Laboratory assignment

Component 2

Authors: Ichim Stefan, Mirt Leonard
Group: 246/1

March 25, 2025

1 Problem Definition

The problem we aim to solve is the implementation of a Pac-Man game simulation using a multi-agent system approach. Pac-Man is a classic arcade game where the player controls a character that navigates through a maze, collecting dots while avoiding ghosts. When the player consumes special power pellets, the ghosts become temporarily vulnerable, allowing Pac-Man to eat them for bonus points.

In our multi-agent system simulation, we approach this problem by decomposing the game into various autonomous agents that interact with each other within a shared environment. This decomposition allows for modular development, parallel execution, and realistic modeling of the game mechanics through agent interaction patterns.

Using the PADE (Python Agent DEvelopment) framework, we aim to create a simulation that accurately replicates the classic Pac-Man game while demonstrating key principles of multi-agent systems, including communication, coordination, and autonomous decision-making.

2 High-Level MAS Specification

2.1 Inputs and Outputs to/from the MAS System

The multi-agent system for the Pac-Man simulation accepts the following inputs:

1. Initial maze configuration (wall layout, dot positions, power pellet locations)
2. Game parameters (speed, scoring rules, level progression criteria)
3. Autonomous behavior parameters for the Pac-Man agent

The system produces the following outputs:

1. Current game state (positions of all agents, remaining dots, score)
2. Terminal-based visualization of the maze and agents using ASCII characters
3. Game events (dot collection, ghost encounters, level completion)
4. Performance metrics (score, time elapsed, efficiency of agents)
5. Logs of agent interactions and decision-making processes

2.2 Types of Agents

Based on our initial analysis, the Pac-Man simulation will consist of three primary types of agents:

2.2.1 Pac-Man Agent

A single agent representing the player character, responsible for navigating the maze, collecting dots, and avoiding or consuming ghosts.

2.2.2 Ghost Agents

Four distinct ghost agents (Blinky, Pinky, Inky, and Clyde), each with unique behavior patterns and strategies for pursuing Pac-Man.

2.2.3 Environment Agent

A specialized agent that manages the maze state, including dot availability, wall configurations, and coordinates the game progression.

2.3 Specification of Agents

2.3.1 Pac-Man Agent

Inputs:

1. Current position in the maze
2. Local perception of surrounding maze elements (walls, dots, power pellets)
3. Ghost positions within perception radius
4. Game state information from the blackboard (ghost modes, score)

Outputs:

1. Movement decisions (up, down, left, right)
2. Updates to the blackboard regarding position and dot collection
3. Interaction events (power pellet consumption, ghost encounters)
4. Character representation for terminal visualization ('P' for Pac-Man)

Task: The Pac-Man agent must navigate the maze efficiently to collect all dots while avoiding ghosts in their normal state and pursuing them when in a frightened state. It employs pathfinding algorithms and strategic decision-making based on its perception of the environment.

2.3.2 Ghost Agents

Inputs:

1. Current position in the maze
2. Current behavior mode (chase, scatter, frightened)
3. Pac-Man's position (either directly perceived or from blackboard)

4. Maze structure within perception radius

Outputs:

1. Movement decisions (up, down, left, right)
2. Updates to the blackboard regarding position and mode changes
3. Interaction events (collisions with Pac-Man)
4. Character representations for terminal visualization ('B', 'P', 'I', 'C' for respective ghosts)

Task: Each ghost agent must navigate the maze according to its specific behavior pattern. In chase mode, ghosts pursue Pac-Man using various strategies unique to each ghost. In scatter mode, they patrol specific corners of the maze. In frightened mode, they attempt to flee from Pac-Man. The ghost agents implement decision trees and simple pathfinding to achieve these behaviors.

2.3.3 Environment Agent

Inputs:

1. Agent positions and actions
2. Game clock signals
3. Level configuration data

Outputs:

1. Updated maze state (dot availability, power pellet status)
2. Game event notifications (level completion, game over)
3. Mode change signals for ghosts (timing for chase/scatter transitions)
4. Blackboard updates with comprehensive game state
5. Terminal-based visualization of the entire game state using ASCII characters
6. Character representations for maze elements (walls, dots, power pellets)

Task: The environment agent serves as the central coordinator for the simulation. It maintains the maze state, processes agent interactions, manages the game clock, and ensures that all game rules are properly enforced. It also handles the transitions between different game phases and levels.

2.4 Communications Among Agents

The communication structure in our Pac-Man MAS implementation follows the blackboard pattern, providing a centralized knowledge repository accessible to all agents. This approach offers several advantages for our simulation:

2.4.1 Blackboard Structure

The blackboard contains shared knowledge including:

1. Complete maze state (walls, dots, power pellets)
2. Agent positions and current modes
3. Game events (collisions, dot collections, mode changes)
4. Global game state (score, level, time remaining)

2.4.2 Communication Patterns

The environment agent acts as the primary manager of the blackboard, updating it with the latest game state after each game cycle. Ghost and Pac-Man agents both read from and write to the blackboard:

1. Reading: Agents retrieve information outside their perception radius
2. Writing: Agents update their positions and report interactions

2.4.3 Message Types

While the blackboard handles most communication needs, direct messages are exchanged in specific scenarios:

1. Position updates from agents to the environment
2. Mode change notifications from environment to ghosts
3. Collision detection and resolution messages
4. Game state transitions (level completion, game over)

3 Agents' Role Within the MAS

Each agent in our Pac-Man simulation fulfills a specific role that contributes to the overall functioning of the multi-agent system:

3.1 Pac-Man Agent Role

The Pac-Man agent serves as the protagonist within the system, embodying the following roles:

Explorer: Systematically navigates the maze to discover and collect dots and power pellets.

Decision-maker: Evaluates risk and reward to determine optimal paths through the maze, balancing the goals of dot collection and ghost avoidance.

Reactor: Responds dynamically to changing game conditions, such as ghost proximity and power pellet effects.

The Pac-Man agent's actions drive the primary gameplay loop and serve as the focal point around which other agents organize their behaviors.

3.2 Ghost Agent Roles

The four ghost agents function as antagonists within the system, each with a specialized role:

Blinky (Red): Acts as the direct pursuer, implementing aggressive pathfinding to chase Pac-Man by the shortest possible route. Functions as the primary threat and pace-setter for the game difficulty.

Pinky (Pink): Serves as the ambusher, attempting to position itself in front of Pac-Man's expected path. Introduces tactical complexity to the pursuit mechanics.

Inky (Cyan): Operates as the flanker, using both Blinky's position and Pac-Man's position to triangulate interception points. Adds unpredictability and coordination to ghost behavior.

Clyde (Orange): Functions as the patroller, alternating between pursuit when distant from Pac-Man and retreat when close. Provides relief from constant pressure and introduces rhythm to the pursuit dynamics.

Collectively, the ghost agents create emergent complexity through their different pursuit strategies, forcing the Pac-Man agent to adapt to multiple simultaneous threats.

3.3 Environment Agent Role

The environment agent serves multiple critical roles within the system:

Coordinator: Synchronizes agent actions according to the game clock, ensuring fair and consistent gameplay progression.

Referee: Enforces game rules, detects collisions, and determines outcomes of agent interactions.

State Manager: Maintains the authoritative representation of the game state, including dot availability, power pellet status, and score.

Director: Controls the pacing of the game by managing mode transitions for ghosts and level progressions.

Visualizer: Generates the terminal-based ASCII representation of the current game state after each simulation step, displaying the maze, agents, dots, and power pellets using appropriate character symbols.

The environment agent provides the structural foundation upon which the other agents operate, maintaining consistency and facilitating ordered interaction between the autonomous agents.

Through the complementary roles of these agents, the multi-agent system creates a cohesive simulation of the classic Pac-Man game, demonstrating effective agent specialization, communication, and coordinated behavior.