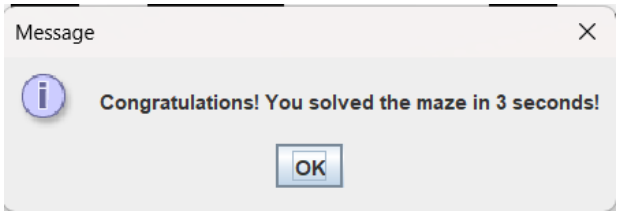
GUI Components

- **Maze Display**:
    - A grid visualizing walls, paths, the player's current position (blue), and the goal (red).
    - Resizes dynamically based on the maze size
    - 

- **Control Panel**:
    - **Timer Label**: Displays elapsed time during the game.
    - **Difficulty Selector**: Dropdown to change maze difficulty (Easy, Medium, Hard).
    - 

- **Pop-Up Notifications**:
    - Alerts the user when:

- The player reaches the goal.



7. Future Improvements

- **Visual Enhancements**: There could be additional visual feedback when solving the maze (e.g., highlighting the solution path).

8. Conclusion

This Maze Solver project demonstrates the implementation of different maze-solving algorithms, including BFS and DFS, in a graphical interface. While the project is functional, there is room for further improvement, such as implementing the A* algorithm and enhancing the visual aspects of the application. It provides a fun and interactive way to learn about algorithms and maze generation.
