DATA STRUCTURE AND ALGORITHM LAB

SEMESTER PROJECT

BSE-3A

Project Name:- Maze Quest: Treasure Hunt

NAME	Enrolment
Samreen Farhat	01-131232-079
Umama Hidayat Khan	01-131232-107

GitHub Link: https://github.com/UmamaK47/Maze-Quest-Treasure-Hunt.git

Department of Software Engineering Bahria University H11 Campus,
Islamabad.

Table of Contents

Maze Quest: Treasure Hunt

---

1. Abstract:

Maze Quest is a maze-navigation game where players must find their way to a treasure in a randomized maze. The game has three difficulty levels, a timer, and a limit on the number of times players can undo moves. It is developed in C++ using SFML, using data structures like stacks for player movement and graphs for maze generation using Prim's Algorithm.

---

2. Introduction:

Motivation: The idea came from the fun and challenge of solving mazes. The goal was to create a game that could be replayed many times with different mazes.
Gameplay: Players control a character to navigate through a randomly generated maze using keyboard inputs.
Objective: The player must reach the treasure at the end of the maze before running out of time or undo moves.

---

3. Game Design:

3.1. Concept:

The game is about solving mazes and finding the shortest path to the treasure. Random maze layouts make every game different.

3.2. Mechanics:

- Players move using arrow keys.
- A countdown timer adds pressure.
- Undo functionality allows limited backtracking.

3.3. Game Flow:

- **Main Menu:**
  - Start Game
  - How to Play
  - Exit
- Transitions between menus, gameplay, and end screens are smooth.

3.4. Difficulty Modes:

Three levels of difficulty:

- **Easy:** 45 seconds, 20 undos.
- **Medium:** 30 seconds, 10 undos.

- **Hard:** 20 seconds, 5 undos.

3.5. Visuals:
The maze is simple with black-and-white graphics. The player and treasure are easy to identify.

3.6. Sound:

- Background music runs throughout the game.
- Sounds play for important events like starting the game or reaching the treasure, running out of time or undo moves.

---

4. Technical Implementation:

4.1. Tools Used:
The game is made in C++ using SFML for graphics and handling inputs.

4.2. Data Structures:

- **Graphs:** The maze is represented as a graph, where each cell is a node, and paths between cells are edges. Prim's Algorithm is used to generate a unique and solvable maze layout.
- **Stacks:** A stack is used to track the player's path, enabling the undo feature. When the player undoes a move, the previous position is popped from the stack.

4.3. Maze Generation:
Randomized Prim's Algorithm is used to create unique and solvable mazes.

4.4. State Management:
Game states like Main Menu, Gameplay, and How to Play are managed using enums.

4.5. Timer:
A countdown timer is shown as a bar on the screen, decreasing with time.

4.6. Movement and Collision:
Players can only move on valid paths. Walls block movement.

4.7. Undo Feature:
A stack is used to store moves for undoing, with a limit based on difficulty.

4.8. Assets:
Textures and sounds are loaded and scaled properly.

---

5. Gameplay Mechanism:

5.1. Player Movement:
Players use arrow keys to move through the maze. Undoing a move uses one of the allowed undos.

5.2. Winning:
The game is won when the player reaches the treasure before time or moves run out.

5.3. Losing:
The game is lost if time runs out or there are no undos left.

5.4. Replayability:
Randomized mazes ensure that no two games are the same.

---

6. Challenges and Solutions:

6.1. Challenges:

- Creating random mazes efficiently.
- Keeping player movement smooth.
- Making sure the timer works with all events.

6.2. Solutions:

- Used Prim's Algorithm for efficient maze generation.
- Used SFML features for smooth controls.
- Synced the timer with game logic for accuracy.

---

7. Testing and Debugging:

Testing:

- Features like maze generation, player movement, and state transitions were tested.
- Boundary cases like moving into walls or running out of moves were checked.

---

8. Conclusion:

Maze Quest is a simple and fun maze-solving game. It challenges players with time limits and limited undo moves, offering a new maze every time. The game effectively demonstrates the use of graphs and stacks for problem-solving

---

9. References:

- **Libraries**: SFML for graphics and event handling.
- **Resources**: Online tutorials on maze generation algorithms and game state management.
- **Assets**: Free-to-use textures, fonts, and sound effects.

---

10. Appendices:

10.1. Pseudocodes:

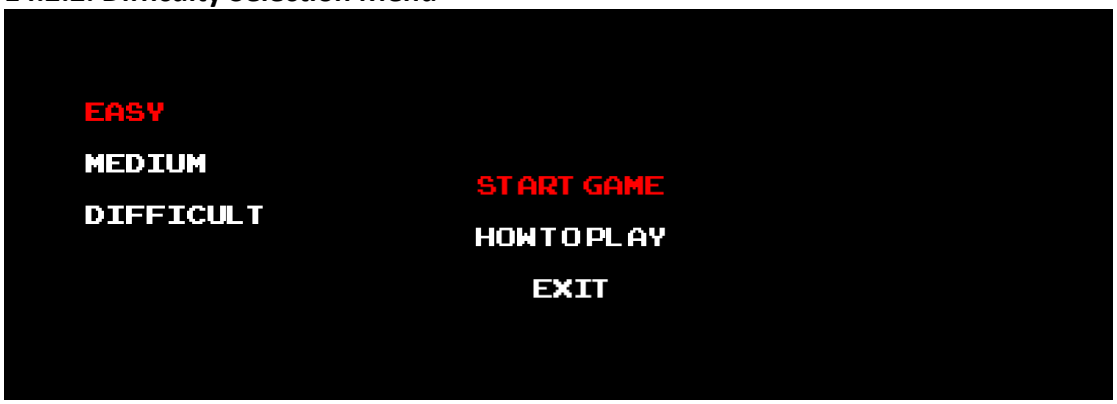
14.1.1: Maze Generating using Randomized Prims Algorithm:

```
Initialize grid with nodes, each having all four walls intact
Set the starting node as visited
Add neighbors of the starting node to the frontier list

WHILE frontier is not empty DO:
  Randomly select a node `currentNode` from the frontier list
  Remove `currentNode` from the frontier list

  Get a random `visitedNeighbor` of `currentNode`
  IF `visitedNeighbor` exists THEN:
    Remove the wall between `currentNode` and `visitedNeighbor`

  Mark `currentNode` as visited
  Add all unvisited neighbors of `currentNode` to the frontier list IF
not already present

END WHILE

Output the generated maze
```

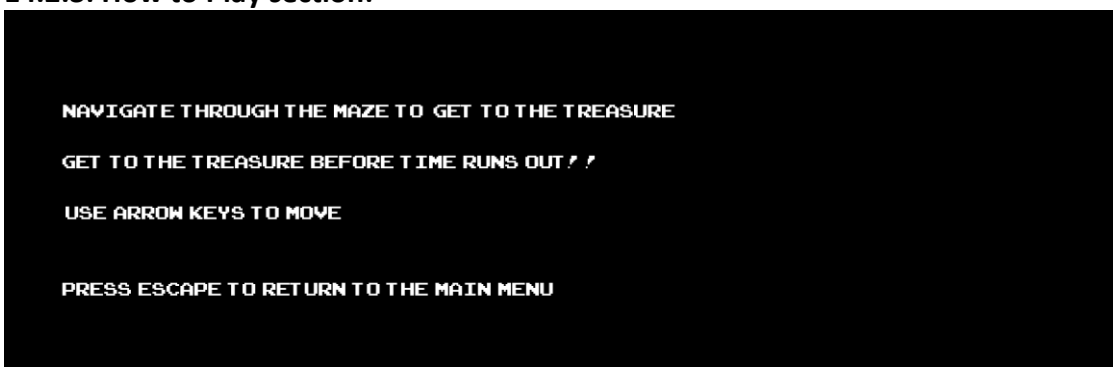
10.2. **Screenshots:**

14.2.1: Main Menu

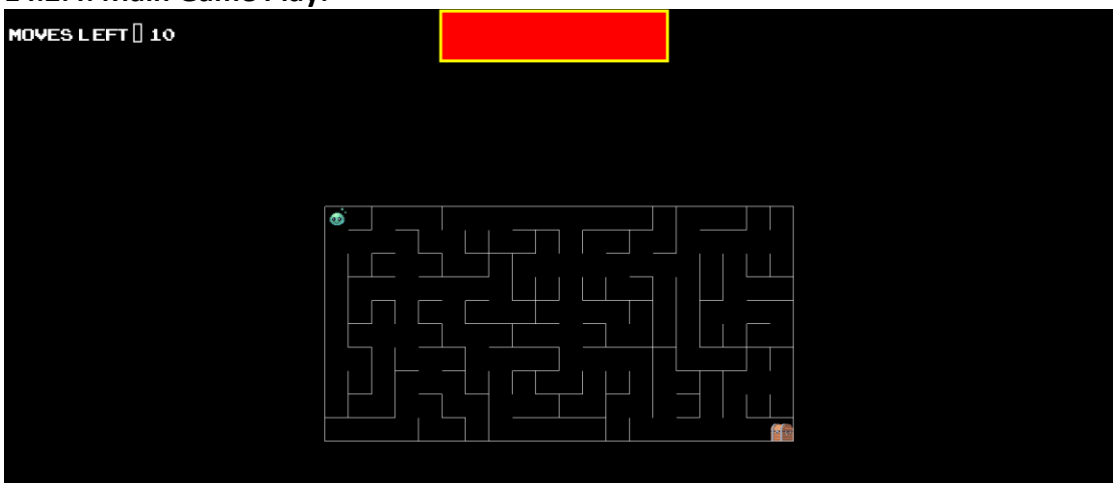
14.2.2: Difficulty Selection Menu



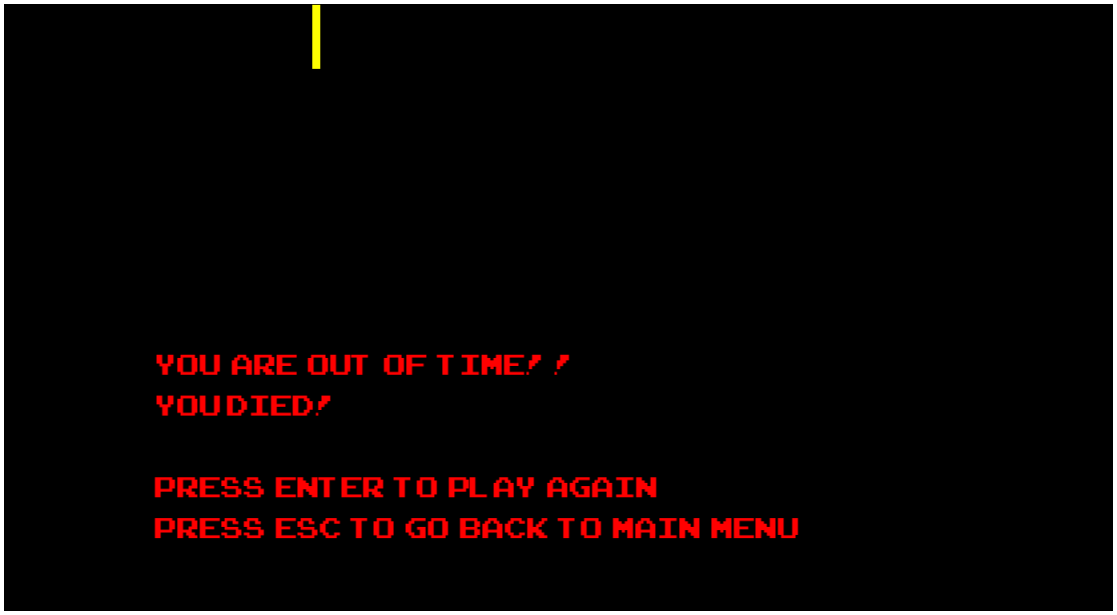
14.2.3: How to Play section:



14.2.4: Main Game Play:



14.2.5: Out of Time screen:



14.2.6: Win Screen:

10.3. Assets and Resources:

1. **Character Sprite:**
    - **Purpose:** Represents the player character in the game. It is animated for movement and interactions with the maze.
    - **Source**: https://craftpix.net/freebies/free-slime-mobs-pixel-art-top-down-sprite-pack/?num=1&count=8&sq=slime&pos=3
2. **Treasure Box Sprite:**
    - **Purpose:** A visual representation of the treasure at the end of each maze level. The player needs to reach it to win.
    - **Source:** https://craftpix.net/freebies/free-pixel-art-icons-for-mine-location/?num=1&count=1799&sq=gemstone&pos=4
3. **Game Font:**
    - **Purpose:** A pixelated font used for in-game text, such as time remaining, or game-over messages.
    - **Source:** https://www.1001freefonts.com/arcade-classic.font
4. **Background Music:**
    - **Purpose:** The background music that plays throughout the game, creating atmosphere and immersion.
    - **Source:** https://pixabay.com/sound-effects/game-music-loop-6-144641/
5. **Game Start Sound Effect:**
    - **Purpose:** A sound effect played when the game starts, helping to set the tone and provide feedback to the player.
    - **Source:** https://pixabay.com/sound-effects/game-start-6104/
6. **Treasure Collect Sound Effect:**
    - **Purpose:** A sound played when the player collects the treasure box, indicating success and progression.
    - **Source**: https://pixabay.com/sound-effects/achievement-video-game-type-1-230515/
7. **Out of Time/Moves Sound Effect:**
    - **Purpose:** Played when the player runs out of time or moves, indicating a game-over or failure state.