Project Title: *Memory Maze: A Strategic 2D Maze Game*

Submitted By: Muhammad Umer Fazal

Course: AI

Instructor: Mehak Mazhar

Submission Date: May 04, 2025

---

1. Project Overview

- **Project Topic**:
  The project centers around the development of *Memory Maze*, a 2D strategic maze game that combines memory mechanics with pathfinding. Players will navigate through a maze, avoiding traps, collecting bonuses, and racing to reach the goal. The game introduces a unique feature where players must use memory tokens to reveal parts of the maze temporarily, helping them navigate more efficiently. The game will be designed for two players with the possibility of adding an AI opponent.
- **Objective**:
  The primary goal of this project is to develop an engaging and strategic game that challenges players to use memory and decision-making skills. Additionally, the project will focus on implementing an AI that can compete against human players, leveraging algorithms such as Minimax for decision-making and pathfinding.

---

2. Game Description

- **Original Game Background**:
  Memory Maze is an original game concept inspired by classic maze games but with a strategic twist. The game's core mechanics are based on players navigating through a grid, avoiding traps, and finding the goal. In the traditional game of maze navigation, players would follow a set path or randomly explore. *Memory Maze* enhances this by introducing the need for players to reveal parts of the maze temporarily using memory tokens, adding complexity and strategy to the decision-making process.
- **Innovations Introduced**:
  - **Memory Tokens**: Players have a limited number of tokens that allow them to temporarily reveal surrounding tiles in the maze. This adds a memory

challenge where players must remember the layout of the maze to avoid traps and find the best path toward the goal.

- **Traps and Bonuses**: Traps are scattered throughout the maze, and hitting a trap decreases a player's ability to continue. Bonuses are placed randomly and provide benefits such as additional memory tokens.
- **Game Progression**: Players must navigate the maze while managing the use of memory tokens and avoiding traps. The first player to reach the goal wins, but they must also avoid being eliminated by triggering too many traps.

These innovations impact gameplay by requiring strategic decision-making. Players must balance their exploration and token usage, deciding when to risk moving without full information or using a token to gain an advantage. The introduction of traps forces players to be cautious, and the bonuses allow for short-term advantages.

---

3. AI Approach and Methodology

- **AI Techniques to be Used**:
  - **Minimax Algorithm**: The AI will utilize the Minimax algorithm to simulate optimal moves. Since the game involves both strategic navigation and memory, the AI will need to plan its moves considering not only the immediate surroundings but also the long-term strategy.
  - **Alpha-Beta Pruning**: This will be used to optimize the Minimax algorithm, pruning unnecessary branches of the decision tree and reducing the computation time.
  - **Reinforcement Learning** (optional): If time permits, we could explore Reinforcement Learning techniques to train the AI to improve based on past experiences and optimize its pathfinding strategy.
- **Heuristic Design**:
  The heuristic for evaluating game states will be based on several factors:
  - **Proximity to the Goal**: The AI will prioritize moves that bring it closer to the goal.
  - **Trap Avoidance**: A penalty will be applied to states where the AI is close to triggering traps.

    - **Memory Token Usage**: The AI will aim to use memory tokens wisely, prioritizing tile reveals that provide strategic information (such as paths leading to the goal).

- **Complexity Analysis**:
  The time complexity of the Minimax algorithm is O(bd)O(b^d)O(bd), where bbb is the branching factor (number of possible moves at each step) and ddd is the depth of the game tree. In a game like *Memory Maze*, the branching factor can be relatively high due to multiple possible moves in a grid, and the depth may depend on the number of turns the game lasts. The introduction of memory tokens and traps may further complicate the evaluation function, increasing the computational complexity.

---

4. Game Rules and Mechanics

- **Modified Rules**:
    - **Memory Tokens**: Each player starts with 2 memory tokens, which can be used to temporarily reveal adjacent tiles. A token is consumed with each use.
    - **Traps**: If a player lands on a tile with a trap, they lose one of their life points. A player is eliminated if they hit a maximum of 3 traps.
    - **Goal**: The goal is located at the center of the board, and the first player to reach it wins. The player must avoid traps while managing memory tokens to reveal useful information.
- **Winning Conditions**:
  A player wins by reaching the goal tile. However, a player is eliminated if they hit 3 traps, which also causes the other player to win automatically.
- **Turn Sequence**:
  Each player takes turns moving in the grid. The player can move up, down, left, or right. Players can use memory tokens to reveal adjacent tiles. If a player lands on a trap, their turn is over, and their position is reset. If a player hits 3 traps, they are eliminated.

---

5. Implementation Plan

- **Programming Language**: Python

- **Libraries and Tools**:
    - **Pygame**: For graphical user interface and rendering the game grid.
    - **NumPy**: For data handling and managing the grid's state.
    - **Minimax Algorithm**: For AI decision-making.
    - **Optional**: TensorFlow or PyTorch (if Reinforcement Learning is implemented).
- **Milestones and Timeline**:
    - **Week 1-2**: Game design and rule finalization. Develop the grid system and basic game mechanics (movement, traps, bonuses, goal).
    - **Week 3-4**: AI strategy development (Minimax algorithm, heuristics for trap avoidance and token usage).
    - **Week 5-6**: Coding and testing the game mechanics (player movement, token usage, trap handling).
    - **Week 7**: AI integration and testing. Implement Minimax with Alpha-Beta pruning for decision-making.
    - **Week 8**: Final testing, debugging, and report preparation.

---

6. References

- "Artificial Intelligence: A Modern Approach" by Stuart Russell and Peter Norvig
- "Algorithms in Python" by Robert Sedgewick
- Pygame Documentation: https://www.pygame.org/docs/