AI Project Proposal

Samank Gupta (G41566207)

Classic Snake Game with a random enemy snake

Github Link

Problem Description

We all have played the snake game on a Nokia 3310. In this project, we build an AI agent to play a modified version of the classic snake game. In this version, a small enemy snake appears randomly on the grid for a few seconds. If our snake touches this enemy snake, the game ends. The agent needs to eat food to grow longer and survive as long as possible while avoiding itself, and this enemy.

Uncertainties Involved

The enemy snake appears at a random time and at a random location for a few seconds. The movement of the enemy snake also follows a random pattern. These make the game unpredictable and more challenging for the AI to plan.

Why This Problem is Non-Trivial

The game becomes dynamic and unpredictable due to the random enemy. Basic search algorithms like BFS or A* cannot handle this kind of uncertainty well. The agent must make decisions over time, thinking not just about the current move but also about future risks and rewards. It needs to balance between eating food quickly and avoiding danger.

Existing Solution Methods

- Markov Decision Processes (MDPs): to model the environment and rewards.
- Policy Iteration / Value Iteration: to find the best moves in a known environment.
- Utility-Based Agents: to make smart decisions based on rewards and risks.

- Partially Observable MDPs (POMDPs): if the agent cannot always see the full grid.

Plan for modeling and solving the problem

State Space: Snake's position and direction, food location, enemy location (if visible), grid size.

Action Space: Turn Left, Move Forward, Turn Right.

Observations: Local view of the grid (like a 5x5 area around the snake), direction of food, enemy snake (if visible).

Rewards:

- +10 for eating food
- -100 for dying (crash or touching enemy)
- -1 for each time step (to avoid waiting too long)

Algorithm:

- Use an MDP model to define states, actions, and rewards.
- Apply Value Iteration or Policy Iteration for small grid sizes.
- Use TD Learning for larger or unknown environments.
- Explore POMDPs if the agent has limited visibility.