# **Reinforcement Learning Game Design - Snake Game**

# **Report**

Student Name: HUANG Xinghua

Student ID: 3160617

Github Repository: [GaryHuang666/CDS524-Assignment-1-HUANG-Xinghua](https://github.com/GaryHuang666/CDS524-Assignment-1-HUANG-Xinghua)

Youtube Recording: https://youtu.be/0jG4YUjO9G8

1. **Introduction**

Reinforcement Learning (RL) is a machine learning paradigm in which an agent learns to make decisions by interacting with an environment and receiving feedback in the form of rewards or penalties. Q-learning is a widely used RL algorithm that allows an agent to learn an optimal policy for maximizing accumulative rewards over time.

This project implemented a Snake Game based on Q-learning. The AI-controlled agent which is the snake learns to play the game independently. The objective is to train an intelligent agent to maximize its score by consuming food while avoiding collisions with walls and the agent itself using Q-learning concepts. Through training, AI can learn from past experiences to improve its behavior and make better decisions over time, thereby improving its strategies.

This report provides an overview of the game design, implementation of the Q-learning algorithm, challenges encountered during training and the overall performance of the AI agent.

The primary objectives of this project are: To design a grid-based Snake Game with clear objective and rules, and to implement a Q-learning algorithm to train an AI agent to play the game. Additionally, a graphical user interface (GUI) was developed using Pygame for game visualization. It also documents the implementation process, challenges, and solutions.

1. **Game Design**

The game is a **grid-based Snake Game** where the agent (the snake) must:

* Move across a **10x10 grid**.
* **Consume food (green square) to grow and increase the score**.
* **Avoid game over condition: collisions** with itself or the walls.

The AI agent must learn an **optimal policy** that allows it to **maximize food consumption while avoiding collisions**. The game ends when the snake collides with the walls or itself.

The game environment is represented using a structured state space to enable artificial intelligence to learn effectively. By using a more efficient relative representation of the state instead of using absolute coordinates to represent the positions of the snake and bait. Conditions include whether the food is to the left, right, above or below the snake's head and whether there are obstacles in those directions. Each value is stored as a binary indicator (0 or 1), which simplifies the training process.

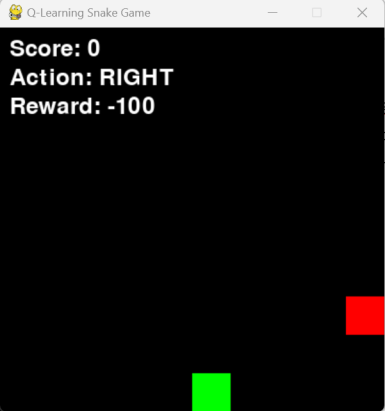
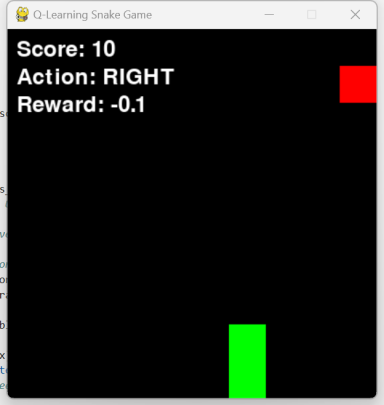
1740475900844

The action space consists of three actions: turn left, turn right, or continue straight. This minimal set of actions allows the AI to navigate the grid efficiently without creating excessive complexity.

1740476085783

A key element of game design is the reward function that guides the learning process. The agent receives a positive reward for eating food. In contrast, a negative reward for hitting an obstacle and a small penalty for each time it walks to avoid unnecessary wandering. Additionally, agents are rewarded when they move toward food which encourages more efficient path finding.

The game features a graphical user interface (GUI) built with Pygame. It providing visual representations of the snake, food, and game state. The interface also displays real-time information such as the current score and the agent’s last action also the reward received for each move.

****

1. **Q-Learning Implementation**
2. learning is a model-free reinforcement learning algorithm that allows an agent to learn an optimal strategy by updating a Q-table over time. A Q-table stores the values of different state-action pairs which represent the expected future rewards when a particular action is performed in a particular state. These values are updated using the following formula: **Q(s,a) = Q(s,a) + α × (r + γ × maxQ(s′,a′) − Q(s,a))**

* α is the **learning rate. It** controls how much new information overrides old values.
* γ is the **discount factor. It** determines the importance of future rewards.
* r is the **immediate reward** received after taking action aaa.
* maxQ(s′,a′) represents the highest Q-value for the next state to ensure the agent aims for long-term gains.

An epsilon-greedy policy is used to balance exploration and exploitation. The agent initially explores the environment by performing random actions. But as training progresses, it gradually evolves to exploit its most familiar moves. The epsilon value starts at 1.0 and decreases to 0.05 over time which allows the AI to fully explore a strategy before optimizing it.

The training process consists of 5000 episodes during which the AI runs through several rounds of the game and continually updates the Q-table. Each episode lasts at most 500 steps to prevent the agent from taking paths that are too long.

1. **Training Process and Performance**

In the initial training phase, the AI moves randomly due to a lack of acquired knowledge. It often crashing into walls or into itself. However, the AI began to recognize patterns and improved its ability to reach food while avoiding obstacles as training progressed.

Between 1000 and 3500 episodes, the agent begins to follow repetitive movement patterns which slows down the learning process. This behavior occurs because the AI has found a sub-optimal but viable strategy that allows it to survive longer but does not necessarily maximize food intake. The agent tends to move in cycles to avoid danger but fails to significantly improve its score instead of actively seeking food. This problem highlights a common challenge in reinforcement learning is the agent can stabilize in a local optimum, meaning it learns a safe but not necessarily the most effective policy.

The AI's performance has not improved significantly after 3500 training episodes. While it can more effectively avoid collisions, it has difficulty consistently acquiring food. The score remains relatively low. It is suggesting that the AI is not effectively optimizing its movement toward food. This learning stagnation suggests that the current reward function and exploration strategy may not be sufficient to push the agent to adopt better policies.



One possible reason for the lack of improvement is the decrease in the exploration rate (epsilon). As epsilon decreases, the agent relies more on its learned policy instead of exploring new ones. The agent will continue to make the same mistakes and will not discover a better approach if the learned policy is not optimal. Another contributing factor may be the state representation. While simplified may not provide the AI with enough information to make effective decisions about food placement.

Additionally, the training time per episode is quite long in some cases. Since the agent does not always prioritize food. So it may take many steps to complete an episode and slowing down the learning process.

1. **Challenges**

The biggest challenge of this project was that the **lack of score improvement** after 3500 episodes. Although the AI learned to avoid collisions, it often followed repetitive movement patterns rather than actively searching for food. This suggests that the agent reached a local optimum by prioritizing survival over reward maximization.

One cause of this problem is the epsilon decay strategy. It quickly shortens exploration and prevents the AI from discovering better strategies. The search can continue for longer by setting the adaptive epsilon decay. Furthermore, the reward function may not provide enough incentive to collect food. Increasing food rewards and introducing penalties for excessive exercise can lead to more efficient play.

Another limitation is that state representations. It may not provide enough information for an AI to navigate efficiently while simplified. Combining distance-based features can improve decision making.

1. **Conclusion**

In this project, Q-Learning was successfully applied to game AI and an agent was trained for the Snake Game. The AI improved its ability to avoid collisions but had problems optimizing its food collection and its learning progress was limited after 3500 training episodes.

The results show that improvements in exploration strategies, reward functions, and state representations are needed to improve performance. Future work will focus on adaptive epsilon decay, reward enhancements, and DQN exploration for more efficient learning. Despite the challenges, this project has laid a solid foundation for future research in the field of reinforcement learning for game AI.