Project Report

Project Title

AI-Enhanced Pac-Man

Submitted By:

Talha Munir (22K-4465)

Irtiza Ahmed (22K-4211)

Ahmed Kabir (22K-4556)

Course:

AI

Instructor:

MR. SHAFIQUE RAHMAN

Submission Date:

10 MAY 2025

1. Executive Summary

Project Overview:

This project implements a scaled version of the classic Pac-Man game with **enhanced AI** for ghost characters. The project focuses on recreating the **original ghost AI** behaviors while adding **responsive game states**, power-up mechanics, and scaled graphics for modern displays. The implementation uses Python and Pygame, featuring four distinct ghost personalities that create challenging gameplay through **state-based decision making** rather than traditional pathfinding algorithms.

2. Introduction

Background:

Pac-Man is a classic arcade game from 1980 where players navigate a maze, eating pellets while avoiding ghosts. The original game is renowned for its sophisticated AI that gives each ghost a unique personality. This project recreates and enhances these AI behaviors while adapting the game for **modern screens** through **dynamic scaling**.

Objectives of the Project:

- Implement the classic Pac-Man ghost AI behaviors **without using conventional pathfinding algorithms**
- Create a scalable game interface that maintains gameplay **integrity** across **different screen sizes**
- Develop **state-based AI** that responds to **power-ups** and **game conditions**
- Demonstrate how **simple AI rules** can create **complex** and **engaging gameplay**

3. Game Description

Original Game Rules:

- Player controls Pac-Man through a maze
- Objective is to eat all pellets while avoiding ghosts
- Power pellets allow Pac-Man to eat ghosts temporarily
- Four ghosts with unique behaviors chase Pac-Man

- Game ends when all lives are lost or all pellets are eaten

Innovations and Modifications:

- **Dynamic scaling** system (SCALE_FACTOR = 0.65) for modern displays
- Enhanced ghost box mechanics with proper entry/exit behaviors
- Visual power-up timer showing remaining time
- Improved **collision detection** for scaled sprites
- **Revival system** for eaten ghosts with timed respawn
- **Dynamic Speed** for both user and ghost.

4. AI Approach and Methodology

AI Techniques Used:

- **State-Based Decision Making**: Ghosts switch between Chase, Scatter, Frightened, and Dead states
- **Personality-Based AI**: Each ghost has unique targeting behaviors
- **Reactive AI**: Dynamic response to power-up activation and game state changes
- **Simple Decision Trees**: Priority-based direction selection at intersections

Algorithm and Heuristic Design:

1. Target System:

```
 def get_targets():

if powerup:

    return scatter_positions

elif ghost.dead:

    return box_center

else:
    return chase_targets
```

2. Ghost Personalities:
    - **Blinky (Red)**: Direct pursuit of Pac-Man
    - **Pinky (Pink)**: Targets ahead of Pac-Man's position
    - **Inky (Blue)**: Complex targeting using Blinky's position
    - **Clyde (Orange)**: Switches between chase and scatter based on distance

AI Performance Evaluation:

- Ghost effectiveness measured by **player capture rate**
- Response time to **state changes** (power-up activation)
- Behavioral **consistency** across different game states
- Emergent behavior complexity from simple rules

5. Game Mechanics and Rules

Modified Game Rules:

- Scaled game board maintains original 30x32 grid structure
- Power-up duration: 10 seconds (600 frames at 60 FPS)
- Ghost revival times: Blinky (2s), Inky (3s), Pinky (4s), Clyde (5s)
- Screen wrapping for both Pac-Man and ghosts
- Dynamic scoring based on ghost eating sequence

Turn-based Mechanics:

- Real-time gameplay at 60 FPS
- Grid-based movement with intersection detection
- Priority-based turn selection for ghosts
- Smooth animation with 4-frame Pac-Man sprite cycle

Winning Conditions:

- Victory: All pellets and power pellets consumed
- Defeat: All three lives lost through ghost collision

6. Implementation and Development

Development Process:

1. Core game loop implementation with Pygame
2. Board rendering system with scaled graphics
3. Player control and collision detection
4. Ghost AI implementation with individual behaviors
5. Power-up mechanics and state management
6. UI elements and game state handling

Programming Languages and Tools:

- **Programming Language**: Python 3.x
- **Libraries**: Pygame, Math, Copy
- **Tools**: GitHub for version control
- **Assets**: Custom sprite images for player and ghosts

Challenges Encountered:

1. **Scaling Issues**: Maintaining precise collision detection with fractional scaling
2. **Ghost Box Navigation**: Implementing proper gate detection and exit behavior
3. **Movement Synchronization**: Ensuring smooth movement at different scales
4. **State Management**: Coordinating multiple ghost states and transitions

7. Team Contributions

Team Members and Responsibilities:

1. **[Talha Munir]**: Core game engine, scaling system,UI design and power-up system
2. **[Irtiza Ahmed]**: Ghost AI implementation and behavior patterns
3. **[Ahmed Kabir]**: Collision detection and movement mechanics

8. Results and Discussion

AI Performance:

The implemented AI successfully recreates the classic Pac-Man ghost behaviors:

- Ghosts exhibit distinct personalities through different targeting strategies
- State transitions occur smoothly with power-up activation
- Emergent behavior creates unpredictable and challenging gameplay
- Simple rules produce complex-seeming intelligence without pathfinding

Key achievements:

- Ghost behaviors match original game patterns
- Responsive state changes enhance gameplay dynamics
- Scaled graphics maintain game integrity across screen sizes
- Revival system adds strategic depth to power-up usage

9. References

1. Pittman, J. (2009). "The Pac-Man Dossier" - Detailed analysis of original ghost AI
2. Birch, S. (2010). "Understanding Pac-Man Ghost Behavior" - Game Developer Magazine
3. Pygame Documentation - https://www.pygame.org/docs/
4. Classic Gaming - "Pac-Man Ghost AI Explained" - retrocomputing.net
5. Game Programming Patterns by Nystrom, R. - State pattern implementation
6. Pac Man Tutorial : How to Make Pac-Man in Python!