Project Report Template

Project Title: AI-Powered Maze Escape Game
Submitted By: 22k-4319 Uzain Ahmed
22k-4492 Syed Muhammad Bilal
22k-4143 Aadil Raja
Course: AI
Instructor: Mr. Muhammad Khalid Khan
Submission Date: 10th,May,2025

1. Executive Summary

- **Project Overview:**
  This project presents AI-Powered Maze Escape, an evolved, dynamic, and competitive maze-based puzzle game where a human player competes against an AI-controlled opponent to reach the exit of an intelligently transforming maze. Initially proposed as a single-player experience involving dynamic obstacles and AI-assisted pathfinding similar to Pacman, the game has now been enhanced into a head-to-head AI vs Player challenge, introducing deeper AI mechanics, strategy elements, and gameplay variety.

  The main objective is to build an engaging and intelligent system where both entities—player and AI—navigate a dynamically evolving maze, utilizing search algorithms and strategic tools to reach the exit before the other. The AI opponent makes decisions using the A (A-Star) heuristic search algorithm, with an adaptive intelligence parameter ranging from 0 to 1. This intelligence level increases with each level, resulting in a smarter and faster AI adversary as the game progresses.*

  In addition to pathfinding and dynamic obstacle handling, the game now incorporates gameplay-enhancing mechanics such as:

  - *Power-ups and special power-ups to provide temporary advantages to the player.*
  - *Sabotage tools (e.g., confuse or freeze the AI) allowing the player to interfere with AI progress.*
  - *Traps, which are designed to affect only the player and increase maze difficulty.*

- *Time-limited levels, maintaining a pressure element consistent with the original design.*

- *AI-generated hints, helping players when they are stuck or disoriented.*

The maze environment itself remains dynamic and AI-controlled, with real-time obstacle adjustments based on the player's movement patterns, ensuring that the game stays challenging and non-repetitive.

2. Introduction

- **Background:**

  The AI-Powered Maze Escape game was initially conceptualized as a single-player puzzle experience, drawing inspiration from classic maze-based games such as Pac-Man. In its conventional form, the game involved a player navigating a maze while avoiding AI-controlled obstacles and trying to reach the exit within a limited time. It emphasized dynamic pathfinding, adaptive challenges, and AI-generated hints to assist the player in difficult situations.

  This familiar and intuitive maze format was chosen due to its versatility and strong potential for AI integration. Maze environments naturally lend themselves to search algorithms and decision-making challenges, making them ideal for applying and testing various AI techniques such as BFS, DFS, A, and Greedy Best-First Search.*

  As the project progressed, the concept evolved from a solo escape to a more competitive and strategic format. The game now features a head-to-head race between a human player and an AI opponent, each trying to find the optimal path to the goal. To enrich gameplay and introduce tactical depth, several new elements were incorporated:

  - *Intelligence-scaled AI (0 to 1): AI difficulty scales across levels, simulating different behavior from clumsy to highly intelligent.*

  - *Power-ups and special abilities: Enhance the player's chance of winning through temporary speed or visibility boosts.*

  - *Sabotage mechanics: Players can freeze, confuse, or delay the AI opponent strategically.*

  - *Traps: Maze sections that are only hazardous to the player, demanding careful planning.*

- *Time pressure: A countdown timer adds urgency, reinforcing the puzzle's challenge and intensity.*

- **Objectives of the Project:**
  The core objectives of the AI-Powered Maze Escape project are:

  - *Develop a dynamic maze-based game where a human player and an AI opponent compete to reach the exit first.*

  - *Design and implement an AI system using the A* algorithm for optimal pathfinding, influenced by a heuristic intelligence scale (0–1) to control AI difficulty as players progress through levels.*

  - *Integrate search algorithms (BFS, DFS, A*, and Greedy Best-First Search) to manage pathfinding, dead-end detection, and AI behavior under different scenarios.*

  - *Simulate increasing challenge across levels by making the AI faster and more intelligent, encouraging strategic thinking from players.*

  - *Incorporate real-time dynamic changes to the maze, such as obstacle placement and modification, based on the player's movement.*

  - *Implement advanced gameplay features like power-ups, sabotage actions, traps, and checkpoints to add variety and decision-making complexity.*

  - *Test the AI behavior against human players to evaluate its effectiveness, adaptability, and competitiveness.*

  - *Create an engaging and replayable game experience that balances fun with the educational application of AI algorithms.*

3. Game Description

- **Original Game Rules:**
  The original version of AI-Powered Maze Escape was inspired by the classic arcade game Pac-Man. It retained several foundational gameplay elements from Pac-Man,

including:

- *The player navigates a maze with pathways and walls.*
- *Movement is allowed in four directions: up, down, left, and right.*
- *The objective is to reach a specific goal or exit point while avoiding dynamic obstacles.*
- *Time-based gameplay is present, encouraging the player to complete the maze within a countdown timer.*
- *The maze contains moving AI-controlled elements that act as hazards, similar to Pac-Man ghosts.*
- *The maze structure and obstacles are fixed per level, with a focus on fast-paced navigation and reaction time.*

- **Innovations and Modifications:**

  To transform the game from a classic arcade experience into a strategic AI-driven puzzle, several significant enhancements and new mechanics were introduced:

  - *AI vs Player Competitive Gameplay: Instead of avoiding ghosts, the player now races against an AI competitor to reach the maze exit. Both the player and the AI navigate simultaneously through the maze.*
  - *Intelligent AI Opponent: The AI uses the A* search algorithm for pathfinding. Its decision-making capability is scaled using an intelligence level (ranging from 0 to 1), which increases with each level, making the AI smarter,faster and more efficient over time.*
  - *Dynamic Maze Adjustments: The maze is no longer static. The AI can add or remove walls during gameplay to increase difficulty or create alternate paths, based on the player's behavior.*
  - *Sabotage Mechanics: Players are given sabotage tools to disrupt the AI. These include:*
    - *Freezing the AI*

    - *Confusing its navigation*
    - *Temporarily blocking its path*
- *Power-ups and Special Power-ups:*
    - *Speed boosts*
    - *Invisibility of obstacles*
    - *Time bonus*
    - *Teleportation instantly to a random point*
    - *Wall phasing*
- *Obstacles / Traps System: The maze now includes traps or obstacles that only affect the player, adding risk to each move and encouraging strategic thinking. E.g Regular Obstacle (teleports you back to starting point),Killer Obstacle (kills you instantly upon contact).*
- *AI-Generated Hints: If the player is stuck or consistently chooses poor paths, the system provides smart hints to guide them toward a valid solution.*
- *Heuristic-Based Difficulty Scaling: As levels progress, the AI's intelligence score increases, making it react faster and more strategically, resulting in an evolving challenge.Except of the AI the walls moving gets more intense and faster as well making it harder for player to decide.*
- *Time Pressure Mechanism: The countdown timer becomes more restrictive in higher levels, putting additional pressure on decision-making and speed.*

These modifications elevate the game beyond its Pac-Man roots, introducing AI planning, search algorithms, competitive strategy, and dynamic environments—making it a unique blend of real-time competition and intelligent puzzle-solving.

4. AI Approach and Methodology

- **AI Techniques Used:**
  The project leverages multiple AI techniques to create an engaging and progressively challenging maze-based puzzle experience:

  - *A* Pathfinding Algorithm:*
    Employed for real-time navigation of both the AI competitor and AI-controlled obstacles. This algorithm uses heuristic-based search (Manhattan Distance) to efficiently find optimal routes while accounting for dynamically placed walls and obstacles.

  - *Breadth-First Search (BFS):*
    Utilized for detecting dead ends, verifying path existence between key nodes, and ensuring overall maze solvability after AI-triggered dynamic modifications. BFS is also used in AI-generated hints provided to the player when they are stuck.

  - *Adaptive AI Behavior:*
    The AI competitor adapts its intelligence level (0.0 to 1.0) and movement speed based on the current level. This ensures increasing difficulty over time. At lower levels, the AI may make mistakes or take suboptimal routes, while at higher levels, it becomes precise and fast.

- **Algorithm and Heuristic Design:**

  Maze Generation and Dynamics

  - *Recursive Backtracking Algorithm:*
    Used to generate an initial perfect maze (one path between any two points).

  - *Strategic Loop Creation:*
    Loops and alternative paths are manually opened to increase gameplay diversity.

  - *Dynamic Modification System:*
    Periodically adds or removes walls based on player position and AI decisions,

ensuring the maze evolves but always remains solvable via BFS checks.

AI Competitor Logic

- *Intelligence Factor (0.0–1.0):*
  Governs the AI's accuracy, pathfinding frequency, and reaction speed. Higher values correlate with more efficient decision-making.

- *Probabilistic Decisions:*
  Occasionally introduces intentional suboptimal moves at lower intelligence levels to simulate human-like imperfection.

- *Path Recalculation:*
  Triggered upon structural maze changes or collisions with traps/sabotage. Ensures real-time adaptability.

Obstacle AI Behavior

- *A* Pursuit Logic:*
  Obstacles can chase the player using their own instance of A search, giving the game an element of pressure and unpredictability.*

- *Obstacle Types:*

  - *Standard obstacles: Interfere passively with the player's path.*

  - *Killer obstacles: Actively chase the player with higher speed and priority.*

- *Sabotage Resistance:*
  Some AI obstacles respond differently to sabotage events (e.g., confusion or temporary freeze), enhancing strategic complexity.

Heuristic Design

- *Manhattan Distance:*
  Chosen as the core heuristic for A due to the grid-based layout of the maze with non-diagonal movement.*

- *Dynamic Obstacle Avoidance:*
  Paths are evaluated with weights adjusted to avoid moving AI obstacles or traps.

- *Path Verification Layer:*
  BFS ensures every AI or player route remains valid and prevents soft-locking.

- **AI Performance Evaluation:**
  AI systems were assessed through a mix of gameplay testing, simulation, and logging, across several performance dimensions:

  - *Pathfinding Efficiency:*
    Measured the average path computation time across small, medium, and large mazes. Optimizations brought average recalculation times below 30ms on standard test hardware.

  - *Adaptivity:*
    Tracked the AI's response time to changes in the maze layout. High-intelligence AI agents adjusted paths within a single frame update (60fps), while lower intelligence agents required multiple cycles.

  - *Challenge Balance:*
    Observed through human-vs-AI playtests. At low levels, players easily outperformed the AI. As levels progressed, the AI began outperforming or closely matching human players, creating a fair but escalating challenge.

  - *Obstacle Effectiveness:*
    Evaluated by logging how frequently obstacles hindered or cornered players. Balancing ensured obstacles created tension without rendering the game unwinnable or frustrating.

5. Game Mechanics and Rules

- **Modified Game Rules:**
  The game is a modern AI-powered adaptation of classic maze chase mechanics (similar in spirit to Pac-Man), enriched with competitive and dynamic gameplay elements. Key

rule modifications and features include:

- *Players begin at a fixed maze entry point and must find their way to a designated exit.*
- *A real-time AI competitor simultaneously attempts to reach the exit using A* pathfinding.*
- *The maze structure dynamically evolves at runtime by adding or removing walls based on AI logic and player movement patterns.*
- *Players encounter roaming obstacles—some passive, others aggressive (killer-type)—that must be avoided.*
- *Power-ups and special power-ups provide temporary boosts such as speed, freeze effects, obstacle immunity, or AI sabotage.*
- *Sabotage tools can delay, confuse, or freeze the AI, giving the player a strategic edge.*
- *The player must avoid traps placed within the maze, which apply penalties or delays.*
- *As players progress through levels, the difficulty increases by scaling AI intelligence, obstacle behavior, and maze complexity.*

- **Turn-based Mechanics:**

  While the game runs in real-time, the internal architecture operates on a structured, timing-based update cycle. This ensures balanced pacing between player actions, AI behavior, and maze dynamics:

  - *Player Movement: Instantly triggered by player input (WASD or arrow keys), with collision checks against walls, traps, and obstacles.*
  - *AI Movement: Regulated by a delay timer that shortens at higher levels, making the AI faster and more competitive.*
  - *Obstacle Movement: Each obstacle moves independently on a timer; behavior differs depending on type (e.g., random, chase).*

- *Maze Modifications: Occur at fixed time intervals and are influenced by player position, path usage, and AI planning.*

- *Power-up Effects: Operate on timed durations, e.g., speed boost for 5 seconds or AI freeze for 3 seconds.*

This event-based simulation ensures that all game systems (player, AI, maze, obstacles) remain synchronized and responsive during gameplay.

- **Winning Conditions:**
  A player is considered victorious when they meet all of the following conditions in a given level:

  - *Reaches the exit before the time limit expires.*

  - *Reaches the exit before the AI competitor completes its own path.*

  - *In levels with checkpoints, the player must visit all required checkpoints before exiting.*

  Completing a level advances the player to a more challenging stage, with an increasingly intelligent AI, more complex maze layouts, faster obstacles, and less margin for error.

- **Losing Conditions:**
  The player loses the level if any of the following conditions are met:

  - *Time expires before reaching the exit.*

  - *AI competitor reach the exit first.*

  - *The player comes into contact with killer obstacles (immediate failure or heavy penalty).*

6. Implementation and Development

- **Development Process:**
  The development of the AI-powered maze game followed a structured, iterative process to ensure smooth integration of game mechanics, AI behavior, and dynamic elements. The key development stages included:

  - *Core Maze Generation: Implemented using recursive backtracking with*

additional routines for loop creation and solvable structure validation.

- *Player Mechanics: Built systems for responsive player movement, wall collision detection, and interactive input handling.*

- *AI Pathfinding: Integrated A* algorithm for optimal path selection and BFS for solvability checks and backup routing.*

- *Obstacle Behavior Systems: Designed multiple obstacle types with distinct AI (random walkers, chasers, killer pursuers).*

- *Power-Up Logic: Developed systems to spawn, collect, and apply power-up effects including speed boost, AI freeze, and obstacle immunity.*

- *Dynamic Maze Features: Added mechanics for rotating walls, shifting sections, and real-time maze modifications based on player behavior.*

- *UI & Visual Feedback: Designed HUD elements, status indicators, and visual cues for events like AI sabotage or obstacle detection.*

- *Advanced Features: Integrated checkpoint tracking, sabotage tools, trap effects, and level progression logic.*

- *Testing & Tuning: Extensively tested gameplay across levels for AI difficulty balance, fairness, and bug-free pathfinding.*

- **Programming Languages and Tools:**
  - Programming Language: *(e.g., Python, C++)*
    - *Python*
  - Libraries: *(e.g., Pygame, NumPy)*
    - *Pygame: For graphics rendering, event handling, and game loop management*
    - *random:For procedural generation of maze structures and events*
    - *heapq:Priority queue support for the A* pathfinding algorithm*
    - *collections.deque: Efficient BFS queue management*
    - *math: For distance metrics and heuristic evaluations*

- Tools: *(e.g., GitHub for version control)*
    - *GitHub: Version control and collaboration*
    - *VS Code: Primary code editor with Pygame extensions*
    - *Google Drive: Documentation and asset management*

- **Challenges Encountered:**

1. Ensuring Maze Solvability

- *Challenge: The maze dynamically changes during gameplay, which could unintentionally block all paths to the exit.*
- *Solution: Implemented a BFS-based path verification system. If no path exists, walls are programmatically removed to restore solvability before resuming the game.*

2. AI Competitor Balance

- *Challenge: Making the AI challenging without making it unbeatable or frustrating.*
- *Solution: Introduced a scaling intelligence factor that increases with level progression. Added probabilistic behavior to allow occasional suboptimal moves for human-like unpredictability.*

3. Performance Optimization

- *Challenge: Real-time pathfinding across dynamic mazes caused CPU load and frame rate drops.*
- *Solution: Optimized A* logic with reuse of previously computed paths, throttled updates, and event-triggered recalculations instead of constant updates.*

4. Coordinating Dynamic Elements

- *Challenge: Simultaneously handling rotating walls, shifting sections, moving obstacles, and timed power-ups without desyncs or logical conflicts.*
- *Solution: Developed a centralized event scheduler and priority-based update*

system to ensure consistent behavior and interaction across components.

7. Team Contributions

- **Team Members and Responsibilities:**

  ○ **Uzain Ahmed**: Served as the lead developer for the project. Responsible for the core game logic, including the **implementation of AI algorithms (A* and BFS)**, the dynamic maze generation system, and the obstacle behavior models. Also handled integration of game mechanics, power-up systems, and overall game balancing and testing.

  ○ **Aadil**: Focused on the design and implementation of the user interface using Pygame,worked on the main menu system, and UI elements such as timers and score indicators. He also contributed to playtesting and game tuning based on feedback.

  ○ **Bilal**: Handled maze and level design, including the creation of progressively challenging level layouts, power-up placement strategies, and rotating/shifting wall patterns..

8. Results and Discussion

AI Performance:

The AI components in the game demonstrated high efficiency and adaptability across different levels of difficulty:

- *Pathfinding Accuracy & Speed:*
  The A algorithm successfully computed optimal or near-optimal paths with average decision-making times ranging between 2–5 milliseconds, even in highly complex, dynamically shifting mazes.*

- *AI Competitor Win Rate:*
  The adaptive AI competitor exhibited a progressively increasing win rate, starting at around 40% on beginner levels and climbing to 75% on higher levels when tested against novice human players. This balance ensured that the AI remained competitive but not overpowering.

- *Obstacle Efficiency:*

Dynamic obstacles using A pursuit behavior created constant pressure without making the game unfair. Players noted a noticeable increase in difficulty once killer obstacles and aggressive patterns were introduced in later levels.*

- *Adaptation to Maze Changes:*
  AI decision-making remained stable even when faced with frequent maze reconfigurations. Real-time path recalculations allowed both the AI competitor and obstacles to dynamically adjust their behavior in response to the evolving environment.

- *Player-AI Interaction:*
  During playtesting, users found the AI competitor's behavior to be natural yet challenging, especially due to the probabilistic decision-making that occasionally mimicked human-like mistakes.

Game Balance and Fairness:

- *The combination of power-ups, dynamic maze transformations, and intelligence scaling provided a consistently engaging experience.*

- *Internal testing showed that while the AI often won in later stages, skilled players could still win with careful planning, particularly by leveraging power-ups and exploiting AI prediction patterns.*

9. References

- *Pygame Community Documentation. (n.d.). Pygame Documentation. Retrieved from https://www.pygame.org/docs/*
  – Used extensively for game development, including sprite management, event handling, and rendering.

- *Amit Patel. (n.d.). Maze Generation Algorithms. Retrieved from https://www.redblobgames.com/articles/maze-generation/*
  – Referenced for implementing recursive backtracking and dynamic maze updates.

- *GeeksforGeeks. (n.d.). Breadth First Search or BFS for a Graph. Retrieved from https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/*
  – Used as a reference for BFS implementation and understanding graph traversal.

- *Stack Overflow. (Various Threads). Retrieved from https://stackoverflow.com/*
  – Utilized to resolve technical issues during development, such as pathfinding edge cases and performance optimizations.

- *Python Documentation. (n.d.). Python Standard Library. Retrieved from https://docs.python.org/3/library/*
  – Referenced for using libraries such as heapq ,math, random and collections.