SmartPac

Intelligent path-finding agent for Pac-Man using A* informed search algorithm

By:

Aya Osama Majar – sec 2

Basma Walid Elsayed – sec 2

Reem Mohammed Eldafrawy – sec 3

Dr. Sara Elsayed ELmetwally

1. Problem Formulation

1.1 Problem Definition

Classic arcade games like Pac-Man are ideal platforms for developing and testing intelligent agents. In this project, we aim to implement a one of algorithms used in developing intelligent agents; which is: A* informed search algorithm.

The agent's goal is to collect all dots (pellets) while avoiding ghosts and walls efficiently.

1.2 Previous Solutions

We have chosen a game that was implemented using all early AI algorithms, or even was developed with no AI, like random movement, after search we concluded that *A** is generally the **Best Choice** for Pac-Man. It provides the optimal balance of:

- Finding shortest paths to dots (efficiency)
- Avoiding ghosts (safety)
- Computational feasibility (performance)

2. PEAS:

- P - Performance Measure:
    - Performance Measure in Pac-Man could be:
        a. The number of dots collected [The more dots Pac-Man collects, the higher the score].
        b. Level completion the performance measure [Once Pac Man collects all the dots on a level, he successfully completes it].

    c. Avoiding ghosts [if Pac-Man gets caught by a ghost, he loses a life].

- <u>E – Environment:</u>
  - maze that Pac-Man moves through, which includes :
    a. Walls.
    b. Pellets / dots.
    c. Ghosts [the enemies that chase Pac-Man].
    d. Pac-Man.
- <u>A – Actuators:</u>
  - The actuators are the movements of Pac-Man that are executed to navigate through the maze :
  - Pac-Man can move: Up / Down / Left / Right.
- <u>S – Sensors:</u>
  - The sensors are how Pac-Man obtains information about the environment:
    a. Location of dots: Pac-Man needs to know where the dots pellets are located.
    b. Location of ghosts: Pac-Man must keep track of the ghosts to avoid colliding with them.
    c. General maze layout: Pac-Man needs to be aware of where walls and corridors are to guide his movements.

3. ODESA:

- O - Observable:
  The environment is fully observable. The player can see the entire maze, the location of all dots, power pellets, ghosts, and Pac-Man himself. Nothing is hidden from view.

- D - Deterministic:
  Traditional Pac-Man is deterministic in that the same action in the same state always leads to the same next state. However, ghost movement patterns can appear random to novice players (though they follow specific algorithms).

- E - Episodic:
  Pac-Man is episodic. Each level is a separate episode, and actions in one level don't affect other levels (except for increasing difficulty in some versions).

- S - Static:
  The Pac-Man environment is not fully static since ghosts move independently of Pac-Man's actions. This makes it dynamic rather than static.

- A - Agent-based:
  Pac-Man involves a single agent (Pac-Man himself) navigating through the environment.

4. SmartPac - Subsystems

Subsystem	Description
Maze Renderer	Builds and draws a grid-based maze layout with walls and dots.
Pac-Man Agent	Autonomous player using A* search to reach the nearest pellet.
Ghost Agent	Dynamic enemies with randomized movement logic.
Collision Detection	Detects when Pac-Man touches a ghost or eats a dot.
Game Loop	Integrates real-time updates, rendering, and agent movement.

5. SmartPac - Technologies

- **Python**: Core programming language.
- **Pygame**: For game simulation and rendering.
- ***A Search****: Heuristic pathfinding used in the player agent.
- **OOP Design**: Maze, Player, and Ghosts as separate classes.

6. Future Work

Enhance the player algorithm to be more efficient and include a penalty cost when the ghost is near the player which will make the player survive longer and avoid the ghost