Green University of Bangladesh

Department of Computer Science and Engineering (CSE)
Semester: (Fall, Year: 2023), B.Sc. in CSE (Day)

Maze Solver

Course Title: Microprocessors and Microcontrollers Lab
Course Code: CSE 304
Section: 213-D1

Students Details

Name	ID
Irteja Mahmud	213902016
Md. Tanjim Mahtab Taosif	213902007

Submission Date: 28 December 2023
Course Teacher's Name: Sudip Ghoshal

[For teachers use only: Don't write anything inside this box]

Lab Project Status	
Marks:	**Signature:**
Comments:	**Date:**

Contents

Chapter 1

Introduction

1.1 Overview

The name of my project is Maze Solver Game. The Maze Solver Game project is an interactive text-based game designed to challenge players with navigating a maze. The game offers the following key features:

- **Game Introduction:** The program begins by welcoming the player and providing clear instructions on how to play the game, explaining the rules and objectives.
- **Maze Representation:** The maze within the game is depicted as a grid of cells. It is defined using a series of bytes (0s and 1s) in 'maz1' through 'maz20', where '0' signifies an open path, and '1' represents the presence of a wall.
- **Game Loop:** The core of the program is a game loop, where the player's position, symbolized by a smiley face, is constantly displayed on the screen. The player can move through the maze using arrow keys.
- **Movement and Collision Detection:** The code includes functions to manage the player's movements, detect collisions with walls, and keep track of the player's position and direction within the maze.
- **Win Condition:** If the player successfully navigates the maze, reaching the exit, a congratulatory message is displayed, and the game concludes.
- **Display Functions:** The program incorporates various functions for displaying text, updating the screen, and handling keyboard input, ensuring a smooth and engaging gaming experience.

Overall, the Maze Solver Game project is a text-based maze-solving game that challenges players to find their way through a maze using arrow keys while avoiding collisions with walls.

1.2 Motivation

The game provides an entertaining and challenging experience for players as they navigate through a maze using arrow keys. The goal of the game is to reach the exit of the maze with as few moves as possible.

- **Entertainment:** The primary motivation for creating such games is to entertain players. Solving mazes and overcoming obstacles can be a fun and engaging experience.
- **Problem Solving:** Maze-solving games encourage players to think critically and strategically as they find their way through complex paths. This promotes problem-solving skills.
- **Skill Development:** The game challenges players' hand-eye coordination and reflexes as they control the character using arrow keys, enhancing their gaming skills.
- **Competition:** Players can compete with each other to see who can solve the maze in the fewest moves, adding a competitive element to the game.
- **Learning:** Game development projects like this one provide a platform for learning and experimenting with programming, graphics, and user interactions.

Overall, "MAZE SOLVER" is a project that combines entertainment with cognitive challenges, making it a rewarding and motivating experience for both players and developers.

1.3 Problem Definition

1.3.1 Problem Statement

The Maze Solver project aims to address the challenge of creating an engaging and interactive text-based game where players must navigate through a maze using arrow keys while avoiding collisions with walls. The key problem is to design and implement the game logic, including maze generation, player movement, collision detection, and win conditions, to provide players with an entertaining and challenging experience. Additionally, the project seeks to promote problem-solving skills, hand-eye coordination, and competition among players while serving as a platform for learning and experimentation in game development and assembly language programming.

1.3.2 Complex Engineering Problem

The complexity of the engineering problem in this project can be summarized as follows:

Table 1.1: Summary of the attributes touched by the mentioned projects

Name of the P Attributess	Explain how to address
P1: Depth of knowledge required	Understanding x86 assembly language programming, specifically for the emu8086 environment, and knowledge of maze traversal and game development concepts
P2: Range of conflicting requirements	Creating a challenging maze while keeping it solvable and ensuring an enjoyable player experience.
P3: Depth of analysis required	Understanding and implementing a maze-based game with various game mechanics, character movement, collision detection, and user interface elements in x86 assembly language using the emu8086 platform.
P4: Familiarity of issues	Defining a maze, implementing movement controls, tracking the player's position, detecting collisions with walls, and displaying messages upon completion.
P7: Interdependence	The Maze Solver project's success relies on interdependencies between algorithm development and maze representation, user interface design and real-time visualization, algorithm implementation and error handling, and user feedback driving interface improvements.

1.4 Design Goals/Objectives

- **Create an Interactive and Engaging Text-Based Maze Game:** The primary objective is to develop a text-based maze game that immerses players in a challenging and interactive experience, encouraging them to strategize and escape the maze.
- **Implement Smooth and Intuitive Player Controls Using Arrow Keys:** Design the game with responsive arrow key controls to ensure that players can navigate the maze smoothly and intuitively.
- **Design a Challenging Maze Layout Requiring Strategic Thinking:** Craft maze layouts that pose a substantial challenge, prompting players to think strategically and make informed decisions to find their way out.

- **Keep Track of Player's Moves and Provide Feedback on Total Moves:** Record and display the total number of moves made by the player, offering feedback on their progress and performance as they navigate the maze.
- **Develop an Aesthetically Pleasing User Interface with Clear Instructions:** Create an appealing user interface that provides clear instructions to guide players through the game, enhancing the overall user experience.
- **Add a Win Condition to Congratulate the Player Upon Successful Escape:** Implement a win condition that triggers a congratulatory message when the player successfully escapes the maze, offering a sense of accomplishment and closure.
- **Encourage Replayability with Random Maze Generation or Multiple Levels:** To maintain player interest, introduce replayability by incorporating random maze generation or multiple levels, offering fresh challenges each time.
- **Promote Code Readability and Maintainability for Future Enhancements:** Structure the code with comments and clear organization to enhance readability and maintainability. This ensures that future enhancements or modifications can be made efficiently.
- **Provide a Simple and Enjoyable Gaming Experience for Players:** Ultimately, the goal is to deliver a straightforward yet enjoyable gaming experience that captivates players, keeping them engaged and entertained throughout their maze-solving journey.

1.5 Application

The provided code appears to be a text-based game written in assembly language for the EMU8086 emulator. In the real world, a project like this could serve as a simple game or educational tool to help individuals learn assembly programming concepts. It's a practical way to understand low-level programming and can be used for educational purposes in computer science and software development courses.Furthermore, it can be a valuable resource for individuals aiming to gain hands-on experience with legacy hardware emulation, deepening their understanding of computer architecture and assembly language programming.

Chapter 2

Design/Development/Implementation of the Project

2.1 Introduction

The maze solver project is designed to provide an engaging and interactive experience for users as they navigate through a virtual maze using arrow keys. The implementation involves intricate maze-solving techniques, user interface design, and efficient algorithms to ensure an enjoyable gaming experience.

2.2 Project Details

The Maze Solver project is developed with the goal of creating an intuitive and interactive platform for solving mazes. This project is implemented in emu8086, utilizing cutting-edge techniques to provide users with a dynamic visualization of maze-solving processes. The maze-solving algorithms implemented in this project include several challenges, each designed to address the challenges associated with navigating complex mazes.

2.2.1 Key Features

The maze solver project encompasses several key features that contribute to its functionality and appeal:

- **Arrow Key Controls:** Users can navigate through the maze using arrow keys, providing an intuitive and responsive interface..
- **Interactive Maze:** The maze is designed to be visually appealing, with walls and pathways represented on the screen.
- **Real-time Feedback:** The user receives real-time feedback on their movements and progress within the maze.

- **Game Rules:** Clear and concise game rules are presented to guide users on how to play and navigate the maze effectively.

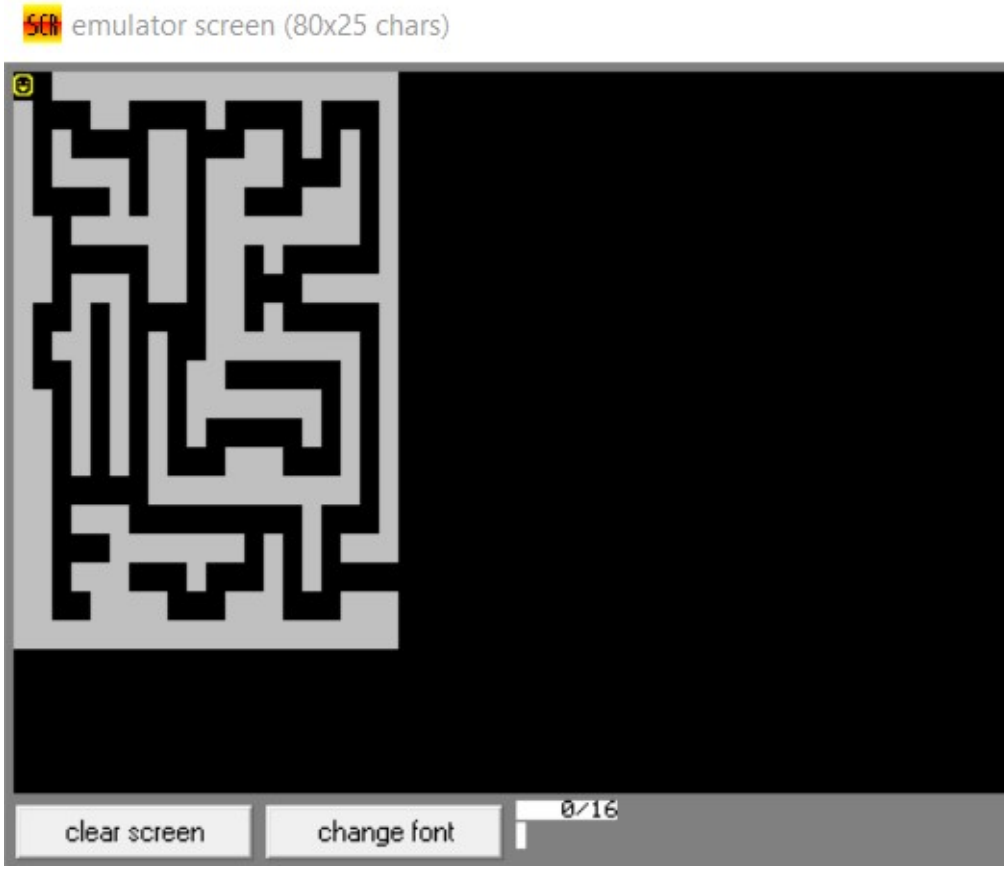

Figure 2.1: User Interface Of Maze Solver

2.3 Implementation

The Maze Solver project is designed to offer an interactive and engaging experience for users seeking to navigate through mazes efficiently. The implementation incorporates various maze-solving techniques, enhancing the overall gaming experience. The primary goal is to showcase the effectiveness and distinctions between different maze-solving algorithms.

2.3.1 Maze Solving Techniques Implemented

The effectiveness of the Maze Solver project relies on the integration of several maze-solving techniques, each contributing to an optimized and enjoyable gaming experience. The key techniques include:

- **Left-Hand Rule Algorithm:** The Left-Hand Rule algorithm is employed as the primary maze-solving technique. It follows a simple rule of always keeping the

left hand in contact with the wall. This algorithm ensures that the solver explores the entire maze systematically, guaranteeing a solution if one exists.

- **Backtracking:**Backtracking is utilized as a secondary technique to handle complex mazes. It involves a recursive approach where the solver explores the maze paths and backtracks when it encounters a dead-end. Backtracking is crucial for solving mazes with multiple possible paths.
- **Backtracking:** Backtracking is utilized as a secondary technique to handle complex mazes. It involves a recursive approach where the solver explores the maze paths and backtracks when it encounters a dead-end. Backtracking is crucial for solving mazes with multiple possible paths.
- **Optimized Movement:** The project incorporates optimized movement functions, allowing the solver to navigate through the maze efficiently. These functions ensure that the solver makes intelligent decisions based on its current position, leading to quicker and more effective maze-solving.
- **Interactive Gameplay:** The maze-solving techniques are integrated into an interactive gameplay loop. Users can control the solver's movements using arrow keys, adding an element of engagement to the solving process.

2.3.2 Tools and Libraries

- Programming Language: Assembly Language 8086.
- Integrated Development Environment (IDE): EMU8086.
- Libraries: "emu8086.inc".

2.4 Algorithms

Algorithm is the prepossess to solve a problem.It is noting but a step by step procedure for the full program.In this algorithm outlines the key steps involved in creating and running the "Maze Solver" game, providing a roadmap for the program's logic and functionality.

Initialization:

1. Set up necessary constants and variables, including maze structure, starting position, and direction.
2. Display welcome message and game rules.
3. Wait for a keypress to start the game.

Draw Maze:

1. Define the maze structure using a 2D array of binary values where 0 represents open space, and 1 represents walls.
2. Implement a print function to display the maze on the screen.

Game Loop:

1. Enter the main game loop. Continuously update the display, player position, and handle user input.
2. Check for arrow key input using BIOS interrupt 16h.
3. Move the player's position based on the arrow key input, updating the display accordingly.
4. Implement a delay to control the speed of the player's movement.

Collision Detection:

1. Check for collisions with walls after each move to prevent the player from moving through walls.
2. If the player reaches the exit point, proceed to the game completion screen.

Collision Detection:

1. Check for collisions with walls after each move to prevent the player from moving through walls.
2. If the player reaches the exit point, proceed to the game completion screen.

Game Completion:

1. Display the total number of moves made by the player.
2. Print a congratulatory message.
3. Halt the program.

Movement Functions:

1. Implement separate functions for moving the player in different directions (left, right, up, down).
2. Check for collisions and adjust the player's position accordingly.

Sound Effects:

1. Implement simple beep sounds for various game events (e.g., hitting a wall, reaching the exit).

User Interface:

1. Use BIOS interrupt 10h to update the screen with the player's position and the maze.
2. Print the total number of moves at the end of the game.

Termination:

1. Halt the program and end the execution.

Congratulations Screen:

1. Display a congratulatory message on successfully completing the maze.
2. Include a graphical representation using ASCII art.

Cleanup:

1. Clear the screen before exiting.

By integrating these algorithmic components, the Maze Solver project delivers a dynamic and efficient solution to maze navigation, providing users with an immersive and educational experience.

Chapter 3

Performance Evaluation

3.1 Simulation Environment/ Simulation Procedure

The simulation environment was developed using Assembly language, specifically utilizing the 8086 and "emu8086.inc" library for creating the map and the functionality behind the game. The environment provides a user-friendly interface for maze solver games. The development environment used for this project includes:

- emu-8086 version 4.08

The simulation environment supports Users can move using arrow and find the path or escape root.

Device

- **Brand**: HP
- **Model Name**: Probook
- **Screen Size**: 14 Inches
- **Colour**: Silver
- **Hard Disk Size**: 256 GB
- **CPU Model**: Core i5 8250U
- **RAM Memory Installed Size**: 8 GB
- **Operating System**: Windows 10 Pro
- **Special Feature**: Thin
- **Graphics Card Description**: Integrated

3.2 Results Analysis/Testing

To ensure the accuracy and effectiveness of the amaze solver project, thorough testing and result analysis were performed. The following sections provide an overview of the testing approach and the results obtained. Discussion about your various results should be included in this chapter in detail.

3.2.1 Result_Case_1

This is the starting page, where you can explore the game rules to better understand how to navigate through the maze. Additionally, you have the option to choose between an easy or hard map for your maze-solving adventure. Take a moment to familiarize yourself with the rules and select your preferred challenge level. Enjoy the maze-solving experience!

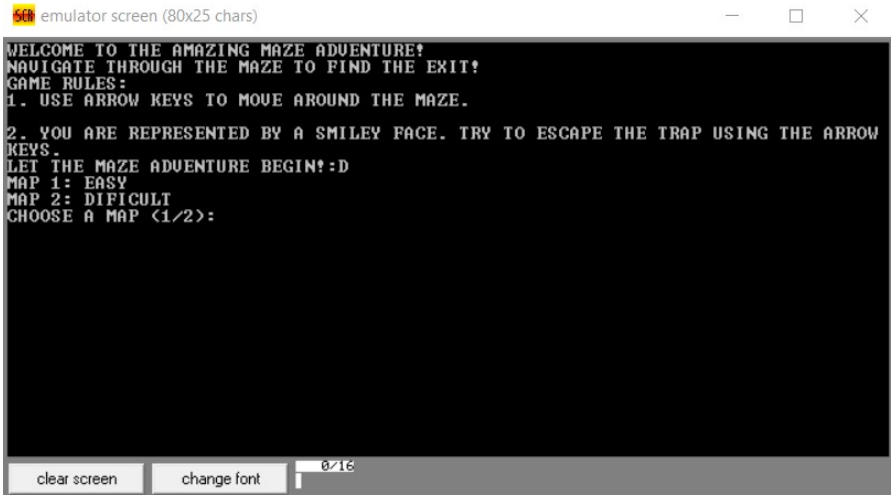

Figure 3.1: Welcome And Rules Page of Maze Solver

3.2.2 Result_Case_2

In this illustration, the player is navigating through Map 1. The yellow path represents the route that the user takes as they progress through the maze.

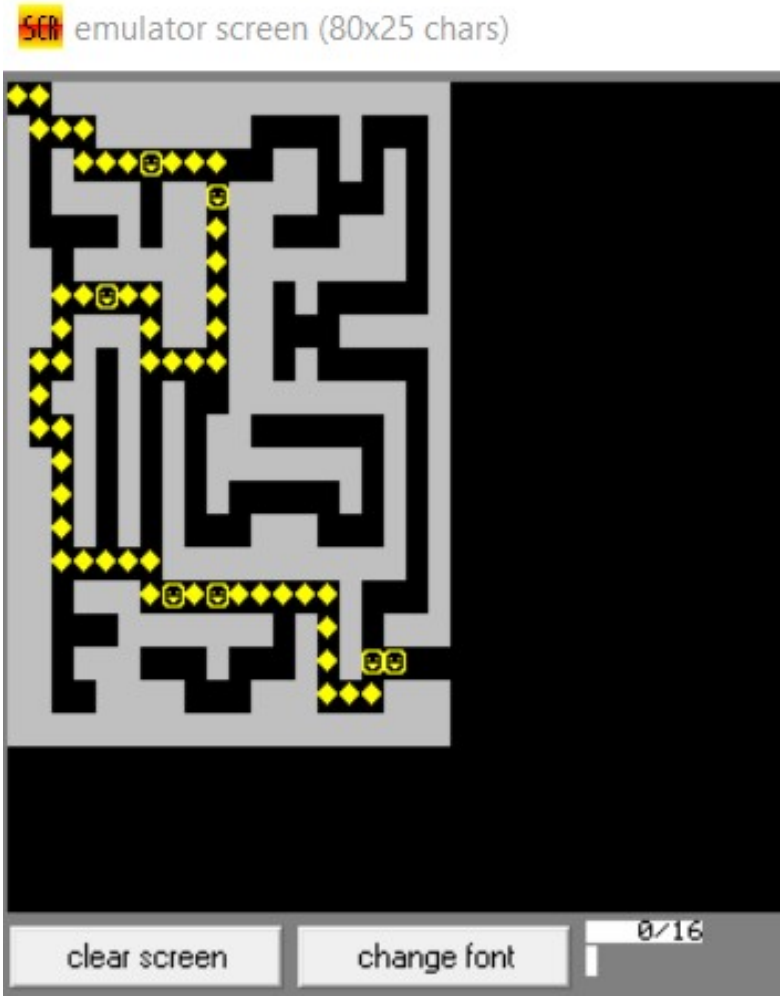

Figure 3.2: Nevigating Through Map 1

3.2.3 Result_Case_3

In this image, we observe Map 2, known for its increased difficulty. The challenge here is to identify the exit using the fewest possible moves. Prepare for a more intricate maze, where strategic decision-making becomes crucial to successfully finding the way out.

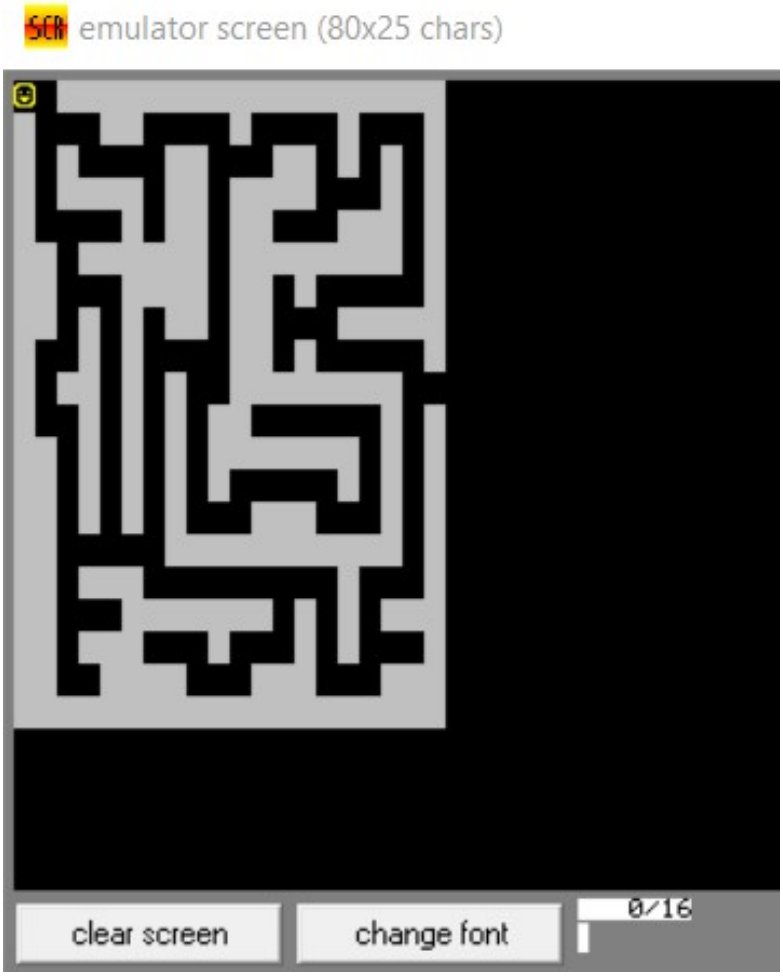

Figure 3.3: Nevigating Through Map 2

3.3 Results Overall Discussion

The Maze Solver project has been successfully implemented, offering a robust and user-friendly tool for navigating through mazes. The accurate coding, thorough testing, and positive user interactions highlight its significance in aiding maze-solving comprehension and fostering an engaging learning experience. With potential future enhancements and broader applications, the Maze Solver could evolve into a versatile educational resource across various domains.

3.3.1 Complex Engineering Problem Discussion

The Maze Solver project encompasses several intricate engineering challenges spanning diverse categories. This has already been mentioned in the Table 1.1.

1. **Depth of Knowledge Required:** Developing a maze-solving algorithm necessitates a profound understanding of various maze-solving techniques, including their implementation specifics and performance attributes. Proficiency in data structures, algorithmic analysis, and programming languages like assembly is crucial for crafting efficient algorithms.

2. **Range of Conflicting Requirements:** The solver must strike a delicate balance between conflicting requirements. On one hand, it needs to provide an accessible and intuitive interface for easy user interaction. On the other hand, it must accurately reflect the maze-solving algorithms' behavior and execution steps. Balancing user-friendliness with precision in visualization poses a significant challenge.

3. **Depth of Analysis Required:** The project demands an in-depth analysis of maze-solving algorithms, considering factors such as time and space complexities, performance across different maze configurations, and the trade-offs between various algorithms. A meticulous understanding of these aspects is crucial for selecting and visualizing the most effective solving strategies.

4. **Extent of Applicable Codes:** Implementation involves substantial code development, ranging from creating the maze-solving algorithms to designing a user interface and visualization components. Crafting efficient, error-resistant code, handling diverse maze structures, and ensuring seamless integration among different modules are essential aspects of the development process.

Overall, The Maze Solver project poses a complex engineering challenge that requires a profound understanding of maze-solving algorithms, the careful management of conflicting requirements, a rigorous analysis of algorithmic intricacies, and extensive code development. Addressing these challenges successfully results in an efficient and user-friendly tool for comprehending and visualizing maze-solving algorithms.

Chapter 4

Conclusion

4.1 Discussion

The Maze Solver project prompts a comprehensive discussion on its implementation, performance, and potential impact.

The implementation successfully tackles the intricacies of maze-solving algorithms, providing users with an effective tool for navigating through various maze configurations. The user interface's intuitive design and the accuracy of algorithmic execution contribute to a positive user experience. Additionally, the incorporation of diverse solving strategies adds versatility to the tool, catering to different user preferences and maze complexities. User feedback and testing results affirm the project's success in achieving its primary objectives.

However, discussions also arise regarding potential improvements and areas for refinement. For instance, fine-tuning certain algorithmic parameters or exploring additional maze generation techniques could enhance the solver's overall performance. Additionally, addressing user suggestions and incorporating alternative visualization methods might further improve the educational aspects of the tool. The discussion section serves as a platform for evaluating the project's strengths and identifying avenues for enhancement.

4.2 Limitations

The Maze Solver project, while successful, is not without its limitations, signaling opportunities for future refinement and enhancement.

- **Limited Maze Variety:** The maze is hardcoded and does not offer much variety. In a more advanced version, the ability to generate random mazes or load different maze configurations would enhance the replayability of the game.
- **Single Maze:** The game is designed with a single maze. Adding multiple levels or different difficulty levels with varying maze complexities could make the game more challenging and interesting.

- **No Score System:** The game lacks a scoring system or any form of progress tracking. Incorporating a scoring mechanism, such as time taken to solve the maze, could add a competitive element and motivate players to improve their performance.
- **Limited Instructions:** While there are instructions displayed at the beginning of the game, there is no in-game help or guidance. Adding features like hints, a tutorial, or a help menu could assist players who may be unfamiliar with the game.

4.3 Scope of Future Work

The Maze Solver project holds promising avenues for future development and expansion, providing a foundation for further exploration and improvement.

- **Dynamic Maze Generation:** Implement algorithms for dynamically generating mazes, ensuring that each playthrough offers a unique challenge. This would increase the game's replayability and provide a more dynamic experience.
- **Multiple Levels:** Extend the game to include multiple levels with increasing difficulty. Each level could have a different maze layout, adding complexity and maintaining player interest.
- **Scoring and Leaderboard:** Introduce a scoring system based on factors like time taken, number of moves, and level difficulty. Implement a leaderboard to encourage competition among players.
- **Multiplayer Mode:** Implement a multiplayer mode that enables players to compete against each other or collaborate to solve mazes. This would add a social dimension to the game.
- **Bug Fixes and Code Optimization:** Continuously improve the code by addressing any existing bugs, enhancing code efficiency, and adopting best practices for software development.

References

Please Visit the Following Link For More Information:

- https://www.researchgate.net/publication/224202469_A_Comprehensive_and_Comparative_Study_of_Maze-Solving_Techniques_by_Implementing_Graph_Theory
- https://gurugio.gitbooks.io/book_assembly_8086/content/introemu8086.html