**Project Report**

Project Title: Tower Trek: Escape the Pursuer - An AI-Powered Grid-Based Strategy Game

Submitted By: Syed Aadil Ahmed 22k4339, Mohsin Manzur 22k4370

Course: AI

Instructor: Miss Alishba Subhani

Submission Date: 11/5/25

1. Executive Summary

● Project Overview:

This project involved the development of "Tower Trek: Escape the Pursuer," a grid-based strategy game where players navigate through a tower while avoiding an AI-controlled pursuer. The main objective was to create an engaging game with intelligent AI pathfinding that actively hunts the player. The game utilizes the A\* pathfinding algorithm to enable the AI to make strategic decisions when pursuing the player, creating a challenging and dynamic gameplay experience.

2. Introduction

● Background:

Grid-based chase games have been a staple in the gaming industry for decades, from classics like Pac-Man to modern roguelike games. Tower Trek builds on this tradition by implementing a turn-based movement system on a grid with obstacles, where the player must strategically navigate to reach a goal while avoiding an intelligent AI pursuer. This project was selected to explore pathfinding algorithms in a practical, interactive application.

● Objectives of the Project:

- Develop a fully functional grid-based strategy game with pygame

- Implement an intelligent AI pursuer using the A\* pathfinding algorithm

- Create multiple difficulty levels that affect obstacle density

- Design a level progression system with increasing challenge

- Provide a user-friendly interface with clear visual feedback

3. Game Description

● Original Game Concept:

Tower Trek is inspired by classic chase games but implements a turn-based approach where the player and AI alternate moves. The game is set on a grid representing levels of a tower, with the player starting at the bottom left and attempting to reach the goal at the top right, while avoiding the AI pursuer.

● Innovations and Modifications:

The game features randomly generated obstacles for each level, ensuring a unique experience each time. The difficulty increases progressively, with more obstacles appearing at higher levels and difficulty settings. The AI pursuer uses sophisticated pathfinding to track and intercept the player, creating a challenging strategic experience.

4. AI Approach and Methodology

● AI Techniques Used:

The game employs the A\* (A-star) pathfinding algorithm to enable the AI pursuer to find the optimal path to the player. This algorithm combines the advantages of Dijkstra's algorithm (guaranteeing the shortest path) and greedy best-first search (using heuristics to speed up the search).

● Algorithm and Heuristic Design:

The A\* implementation uses a Manhattan distance heuristic to estimate the distance between the AI and the player. The algorithm maintains open and closed sets of nodes, calculating f(n) = g(n) + h(n) for each node, where g(n) is the cost from the start node and h(n) is the heuristic estimate to the goal. This allows the AI to efficiently find the shortest path to the player while avoiding obstacles.

● AI Performance Evaluation:

The AI's performance was evaluated based on its ability to catch the player across different difficulty levels and maze configurations. The pathfinding algorithm consistently finds the optimal path when one exists, and the AI employs fallback strategies when direct pathfinding fails, ensuring it always makes progress toward the player.

5. Game Mechanics and Rules

● Game Rules:

- The player and AI take turns moving one square at a time (up, down, left, or right)

- The player must reach the goal (top right) to complete a level

- If the AI catches the player (occupies the same square), the game ends

- Obstacles block movement for both the player and AI

- Each level has a randomly generated layout of obstacles

● Turn-based Mechanics:

The game follows a strict turn-based system where the player moves first, followed by the AI. After each player move, the game checks if the player has reached the goal. If not, the AI makes its move, and the game checks if the AI has caught the player.

● Winning Conditions:

The player wins a level by reaching the goal position. The overall game progresses through increasingly difficult levels, with the difficulty increasing every three levels.

6. Implementation and Development

● Development Process:

The game was developed using an iterative approach, starting with the basic grid and movement mechanics, then adding the AI pathfinding, and finally implementing the game states and user interface. The A\* algorithm was implemented from scratch to ensure it met the specific requirements of the game.

● Programming Languages and Tools:

○ Programming Language: Python

○ Libraries: Pygame for graphics and input handling, heapq for priority queue implementation in A\*

○ Tools: Visual Studio Code for development, Git for version control

● Challenges Encountered:

- Ensuring the randomly generated levels always had a valid path from player to goal

- Optimizing the A\* algorithm for real-time performance on larger grids

- Implementing effective fallback strategies for the AI when no direct path was available

- Balancing difficulty across levels to create a challenging but fair experience

7. Technical Implementation Details

● Grid System:

The game uses a 2D grid system represented as a 2D array of Cell objects. Each cell tracks its position, whether it contains an obstacle, and special properties like being a goal or start position.

● Pathfinding Implementation:

The A\* pathfinding algorithm was implemented using a priority queue (via heapq) to efficiently select the next node to explore. The algorithm maintains:

- An open set of nodes to be evaluated

- A closed set of already evaluated nodes

- A came\_from map for path reconstruction

- g\_score and f\_score maps for node evaluation

● Collision Detection:

The game implements simple collision detection by comparing the coordinates of the player and AI after each move. When they match, the game transitions to the Game Over state.

8. Results and Discussion

● AI Performance:

The AI pursuer demonstrates effective pathfinding, consistently finding the shortest path to the player when one exists. On higher difficulty levels with more obstacles, the AI still manages to navigate efficiently, creating a challenging experience for the player. The fallback movement strategies ensure the AI always makes progress toward the player, even in complex maze configurations.

● Game Balance:

The progressive difficulty system creates a well-balanced experience, with early levels being more forgiving and later levels requiring more strategic thinking. The random generation of obstacles ensures replayability, as each playthrough presents new challenges.

9. Conclusion and Future Work

● Conclusion:

Tower Trek successfully implements an engaging grid-based strategy game with intelligent AI pathfinding. The A\* algorithm provides the AI pursuer with effective decision-making capabilities, creating a challenging and dynamic gameplay experience.

● Future Improvements:

- Implementing power-ups that temporarily affect player or AI movement

- Adding multiple AI pursuers for increased difficulty

- Creating special grid elements like teleporters or one-way passages

- Implementing a high score system to track player performance

- Adding a timer to create additional pressure on the player

10. References

- Millington, I., & Funge, J. (2009). Artificial Intelligence for Games. CRC Press.

- Pygame Documentation: https://www.pygame.org/docs/

- Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Transactions on Systems Science and Cybernetics, 4(2), 100-107.

- Patel, A. (2021). Introduction to the A\* Algorithm. Retrieved from https://www.redblobgames.com/pathfinding/a-star/introduction.html