Laboratory assignment

Component 3: Conceptual Analysis and Design

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

April 9, 2025

1 Conceptual Modeling Using Agents (PAGES)

1.1 Pac-Man System Overview

The Pac-Man game simulation is conceptually modeled as a multi-agent system using the PAGES framework (Perception, Action, Goal, Environment, State). This approach provides a systematic way to analyze the agents and their interactions within the system.

1.2 PAGES Model Components

Table 1: PAGES Model for Pac-Man Agent

Perception (P)	• Local maze visibility (walls, dots, power pellets) • Ghost positions and states within perception radius • Score changes and remaining lives
Actions (A)	• Movement in four directions (UP, DOWN, LEFT, RIGHT) • Collection of dots and power pellets • Consumption of vulnerable ghosts
Goals (G)	• Maximize score by collecting all dots and power pellets • Avoid ghosts in normal state • Consume ghosts when vulnerable
Environment (E)	Grid-based maze with walls, dots, power pellets, and tunnels
State (S)	Position, direction, and power status (normal or powered-up)

Table 2: PAGES Model for Ghost Agents

Perception (P)	• Local maze visibility • Pac-Man's position (when within perception radius) • Current behavior mode (chase, scatter, frightened)
Actions (A)	• Movement in four directions • Return to ghost house when consumed • Respawn from ghost house
Goals (G)	• Blinky: Direct pursuit of Pac-Man • Pinky: Intercept Pac-Man ahead of his path • Inky: Flank Pac-Man through coordination • Clyde: Alternate between pursuit and patrol
Environment (E)	Same grid-based maze as Pac-Man with ghost house region
State (S)	Position, direction, and mode (normal, frightened, returning)

Table 3: PAGES Model for Environment Agent

Perception (P)	• Complete maze state • All agent positions and states • Game timer and score information
Actions (A)	• Update maze state • Signal mode changes to ghosts • Process collisions between agents • Progress game level and visualization
Goals (G)	• Maintain game consistency • Enforce rules and fair progression • Provide visualization and scoring
Environment (E)	The entire game system it manages and coordinates
State (S)	Complete game state including all agents, maze elements, and timers

2 Properties of the Environment

The Pac-Man environment can be characterized according to standard properties of multi-agent environments. The table below summarizes these key properties:

Table 4: Properties of the Pac-Man Environment

Property	**Classification**	**Key Characteristics**
Accessibility	Partially Accessible	• Environment Agent: complete accessibility • Pac-Man/Ghosts: limited by perception radius
Determinism	Deterministic	• Predictable outcomes for actions • Well-defined transition function
Episodic vs Sequential	Sequential	• Current decisions affect future states • Power pellet timing creates dependencies
Static vs Dynamic	Dynamic	• Game state evolves independently • Timer-based mode changes occur
Discrete vs Continuous	Discrete	• Grid-based positions • Discrete time steps and actions
Agent Structure	Multi-agent	• Multiple concurrent agents • Mix of cooperation and competition
State Dependency	Primarily Markovian	• Next state depends on current state and actions • Limited non-Markovian elements in ghost behavior

2.1 Class Diagram

The Pac-Man multi-agent system is built around several core object classes that define the structure and behavior of the simulation. The following class diagrams illustrate the main components of the system.

2.1.1 Agent Classes

The agent hierarchy consists of a base Agent class extended by PacManAgent and GhostAgent classes. The GhostAgent is further specialized into four distinct ghost types, each with unique targeting and behavior patterns.

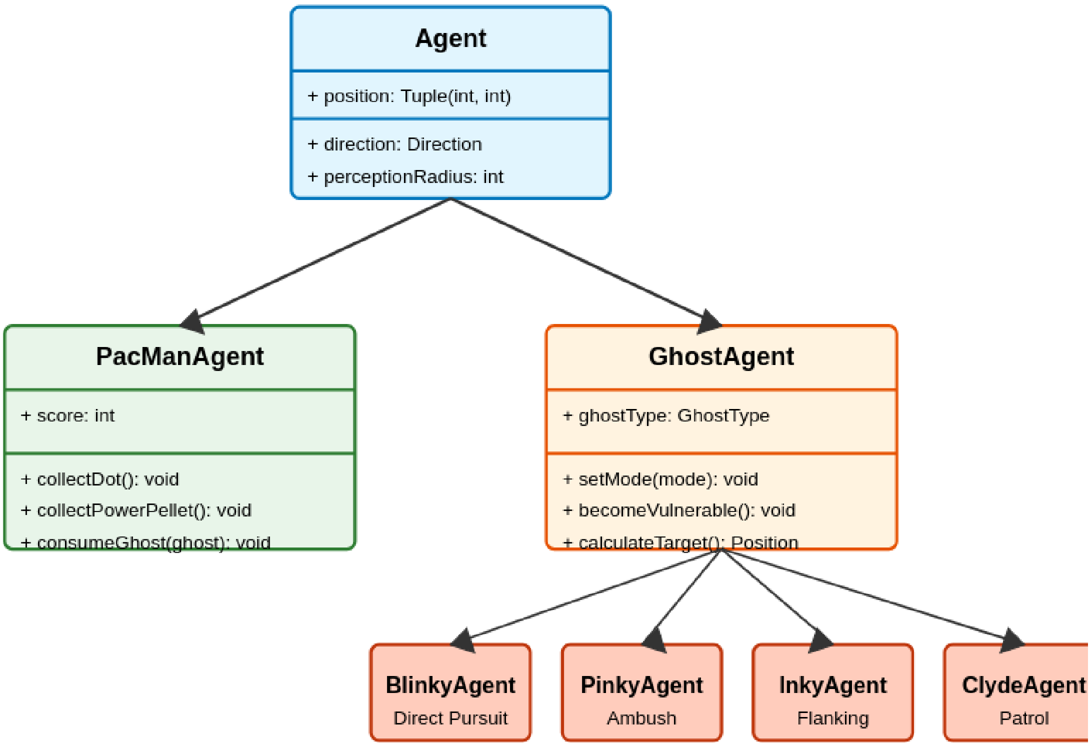

Figure 1: Pac-Man and Ghost Agent Class Hierarchy

Key features of the agent classes:

- **Agent**: Abstract base class defining common properties such as position, direction, and perception radius
- **PacManAgent**: Implements dot collection, power pellet effects, and ghost consumption
- **GhostAgent**: Defines common ghost behaviors including mode changes and vulnerability states
- **Specialized Ghosts**: Each ghost type implements unique targeting strategies:
  - Blinky: Direct pursuit targeting Pac-Man's current position
  - Pinky: Ambush tactics targeting positions ahead of Pac-Man
  - Inky: Flanking behavior using both Blinky and Pac-Man positions
  - Clyde: Alternating between pursuit and patrol behaviors

2.1.2 Environment System

The environment system consists of three main components: the EnvironmentAgent that manages the game state, the Maze that defines the playfield, and the Blackboard that facilitates agent communication.

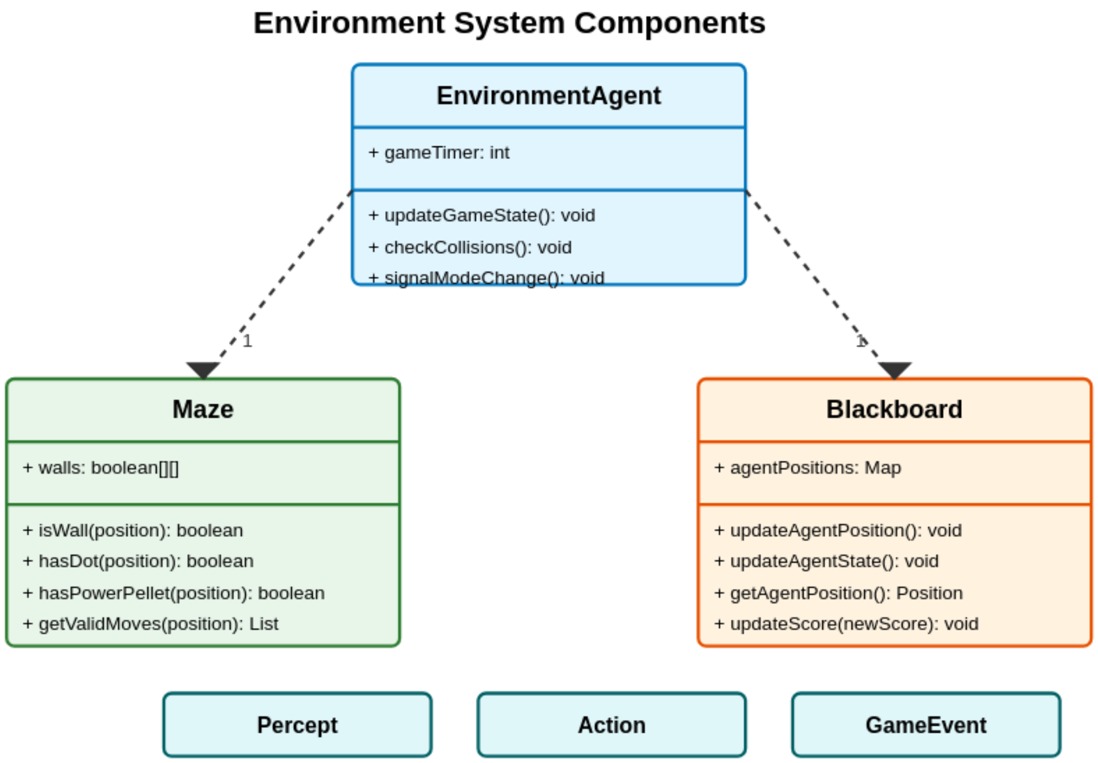
Environment System Components
EnvironmentAgent
+ gameTimer: int
+ updateGameState(): void
+ checkCollisions(): void
+ signalModeChange(): void
1
1
Maze
+ walls: boolean[][]
+ isWall(position): boolean
+ hasDot(position): boolean
+ hasPowerPellet(position): boolean
+ getValidMoves(position): List
Blackboard
+ agentPositions: Map
+ updateAgentPosition(): void
+ updateAgentState(): void
+ getAgentPosition(): Position
+ updateScore(newScore): void
Percept
Action
GameEvent

Figure 2: Environment System Components

Key components of the environment system:

- **EnvironmentAgent**: Coordinates the game simulation, manages collisions, and controls ghost mode transitions
- **Maze**: Represents the physical game space with walls, dots, power pellets, and provides navigation utilities
- **Blackboard**: Serves as the central knowledge repository, storing agent positions, game state, and facilitating indirect communication
- **Support Classes**: Include Percept, Action, and GameEvent structures that facilitate agent interaction

This modular design enables clean separation of concerns while providing the necessary communication channels between components. The agent-based architecture allows for autonomous decision-making while the environment system maintains game consistency and rule enforcement.

3 Communication and Interaction Model

3.1 Communication Architecture

The Pac-Man multi-agent system employs a hybrid communication architecture that balances efficiency with flexibility.

Table 5: Communication Mechanisms in Pac-Man MAS

Mechanism	**Purpose and Characteristics**
Blackboard Pattern	• Central knowledge repository for shared game state • Asynchronous, non-direct communication • Suitable for non-time-critical information
Direct Messaging	• Point-to-point agent communication • Used for time-critical interactions • Event-driven notifications

3.2 Blackboard Structure and Access

Table 6: Blackboard Access Patterns

Agent	**Read Access**	**Write Access**
Environment Agent	Complete game state	All sections
Pac-Man Agent	Ghost positions and modes	Own position, dot collection events
Ghost Agents	Pac-Man position, other ghosts	Own position and mode

3.3 Message Types

The following message types facilitate direct communication between agents:

- **ModeChangeMessage**: Notifies ghosts of behavior mode transitions
- **CollisionMessage**: Communicates agent interaction outcomes
- **StateTransitionMessage**: Signals significant game state changes

3.4 Core Interaction Protocols

Table 7: Key Interaction Protocols

Protocol	**Key Steps**
Game Cycle	1. Update timers and check transitions 2. Agents perceive environment 3. Agents decide and execute actions (prioritized) 4. Resolve interactions and update state 5. Generate visualization
Collision Resolution	1. Detect position coincidence 2. Check ghost vulnerability status 3. Execute appropriate outcome (score update or life loss) 4. Update agent positions and states
Power Pellet Effect	1. Pac-Man collects power pellet 2. Notify all ghosts of frightened mode 3. Start timer and track duration 4. Restore normal ghost behavior when expired