Labyrinth AI – Maze Runner with Dynamic Obstacles

Submitted By: Hafsa Atiqi 22k-4584, Nawal Salman 22k-4236, Raaiha Syed 22k-4460
Course: AI Lab
Instructor: Talha Shahid
Submission Date: 20 April 2025

---

1. Project Overview

● Project Topic:
We propose a strategic AI adaptation of the board game *Labyrinth*, where players compete to reach specific targets on a shifting maze. Our version introduces dynamic AI-controlled traps that react to player movement and adds adversarial tile shifting to block opponents. Players must plan optimal paths while simultaneously manipulating the maze to their advantage or to hinder others.

● Objective:
To develop a smart AI agent capable of:

- Determining the optimal path using **A*** in a constantly changing environment.
- Strategically shifting maze tiles to block or slow opponents using **adversarial search** (Minimax).
- Reacting to dynamic "trap" obstacles and making real-time decisions in a multi-agent setting.

---

2. Game Description

● Original Game Background:
Labyrinth is a tile-based board game where players slide rows or columns of tiles to shift paths on a grid, aiming to collect treasures or reach goals. Each turn consists of a tile shift followed by a player movement, with the board changing constantly.

● Innovations Introduced:

- **Dynamic AI-Controlled Traps:** Traps are randomly activated and change positions based on players' proximity.
- **Adversarial Tile Movement:** Players use their turns not only to progress but to *strategically block* opponents.
- **Target Rerouting:** Targets may shift to different locations mid-game, increasing complexity.

● Impact on Complexity & Strategy:
These innovations make the game non-deterministic and highly dynamic, requiring adaptive strategies. AI must constantly reevaluate paths, anticipate future board states, and predict opponent behavior.

---

3. AI Approach and Methodology

● AI Techniques to be Used:

- **A*** for real-time pathfinding in dynamic maze layouts.
- **Minimax Algorithm** with modifications for multi-agent decision making and tile-shifting strategy.
- **Alpha-Beta Pruning** to optimize Minimax performance.
- **Optional Reinforcement Learning** (RL) for training an AI to adapt to traps over time.

● Heuristic Design:

- **Path Cost Heuristic (A*):** Estimate distance to target based on current and projected maze states.
- **Blocking Score:** Evaluate how effective a tile shift is at slowing an opponent.
- **Trap Avoidance Metric:** Adjust utility score if a trap lies within the predicted path.

● Complexity Analysis:

- **A* Pathfinding:** O(b^d), where b = branching factor, d = path depth (frequent due to dynamic tiles).
- **Minimax with Alpha-Beta:** Exponential in depth but optimized through pruning.
- **Main Challenge:** AI must handle both adversarial shifts and frequent state changes from traps and opponents.

---

4. Game Rules and Mechanics

● Modified Rules:

- Each player gets 2 actions per turn:
    1. Shift one row or column to change the maze.
    2. Move toward their goal using updated paths.
- Traps activate every 3 rounds and relocate based on players' positions.
- If a player steps into a trap, they lose their next turn.

● Winning Conditions:

- First player to reach **three assigned goal tiles** wins the game.
- Stepping into a trap or getting blocked delays progress.

● Turn Sequence:

- Players take turns in clockwise order.
- On each turn:
    1. Shift a maze tile (any row or column except the one just shifted).

2. Move the pawn.
3. Traps are re-evaluated after every full round.

---

5. Implementation Plan

● **Programming Language:** Python

● **Libraries and Tools:**

- **Pygame** – for GUI and interactive game interface
- **NumPy** – for board matrix manipulation
- **NetworkX or custom graphs** – for A* pathfinding
- **Optional:** TensorFlow or Scikit-learn if RL is implemented later

● **Milestones and Timeline:**

- **Week 1-2:** Game design finalization and maze logic implementation
- **Week 3-4:** Develop A* pathfinding and trap logic
- **Week 5-6:** Integrate Minimax for tile-shifting decisions
- **Week 7:** Full game integration and AI testing
- **Week 8:** Final refinements, testing, and report/documentation

---

6. References

- Official *Labyrinth* game rules and variants
- Online resources and documentation for Pygame and NumPy