# Project Report

## Project Title:

**Pac Ludo Fusion**


## Submitted By:

**Kainat Faisal (22k-4405)**
**Laiba Khan (22k-4610)**


## Course:

Artificial Intelligence (AI)

## Instructor:

**Miss Ravia Ejaz**
**Miss Alina Arshad**

## Submission Date:

**14/05/2025**

---

# 1. Executive Summary


**Project Overview:**
*Pac Ludo Fusion* is an innovative reimagination of the classic Pac-Man game. This project integrates AI-driven enemies with a rotating hexagonal maze to enhance gameplay complexity. The main goal was to implement three AI strategies—Minimax, A* Pathfinding, and Q-Learning—to simulate intelligent ghost behavior. Players must collect pellets and survive as the maze dynamically shifts every 15 seconds, requiring constant adaptation and strategic movement.


---

# 2. Introduction

### Background:

Pac-Man is a classic arcade game centered around navigating a maze, collecting pellets, and avoiding ghosts. This project introduces several enhancements to the original mechanics: a hexagonal grid for greater path diversity and dynamic maze rotation for added challenge. The goal was to explore AI behaviors in a non-trivial, constantly evolving environment.

### Objectives of the Project:

- Develop a playable maze game with AI-driven enemy behaviors.
- Implement three distinct ghost AIs using Minimax, A* Pathfinding, and Q-Learning.
- Evaluate and compare the effectiveness of these AI techniques.
- Introduce rotating maze mechanics to encourage player adaptability and strategic planning.

---

# 3. Game Description

### Original Game Rules:

In traditional Pac-Man, the player navigates a fixed maze, eats pellets to score points, and avoids ghosts. Consuming special pellets can temporarily turn the tables, allowing the player to defeat ghosts.

### Innovations and Modifications:

- **Hexagonal Maze Layout** – Replaces the conventional square grid with a hex grid for more complex navigation.
- **Maze Rotation** – Rotates every 15 seconds, changing viable paths and strategies.
- **Smart Ghosts** –

- **Minimax Ghost**: Uses predictive modeling to intercept the player.
- *A Ghost**: Calculates the shortest path to the player.
- **Q-Learning Ghost**: Learns from experience using reinforcement learning.

- **Power-Ups** –
  - Temporary invincibility upon spawn and life loss.
  - Bonus life after collecting every 50 pellets.

---

# 4. AI Approach and Methodology

## AI Techniques Used:

- **Minimax Algorithm** – Simulates player and ghost moves to anticipate the most advantageous path.
- *A Pathfinding** – Computes shortest paths using a heuristic suited to hex grids.
- **Q-Learning (Reinforcement Learning)** – Employs a neural network (via TensorFlow) that learns optimal behavior over time.

## Algorithm and Heuristic Design:

- **Minimax** uses depth-limited search with state evaluation based on proximity and threat.
- **A*** leverages a hex-based distance function for accuracy in dynamic pathfinding.
- **Q-Learning** utilizes a reward matrix and state-action table, training through gameplay feedback.

## AI Performance Evaluation:

- **Minimax Ghost** showed strong anticipatory behavior, effectively predicting player moves.
- *A Ghost** offered fast, efficient pursuit with low computation time.
- **Q-Learning Ghost** demonstrated significant improvement with training, adapting well to changes over time.
  Each ghost's behavior was evaluated based on capture success rate, path efficiency, and response time to the rotating maze.

---

# 5. Game Mechanics and Rules

## Modified Game Rules:

- Players begin with **5 lives**.
- The **maze rotates** every **15 seconds**.
- Players collect **pellets for points** and can earn extra lives.

## Turn-Based Mechanics:

- Gameplay is real-time, controlled using **W, A, S, D** keys.
- Players can restart the game after victory or defeat by pressing **R**.

## Winning Conditions:

- **Victory**: Collect all pellets.
- **Game Over**: Lose all 5 lives.

---

# 6. Implementation and Development

## Development Process:

The game was developed in Python using the **Pygame** library. Core gameplay, UI, and AI logic were structured into object-oriented modules. Each ghost type was implemented in a modular AI engine that interacts seamlessly with the game environment.

## Programming Languages and Tools:

- **Language:** Python
- **Libraries:** Pygame, NumPy, TensorFlow
- **Tools:** GitHub (version control), PyCharm (IDE)

## Challenges Encountered:

- Implementing rotation for a hex grid without disrupting pathfinding logic.
- Optimizing performance with three concurrent AI systems.
- Training the Q-Learning agent effectively in a real-time, dynamic maze.

---

# 7. Team Contributions

- **Kainat Faisal**
  - Designed the hexagonal maze system.
  - Implemented the A* pathfinding ghost.
  - Co-developed the Q-learning agent and game mechanics.
- **Laiba Khan**

- Designed and implemented the Minimax-based ghost logic.
- Worked on decision heuristics and AI evaluations.
- Co-developed the Q-learning agent and game mechanics.

---

# 8. Results and Discussion

### AI Performance Overview:

- **Minimax Ghost**:
  - Success Rate: 85%
  - Strong predictive capabilities.
- *A Ghost**:
  - Decision Time: ~10ms
  - Most efficient pathing algorithm.
- **Q-Learning Ghost**:
  - Initial Success Rate: 30%
  - Post-training Success Rate: 65%
  - Performance improves significantly with training and adaptation.

The rotating maze added dynamic complexity, requiring the AI systems to be robust and flexible. The A* algorithm remained consistently reliable, while Q-Learning adapted best over time.

---

# 9. References

1. Russell, S., & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*.
2. TensorFlow Documentation – https://www.tensorflow.org/
3. Pygame Documentation – https://www.pygame.org/docs/
4. A* Pathfinding Overview – https://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html
5. Deep Q-Learning Guide – https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf