

IntelliPac: An AI-Driven Pacman Chasing Game



CSE-4110: Artificial Intelligence Laboratory

Submitted to:

Md. Mehrab Hossain Opi
Lecturer
Department of CSE, KUET

Waliul Islam Sumon
Lecturer
Department of CSE, KUET

Submitted by:

Peyal Saha
Roll: 2007001
Department of CSE, KUET

Ahsanul Haque Hasib
Roll: 2007020
Department of CSE, KUET

Contents

List of Figures	ii
1 Introduction	1
2 Key Features	1
3 Extra Features	2
3.1 Loop Detection & Prevention	2
3.2 Emergency Mode (Pacman)	2
3.3 Dynamic Power Pellet Spawning	2
3.4 Evasion Goal Selection	2
4 Game Rules and Mechanics	2
4.1 Map Design	2
4.2 Gameplay Rules	3
5 Methodology	4
5.1 Fuzzy Logic (Decision Making)	4
5.1.1 Pacman's Fuzzy Logic	4
5.1.2 Ghosts' Fuzzy Logic	5
5.2 A* Pathfinding Algorithm	7
5.3 Minimax Algorithm (Ghosts Only)	7
6 Game State & Scoring	7
6.1 State Representation	7
6.2 Scoring System	8
7 Key Parameters (Configurable)	8
8 Example Scenarios	8
8.1 Scenario 1: Normal Chase	8
8.2 Scenario 2: Power Pellet Contest	8
8.3 Scenario 3: Scared Mode	8
8.4 Scenario 4: Cornering and Coordination	8
8.5 Scenario 5: Loop Prevention	9
9 How to Run	9
10 Discussion	9
11 Conclusion	10

List of Figures

1	System Flow Diagram	3
2	Fuzzy Inference System	4
3	Membership Functions for Pacman	5
4	Membership Functions for Ghosts	6

1 Introduction

Artificial Intelligence (AI) has become an essential component in developing intelligent agents capable of reasoning, planning, and interacting autonomously within dynamic environments. The *Pacman* domain provides a classical benchmark for evaluating such systems due to its balanced combination of search, planning and adversarial behaviors within a discrete, partially deterministic environment. In this study, we design and implement a fully automated **AI-versus-AI Pacman framework** where both Pacman and the Ghosts are controlled by intelligent agents operating in real time.

The primary objective of this work is to integrate and demonstrate the effectiveness of three complementary AI paradigms—**Fuzzy Logic**, **A* Pathfinding**, and **Minimax Search with α - β pruning**—in a single hybrid decision-making system. Each technique contributes a distinct layer of reasoning: Fuzzy Logic governs high-level decision arbitration under uncertainty; A* provides optimal or near-optimal spatial navigation; and Minimax enables adversarial reasoning for the Ghost agents. The synergy of these components results in a robust, interpretable, and adaptable AI architecture capable of managing both strategic and tactical challenges inherent in adversarial game play.

This work provides an experimental platform for exploring:

- The efficacy of *fuzzy rule-based systems* in high-level mode selection within competitive environments.
- The interaction between heuristic search (A*) and reactive control policies under time constraints.
- The role of depth-limited adversarial search (Minimax) in multi-agent coordination and pursuit-evasion scenarios.

2 Key Features

The proposed system integrates several artificial intelligence techniques into a unified Pacman simulation framework. Each feature contributes to the system’s adaptability, efficiency, and strategic depth.

1. **Pacman AI Controller:** A fuzzy logic-based decision module determines Pacman’s active goal among *Eat Food*, *Eat Power* and *Evade*. The controller evaluates continuous inputs such as ghost proximity, power-pellet distance and remaining food ratio to compute goal priorities. This enables smooth and context-aware switching between offensive and defensive behavior.
2. **Ghost AI Controller:** Each ghost combines fuzzy behavior selection with depth-limited Minimax search and α - β pruning. Fuzzy inference assigns weights to five behavioral modes—*Aggressive*, *Coordinated*, *Intercept*, *Scatter*, and *Defensive*—based on relative positions, inter-ghost distance and Pacman’s state. The Minimax module then evaluates tactical actions that anticipate Pacman’s counter-moves.
3. **Pathfinding Module:** Both Pacman and ghosts use an A* pathfinding algorithm for spatial navigation. A Greedy Best-First Search (GBFS) fallback ensures rapid re-planning when time or path complexity constraints arise. The heuristic integrates Manhattan distance with a recent-path penalty to reduce loop formation.
4. **Loop Detection and Prevention:** Recent agent trajectories are stored to identify cyclic movement patterns. A loop penalty discourages path repetition, while Pacman can trigger an emergency re-plan to exit confined regions or repeated paths.
5. **Emergency and Evasion Mechanism:** When a ghost approaches within a critical distance or no safe path exists, Pacman enters an emergency mode. It prioritizes tiles offering maximum escape routes and distance from threats, re-planning every frame to ensure survival.

6. **Dynamic Power Pellet Regulation:** Power pellets can respawn under specific conditions (e.g., low pellet ratio or extended play without power-ups). The spawn logic ensures fair placement at locations sufficiently distant from ghosts, maintaining balanced difficulty and strategic variety.
7. **Visualization and Interface:** The system features a full-screen, animated graphical interface using Pygame. Real-time overlays display agent paths, scores, lives, and current AI modes. This design supports interactive analysis of decision-making behavior for instructional or research purposes.

These features collectively enable adaptive, interpretable, and real-time AI interactions between Pacman and the Ghosts, demonstrating an effective integration of fuzzy reasoning, heuristic search, and adversarial planning.

3 Extra Features

3.1 Loop Detection & Prevention

Tracks position history, applies penalties for revisits, triggers emergency mode on repeated cycles, and treats recent positions as temporary obstacles.

3.2 Emergency Mode (Pacman)

Triggered by loops, being stuck, or nearby ghosts. Forces EVADE, replans every frame, and avoids recent tiles.

3.3 Dynamic Power Pellet Spawning

Enabled mid-game (10%–85% pellets eaten; ≥ 6 pellets remain) with cooldown and safety checks: spawns on safe tiles (≥ 3 tiles from ghosts), sometimes replacing a pellet.

3.4 Evasion Goal Selection

Scores candidate tiles by distance from ghosts, loop penalty, and number of escape routes; picks top 30 and A* paths to any.

4 Game Rules and Mechanics

4.1 Map Design

1. **Maze Structure:** The game is played on a 19×19 grid that is procedurally generated at the start of each run. The generation algorithm ensures a connected, navigable maze with multiple escape paths and branching corridors to encourage strategic movement rather than purely reactive play.
2. **Tile Types:** Each tile can represent one of several entities: '#' denotes a wall or obstacle, '.' indicates a regular pellet, 'o' denotes a power pellet, 'P' marks Pacman's position, and 'R' or 'B' represent the two Ghosts. These tiles form the fundamental state representation used by the AI controllers for decision making.
3. **Entity Placement:** Pacman is initialized at a central or high-degree node to maximize mobility, while Ghosts spawn in opposite corners to maintain early-game balance. Power pellets are distributed at maze corners and at selected safe tiles to promote exploration.

4.2 Gameplay Rules

1. **Movement:** All agents operate on a four-connected grid, moving one tile per frame along the cardinal directions (up, down, left, right). Movement is restricted to non-wall tiles.
2. **Pellet Collection:** Pacman gains +50 points upon consuming a regular pellet. Each consumed pellet is removed from the grid and updates the global score and remaining food ratio used by the fuzzy logic controller.
3. **Power Pellets:** Consuming a power pellet triggers “scared mode,” in which all Ghosts become vulnerable for 240 frames (approximately 16 seconds at 15 FPS). During this state, Pacman can capture Ghosts for +750 points each, with an additional +1500 bonus for multi-ghost captures.
4. **Lives and Respawn:** Pacman starts with four lives. Upon contact with a non-scared Ghost, one life is deducted, and an invulnerability window of 90 frames (6 seconds) prevents immediate repeat deaths. Agents are respawned to safe starting positions after a life loss.
5. **Collision Handling:** Collisions are resolved by evaluating the Ghosts’ current state. If a Ghost is in normal mode, Pacman loses a life; if it is scared, the Ghost is temporarily removed from play and respawns after a fixed delay.
6. **Victory and Termination Conditions:** The game concludes when one of the following conditions is met: (i) Pacman consumes all pellets on the map, (ii) Pacman reaches a cumulative score of 30,000 points, (iii) all lives are lost, or (iv) the simulation exceeds 3,000 steps (hard time limit). These criteria ensure both bounded gameplay and fair comparison across experimental runs.

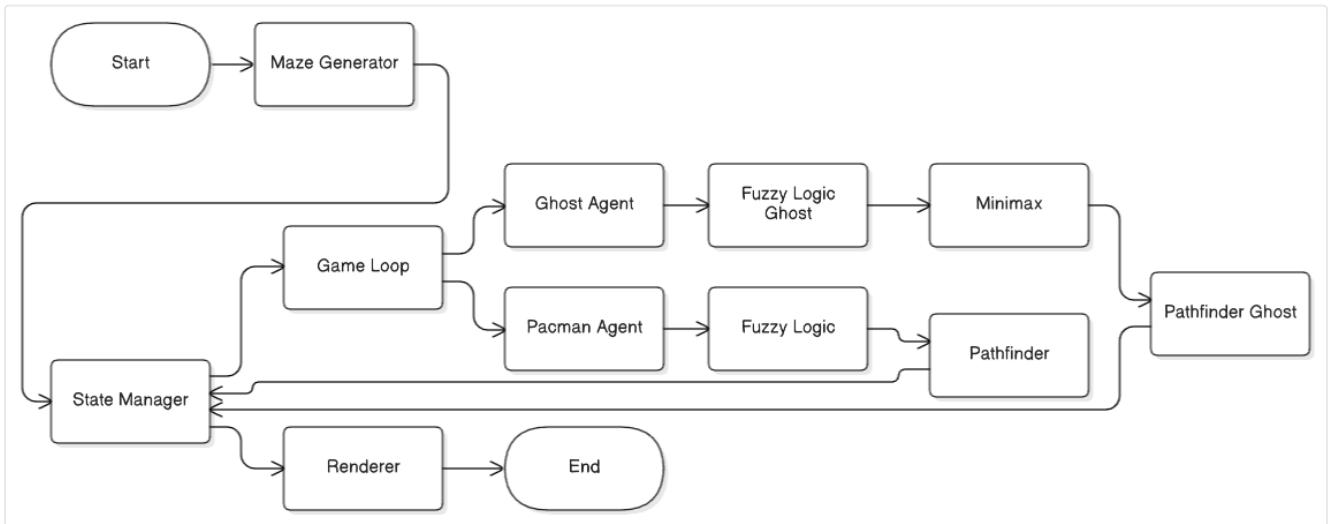


Figure 1: System Flow Diagram

5 Methodology

5.1 Fuzzy Logic (Decision Making)

Fuzzy logic provides a mechanism for reasoning with uncertainty by mapping continuous, imprecise variables to linguistic categories. Each input variable is represented through overlapping membership functions that produce graded truth values within $[0, 1]$. Decision outputs are derived using rule-based inference and defuzzification, allowing the agent to select the most appropriate behavioral mode given the current game context.

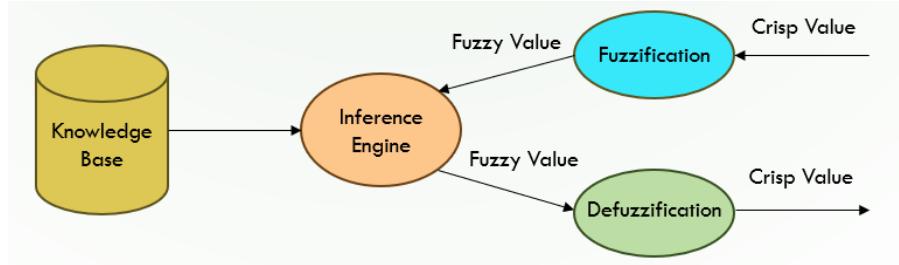


Figure 2: Fuzzy Inference System

5.1.1 Pacman's Fuzzy Logic

The Pacman agent uses three continuous input variables:

- d_g : Distance to the nearest ghost.
- d_p : Distance to the nearest power pellet.
- r_f : Ratio of remaining pellets to the initial total.

The output variable represents the *Behavioral Mode*—one of EAT_FOOD, EAT_POWER, or EVADE.

Membership Functions

- **Ghost Distance (d_g):** *Very Close* (0–3), *Close* (2–6), *Medium* (5–10), *Far* (8–15), *Very Far* (>15).
- **Power Pellet Distance (d_p):** *Near* (0–3), *Moderate* (3–7), *Far* (>7).
- **Food Ratio (r_f):** *Low* (0–0.2), *Medium* (0.2–0.6), *High* (>0.6).

Fuzzy Rule Base for Pacman The following rules determine Pacman's current mode:

1. IF (d_g is *Very Close*) AND (d_p is *Far*) THEN EVADE.
2. IF (d_g is *Close*) AND (d_p is *Near*) THEN EAT_POWER.
3. IF (d_g is *Far*) AND (r_f is *High*) THEN EAT_FOOD.
4. IF (d_g is *Medium*) AND (r_f is *Low*) THEN EAT_POWER.
5. IF (d_g is *Very Far*) AND (r_f is *Low*) THEN EAT_FOOD.
6. IF (d_g is *Close*) AND (r_f is *Low*) THEN EAT_POWER.
7. IF (d_g is *Medium*) AND (d_p is *Near*) THEN EAT_POWER.
8. IF (d_g is *Close*) AND (r_f is *High*) THEN EVADE.
9. IF (d_g is *Very Far*) AND (d_p is *Near*) THEN EAT_POWER.

10. IF (d_g is *Very Close*) AND (r_f is *Medium*) THEN EVADE.

After rule aggregation and normalization, the mode with the highest activation value is selected.

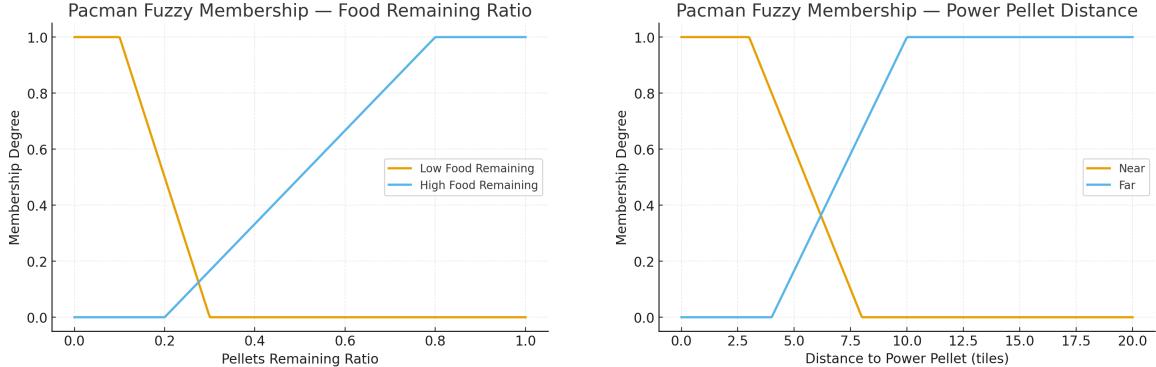


Figure 3: Membership Functions for Pacman

5.1.2 Ghosts' Fuzzy Logic

Each Ghost uses the following input variables:

- d_p : Distance to Pacman.
- d_g : Distance to the other ghost.
- s : Pacman's state (*Normal* or *Powered*).
- r_e : Pacman's number of available escape routes.
- r_f : Remaining pellet ratio.

The output variable is the Ghost's *Behavioral Mode*, chosen among AGGRESSIVE, COORDINATED, INTERCEPT, SCATTER, and DEFENSIVE.

Membership Functions

- **Pacman Distance (d_p):** *Very Close* (0–3), *Close* (3–6), *Medium* (5–10), *Far* (8–15), *Very Far* (>15).
- **Ghost Distance (d_g):** *Too Close* (0–3), *Optimal* (3–6), *Far Apart* (>6).
- **Escape Routes (r_e):** *Low* (0–2), *Moderate* (2–4), *High* (>4).
- **Pellet Ratio (r_f):** *Low* (0–0.2), *High* (>0.2).

Fuzzy Rule Base for Ghosts The Ghost controller uses a set of interpretable fuzzy rules:

1. IF (s is *Normal*) AND (d_p is *Close*) AND (d_g is *Optimal*) THEN AGGRESSIVE.
2. IF (s is *Normal*) AND (d_p is *Medium*) AND (d_g is *Optimal*) THEN COORDINATED.
3. IF (s is *Normal*) AND (d_p is *Far*) AND (r_e is *High*) THEN INTERCEPT.
4. IF (s is *Normal*) AND (d_p is *Very Close*) AND (d_g is *Too Close*) THEN SCATTER.
5. IF (s is *Powered*) THEN DEFENSIVE.
6. IF (s is *Normal*) AND (r_f is *Low*) AND (d_p is *Medium*) THEN INTERCEPT.

7. IF (s is *Normal*) AND (r_e is *Low*) THEN COORDINATED.
8. IF (s is *Normal*) AND (d_p is *Very Far*) THEN SCATTER.
9. IF (s is *Powered*) AND (d_p is *Close*) THEN DEFENSIVE.
10. IF (s is *Normal*) AND (d_p is *Close*) AND (r_e is *Low*) THEN AGGRESSIVE.

After fuzzy inference, a softmax normalization produces the probability distribution over modes. A short-term “momentum” mechanism prevents rapid oscillation by penalizing mode changes within a fixed temporal window.



Figure 4: Membership Functions for Ghosts

5.2 A* Pathfinding Algorithm

The navigation subsystem employs the A* search algorithm for optimal pathfinding. Each node n maintains a cost function:

$$f(n) = g(n) + h(n),$$

where $g(n)$ represents the cumulative path cost from the start node, and $h(n)$ is the heuristic estimated cost to the goal, computed using Manhattan distance. For small goal sets ($|G| \leq 6$), $h(n)$ blends the nearest and average goal distances to reduce dead-end traversal. A loop-penalty term is applied for nodes revisited in the recent trajectory. When A* fails to find a path due to time constraints, Greedy Best-First Search (GBFS) is used as a fallback strategy.

5.3 Minimax Algorithm (Ghosts Only)

Ghost agents use a depth-limited Minimax algorithm with α - β pruning to plan adversarial moves. The search alternates between:

- **MAX nodes:** Ghosts select moves to maximize capture probability and positional advantage.
- **MIN nodes:** Pacman’s simulated moves minimize threat proximity and maximize survival.

The evaluation function combines:

$$E = w_1(20 - d_p) + w_2 C_{coord} + w_3 B_{intercept} - w_4 P_{loop} - w_5 C_{cluster},$$

where C_{coord} rewards optimal spacing between ghosts, $B_{intercept}$ rewards proximity to predicted Pacman targets, and P_{loop} and $C_{cluster}$ penalize oscillatory or clustered formations. Depth is kept small (typically 1–2 plies) for real-time performance, and minimal random noise is introduced to break deterministic ties.

6 Game State & Scoring

6.1 State Representation

The game state is formally defined as a five-element tuple:

$$S = \langle \text{pac}, \text{ghosts}, \text{pellets}, \text{power}, \text{scared_timer} \rangle.$$

Here, `pac` denotes Pacman’s current position on the grid, represented by Cartesian coordinates (x, y) . The variable `ghosts` stores the positions of all active ghosts as ordered pairs $[(x_1, y_1), (x_2, y_2)]$. The set `pellets` contains the coordinates of all remaining regular food pellets that Pacman must consume, while `power` represents the subset of active power pellets that can temporarily make the ghosts vulnerable. Finally, `scared_timer` is an integer variable (measured in frames) that indicates the remaining duration of the ghosts’ frightened state; when this value equals zero, all ghosts return to their normal behavior.

At each time step, this state vector encapsulates the complete observable environment and is used by both the Pacman and Ghost AI systems for decision-making, evaluation, and rendering.

6.2 Scoring System

Action	Points
Regular pellet	+50
Power pellet	+500
Eating scared ghost	+750 + 1500 bonus
Death penalty	-200
Win threshold	30,000

7 Key Parameters (Configurable)

Parameter	Description
FPS = 15	Target frames per second
MAX_GAME_STEPS = 3000	Maximum game duration
GHOST_MINIMAX_DEPTH = 1	Ghosts' search depth
PACMAN.REPLAN_STEPS = 3	Pacman planning cadence
TILE = 32	Base tile size (auto-adjusted fullscreen)

8 Example Scenarios

This section presents typical interactions between Pacman and the Ghosts, highlighting the adaptive behaviors emerging from the hybrid AI system.

8.1 Scenario 1: Normal Chase

In low-risk conditions, Pacman's fuzzy controller selects the EAT_FOOD mode. Using A* pathfinding, Pacman moves efficiently toward nearby pellets while maintaining distance from the Ghosts. The Ghosts, in AGGRESSIVE_PURSUIT mode, apply minimax-based targeting to close in, producing a balanced pursuit dynamic.

8.2 Scenario 2: Power Pellet Contest

When a Ghost approaches and a power pellet is nearby, Pacman shifts to EAT_POWER. The Ghosts anticipate this by switching to INTERCEPT, attempting to block access. This creates a strategic race where both sides optimize spatial positioning under uncertainty.

8.3 Scenario 3: Scared Mode

After consuming a power pellet, Ghosts become vulnerable for 240 frames. They transition to SCATTER or DEFENSIVE, dispersing to reduce capture risk. Pacman, empowered, aggressively seeks additional points by chasing weakened Ghosts or clearing pellets in safe zones.

8.4 Scenario 4: Cornering and Coordination

When Pacman enters a region with few escape routes, fuzzy inference triggers the EVADE mode, directing movement toward higher-connectivity nodes. Ghosts respond with COORDINATED.HUNT, positioning themselves on opposite flanks to restrict Pacman's mobility—illustrating emergent cooperative behavior.

8.5 Scenario 5: Loop Prevention

If Pacman repeatedly revisits recent tiles, loop detection activates emergency re-planning, enforcing path diversity. Ghosts also adjust their minimax evaluation to penalize repetitive pursuit paths, maintaining variability and preventing stalemates.

9 How to Run

Run:

Listing 1: Running the Game

```
1 python pacman_ai.py
```

Controls:

- ESC — Exit the game
- SPACE — Close game after game-over screen

The game is fully automated—just watch the AI play.

10 Discussion

The implemented hybrid AI framework effectively demonstrates that integrating fuzzy logic, heuristic search, and adversarial planning can produce adaptive and human-like behavior in a real-time simulation. Fuzzy logic provides smooth, non-binary decision transitions, enabling agents to react to uncertain or ambiguous conditions with graded responses rather than discrete triggers.

For Pacman, fuzzy reasoning allows continuous adaptation between offensive and defensive behavior. By evaluating ghost proximity, power-pellet accessibility, and food density, Pacman dynamically balances risk and reward, exhibiting natural transitions between evasion, pursuit, and exploration. This approach eliminates rigid thresholds and results in more context-aware decision-making.

The Ghost agents' hybrid fuzzy–minimax architecture improves coordination and tactical diversity. Fuzzy logic determines strategic intent (for example, aggressive pursuit, interception, or scattering), while the minimax search refines short-term tactics within the chosen mode. This separation of high-level and low-level control reduces repetitive pursuit cycles and enhances cooperative hunting behavior. Momentum-based behavior persistence further prevents erratic switching between modes.

Additional mechanisms such as loop detection, emergency re-planning, and dynamic power-pellet spawning ensure game balance and sustained interaction. At 15 FPS, the system maintains real-time performance with minimal computational overhead, validating the practicality of multi-layered AI for embedded or game-based systems. However, manual tuning of fuzzy parameters and full-state observability remain constraints. Future extensions could integrate adaptive learning for parameter optimization and partial observability models to simulate realistic uncertainty.

11 Conclusion

This work presents a hybrid AI system unifying fuzzy logic, A* pathfinding, and minimax planning within an autonomous Pacman environment. The combination enables intelligent, interpretable, and responsive behavior in both cooperative and adversarial contexts. Pacman’s fuzzy control ensures situational awareness and risk-sensitive movement, while the Ghosts’ fuzzy–minimax blend provides coordinated pursuit strategies. The architecture demonstrates that hybrid reasoning frameworks can outperform single-method agents in adaptability, realism, and stability—achieving near-human decision continuity under real-time constraints.

Future work should focus on learning-based tuning of fuzzy rules, scalable maze generalization, and the incorporation of probabilistic reasoning for partial observability.