Practice Two

Stephen Griffin: 100559805

Corey Gross: 100559804

Phase Two

Attributes to use

1. **danger_straight** – Is there a collision if the snake keeps going in its current direction?
2. **danger_right** – Would turning right cause a collision?
3. **danger_left** – Would turning left cause a collision?
4. **food_left** – Is the food somewhere to the left of the snake's head?
5. **food_right** – Is the food somewhere to the right?
6. **food_up** – Is the food above the head?
7. **food_down** – Is the food below the head?

Reward Function

We use a composite reward scheme to guide the agent toward food while penalizing unsafe behavior. Eating a piece of food yields a large positive reward (+100), reinforcing the primary goal of survival. Death, whether by collision with a wall or its own body, is heavily discouraged with a substantial negative penalty (−15). For all intermediate steps, we apply a small shaping reward based on the change in Manhattan distance to the food: the agent receives a modest bonus whenever it moves closer and a slight penalty when it moves away. This distance-based reward,

clipped so that the worst step penalty is −0.1, encourages the snake to make progress toward its target without overwhelming the primary eat-or-die signals. The combination of sparse, high-magnitude terminal rewards and continuous, distance-based shaping fosters both long-term strategy and incremental improvement.

Get State Function

We represent the game's situation as a compact 7-bit integer that simultaneously encodes imminent danger and food direction. The three highest bits flag whether moving straight, right, or left would immediately collide with the wall or the snake's own body—providing crucial safety information. The lower four bits indicate whether the food lies to the left, right, up, or down of the head—guiding the agent toward its goal. By shifting and OR-ing these flags into a single byte, we obtain a fast, fixed-size state index that balances environmental awareness (collision risk) with navigational cues (food position) for efficient Q-learning updates.

Update Q-Table

The Q-learning update rule blends prior experience with newly observed outcomes to iteratively refine action values. When the agent receives reward r for taking action a in state s, we first read the old Q-value Q(s,a). If the transition ends in a terminal (death) state—signaled by a negative reward—we zero out any future expectations; otherwise we look up the maximum Q-value achievable from the next state s′. We then perform a weighted average: we decay the old estimate by (1–α) and add α times the sum of the immediate reward plus the discounted future value

($\gamma \times \max_{a'} Q(s',a')$). This balance (controlled by learning rate α and discount factor γ) ensures stable convergence toward the optimal Q–function over many episodes.

Data Collection

Ran 10000 instances to fill q table using only +100 for eating fruit, -10 for running into a wall, and -0.1 for each step individually. After, ran 100 instances with epsilon frozen in order to determine how good the actual snake was. Here is what we have after first attempt at training the snake:

- **Crashes on first move** (–10 or –10.9): 18 runs (~18%)
- **Tiny runs** (≈ –10…–20): 18 runs
- **Medium runs** (≈ 80…200): 34 runs
- **Long runs** (> 200): 30 runs
- **Average total reward**: ~ 260
- **Median total reward**: ~ 188
- **Best run**: 1 167
- **Worst non–first‑move crash** (–11.2 at run 1)

Thinking about reducing first move crashes by changing the instantiation to make sure that the snake is never facing a wall when it spawns in. Decided that this was too manipulative of game principles, and decided to go another route. We changed the reward function of the snake, by rewarding it for getting closer to the fruit, and penalizing it when going further away from the

fruit. For this, we had to change the calculate_reward function, as well as the reset() function (to include the previous distance).

From here, we fine tuned some of the parameters to make the training a little bit smoother. We reduced the alpha value from 0.1 to 0.05 for smoother Q updates. We also raised the discount variable (gamma) to 0.95 since we care more about future rewards now (not only eating the fruit, but also going towards the fruit).

After training another 10,000 instances, but this time with the reward to move towards the fruit, we ran the frozen epsilon on 100 instances again. This time, the results were far more appealing. Here is what the numbers tell us:

- **High‑reward runs (>200)**: ~85% of episodes
- **Occasional total crashes (≈–10)**: ~7%
- **Rare heavy negative shaping (≈–19.9)** or tiny runs (<100): ~8%

However, there are still a few -19.9 runs that are confusing. To handle this, we will change the reward function to take the max of the shaping penalty by doing this:

```
shaped = delta * 0.1 - 0.1
return max(shaped, -0.1)
```

Attributes we use for Epsilon, Gamma, and Alpha

We balance three key hyperparameters to guide learning. α (alpha), the learning rate, determines how heavily new experiences overwrite old estimates. A modest value (0.1) ensures stability, preventing noisy updates from derailing long‑term value trends. γ (gamma), the discount factor, weights immediate versus future rewards; by choosing $\gamma \approx 0.99$ we encourage the snake to both pursue food now and plan ahead for larger cumulative gains. Finally, ε (epsilon) in our ε-greedy policy controls exploration: starting at zero (pure exploitation) but decaying toward a small floor ($\varepsilon_i \approx 0.01$) with $\varepsilon\_decay \approx 0.995$ ensures that, even late in training, the agent still tries occasional random moves, guarding against local optima and stale policies.

After Running a few times

After testing the model a few times, we decided that our attributes were not acceptable for the snake. It was not averaging high enough scores, so we added five more. The new attributes that we added were:

1. distance - separated into four separate “buckets” to determine if the distance is short, medium, long, or very long
2. direction - is the snake currently moving up, down, left, or right

Now, the performance of the snake is far better than it was previously. On a non growing test, the snake collects tens of thousands of fruits before dying. It also collects plenty more fruit on growing tests, and tests with changed board sizes. Here are the exact values:

• Crashes on first move (≈ –30): 0 runs (~0%)

• Small runs (≤ 2000): 17 runs (~17%)

• Medium runs (2000–10000): 51 runs (~51%)

• Long runs (>10000): 32 runs (~32%)

• Average total reward: ~10,345

• Median total reward: ~6,184

• Best run: 61,884 (at episode 79)

• Worst run: 274 (at episode 19)

Change of Reward

Now that we changed the states, we decided to change the reward to lower the number of times that the snake runs into the wall, or crashes into itself. Our original reward function consisted of +100 for collecting a fruit, and -10 for dying. This led the snake to collect a couple of fruits, and then run into the wall because the penalty was not strong enough for the snake to not want to die. We change the penalty to **-30 for dying** by crashing into a wall or into itself, and the reward for **collecting a fruit to +10**. We also decided that there should be a **reward for going towards the fruit, and a penalty for moving away from the fruit**.

Phase Three

Overview

In this section, we discuss the qualitative and quantitative results of the agent's behavior trained using Q-learning for the Snake game. We also added the attribute for determining if the snake was going to run into itself (given that it now grows). The agent was tested under various conditions, including different board sizes (150x150, 300x300, and 450x450) and with both growing and non-growing snake bodies. The aim was to evaluate the agent's capability in terms of survival, efficiency in collecting food, adaptability to different environments, and consistency of learned behavior.

Quantitative Results

We evaluated our snake under eight conditions:

• 150×150 growing snake

• 300×300 growing snake

• 450×450 growing snake

• 150×150 non-growing snake

• 300×300 non-growing snake

• 450×450 non-growing snake

150x150 Board – Growing Snake

In the smaller growing board (150x150), the snake performed moderately well after training. The average reward hovered around **300-400** points, with a maximum of **over 500** in many games. Only a few early deaths occurred, mainly due to unfortunate initial conditions or unexpected self-collisions as the snake length increased.

300x300 Board – Growing Snake

On the **300x300** growing board, the results were even more encouraging. Average rewards were consistently around **800–1000** points, with several games breaking the **1,500–2,000** mark. The larger space allowed the agent more room to correct mistakes, resulting in fewer deaths from tight turns or sudden walls.

450x450 Board – Growing Snake

When expanded to a **450x450 growing** environment, the snake showed impressive robustness. Frequent scores above **1,500** were observed, and the highest reward recorded was **3,255**, showing that the snake could navigate complex scenarios even as it grew larger. Interestingly, deaths here were rarely from random crashes but usually from very subtle mistakes in pathing late into games, when the snake occupied a substantial portion of the board.

Non-Growing Snake Performance

Significant differences were observed when testing the agent with a non-growing snake.

- The non-growing tests yielded astonishing results. On **150x150 without growth**, the snake could survive for extended periods, often exceeding **5,000–10,000** points. The highest score observed was **over 21,000**, meaning the snake collected **over 2000 pieces of fruit without dying**. Similarly, on **300x300**, many runs surpassed **30,000 points**, and on the **450x450 non-growing** board, scores of **40,000–60,000** were typical, with a peak at an absolutely massive **126,000 points**.
- Clearly, the non-growing snake approached "perfect play" limited only by the max steps or sheer board exhaustion. Growing snakes, while not perfect, still performed at a highly competent level, especially considering the exponential difficulty added by their expanding bodies.

Qualitative Analysis

Qualitative observations revealed that the agent gradually developed strategic behaviors that effectively balanced immediate rewards (fruit consumption) and long-term survival (collision avoidance). Initially, the snake demonstrated inefficient or suboptimal paths, frequently engaging in unnecessary loops or movements away from food. However, as training progressed, it increasingly adopted direct and efficient paths toward food.

The agent's collision detection attributes (danger straight, danger right, danger left, body collision) significantly improved its decision-making skills. It avoided walls and self-collisions consistently, although occasional misjudgments still led to abrupt game-ends. The inclusion of direction-based and distance-based states allowed the snake to refine its strategies further, demonstrating anticipation of future rewards rather than purely reactive movements.

Adaptability and Robustness

The agent's performance on varying board sizes suggests a high degree of adaptability. Smaller boards exhibited more immediate challenges due to the proximity of walls, causing relatively frequent collisions initially. Larger boards allowed the agent to better demonstrate strategic depth, navigating more freely and planning optimal routes. The impressive performance of the non-growing snake across all board sizes further illustrates that the primary challenge for the agent lies in adapting to dynamic environments (the growing snake scenario).

Conclusion and Recommendations

In this assignment, we successfully developed a robust Q-learning agent capable of effectively navigating the Snake game environment. Through iterative refinement of state representations, reward functions, and hyperparameters (α, γ, ε), we substantially improved the snake's decision-making capabilities. The integration of distance-based rewards, collision-awareness attributes, and directional state encoding led to marked improvements in both survival duration and score accumulation. Quantitative testing showed consistently strong performance across varying board sizes and conditions, with great outcomes particularly in non-growing scenarios, where the snake approached optimal play. Qualitatively, the agent exhibited strategic foresight and adaptive behaviors reminiscent of human decision-making. Future improvements could explore advanced reinforcement learning techniques such as deep Q-learning or policy gradients, potentially addressing remaining minor weaknesses like late-game collisions in growing environments. Overall, the implemented Q-learning methodology demonstrated notable effectiveness, providing a solid foundation for further research into intelligent agent design.