

Egypt University of Informatics

Computer and Information Systems

Artificial Intelligence Course

Introduction to Artificial Intelligence Project

Submitted by:

Ali Wael Hallaba (22-101000)

Omar Elhouseiny (22-101165)

Yousief Khadr (22-101148)

Supervised by:

Dr. Mohamed Taher El Refaei

Eng. Tukka Mohammed

5/1/2025

# Introduction:

The Maze Game is an interactive application developed using Python's Tkinter library, allowing players to navigate a randomly generated maze, collect treats, and compete against an automated agent that uses the A\* algorithm. It combines game design, algorithm implementation, and user interaction for an engaging experience.

# Features:

1. **Maze Generation**: Generates mazes of varying difficulty (beginner, mid, hard) with paths, walls, and randomly placed treats.
2. **User Interaction**: Players control movement using arrow keys, with scores based on treats collected and steps taken.
3. **Agent Solver**: An automated agent solves the maze using the A\* algorithm, allowing performance comparison with the player.
4. **Scoring System**: Tracks cumulative scores reflecting total treats collected across rounds, providing performance feedback.
5. **Graphical Interface**: Simple interface visually represents the maze, player position, treats, and walls.
6. **Step Tracking**: Tracks player steps, encouraging efficient movement strategies.
7. **Game Reset and Replay**: Players can reset or replay the game after completion for continuous engagement.

# Technical Implementation:

1. **Maze Generation**

Creates a grid with **1** (walls), **0** (paths), and **'T'** (treats), ensuring start and end points are paths and treats are placed randomly.

1. **Solvability Check**

Uses a depth-first search (DFS) algorithm to ensure the maze is solvable, regenerating if necessary.

1. **A\* Pathfinding Algorithm**

Finds the optimal path using a heuristic (Manhattan distance) and a priority queue, with a reconstruction method for the final path.

1. **User Movement**

Players move using arrow keys, with checks for valid moves and updates to position, steps, and treats collected.

1. **Agent Movement**

The agent automatically solves the maze using the A\* algorithm, animating its movement and collecting treats.

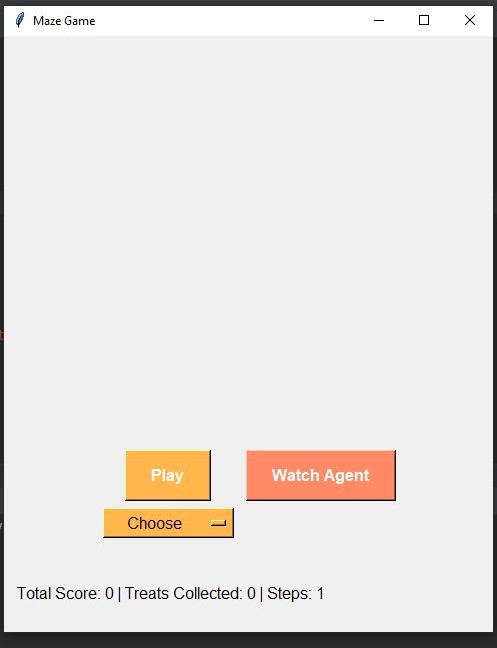
1. **User Interface**

Built with Tkinter, featuring a canvas for the maze, buttons for game control, a difficulty dropdown, and a score label

# Technical Challenges:

1. **Syncing Gameplay with the GUI:** Achieving real-time synchronization between the game state and the GUI was difficult, leading to discrepancies in the displayed game state.
2. **Non-Optimal Pathfinding by the Agent:** The automated agent sometimes took non-optimal paths due to inefficiencies in the pathfinding algorithm, necessitating optimization.
3. **Generation of Unsolvable Mazes:** The maze generation occasionally resulted in unsolvable configurations, requiring validation checks to ensure at least one valid path existed.
4. **User Input Responsiveness:** Capturing and processing user inputs in real-time was essential for smooth gameplay, demanding effective event handling.
5. **Debugging Complex Interactions:** Troubleshooting bugs related to maze generation and player movement involved extensive testing, often revealing unexpected interactions between components**.**

# Visuals:



Here we start the program. The user gets to choose either they want to play or watch the AI agent play. They also get to pick the level they want from Easy, Mid, or Hard.

A screenshot of a game

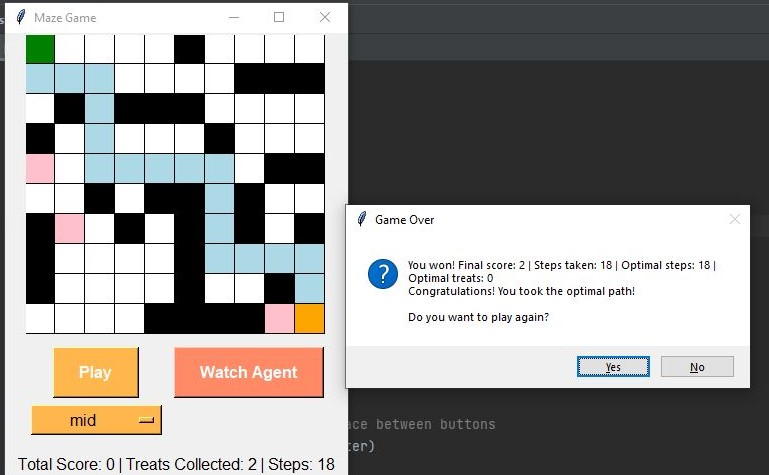
Description automatically generated

Here the user chose to watch the agent play. After the agent solves the maze successfully, it shows the user how many treats the agent collected and its score.

A screenshot of a crossword puzzle

Description automatically generated

The user is now playing. They hit a wall so the program warns them that they should take another move.



The user won!! When the user wins the program shows them their score and the steps taken, and the optimal steps for this maze (calculated by A\* algorithm). Also, the program asks them if they want to play again to either reset the score or continue counting as the player gets treats.

A screenshot of a crossword puzzle

Description automatically generated

Here, compared to the A\* optimal steps for this maze, the player didn’t take the optimal path. This doesn’t mean they didn’t win the game by reaching the end point. So the program tells them they won, but could’ve took a better path.

# Conclusion:

The Maze Game effectively combines game mechanics with algorithmic problem-solving, providing an engaging and educational experience. It showcases random maze generation, pathfinding algorithms, and user interface design in Python. The project serves as a valuable tool for education and entertainment, demonstrating the application of programming concepts in a fun and interactive way.