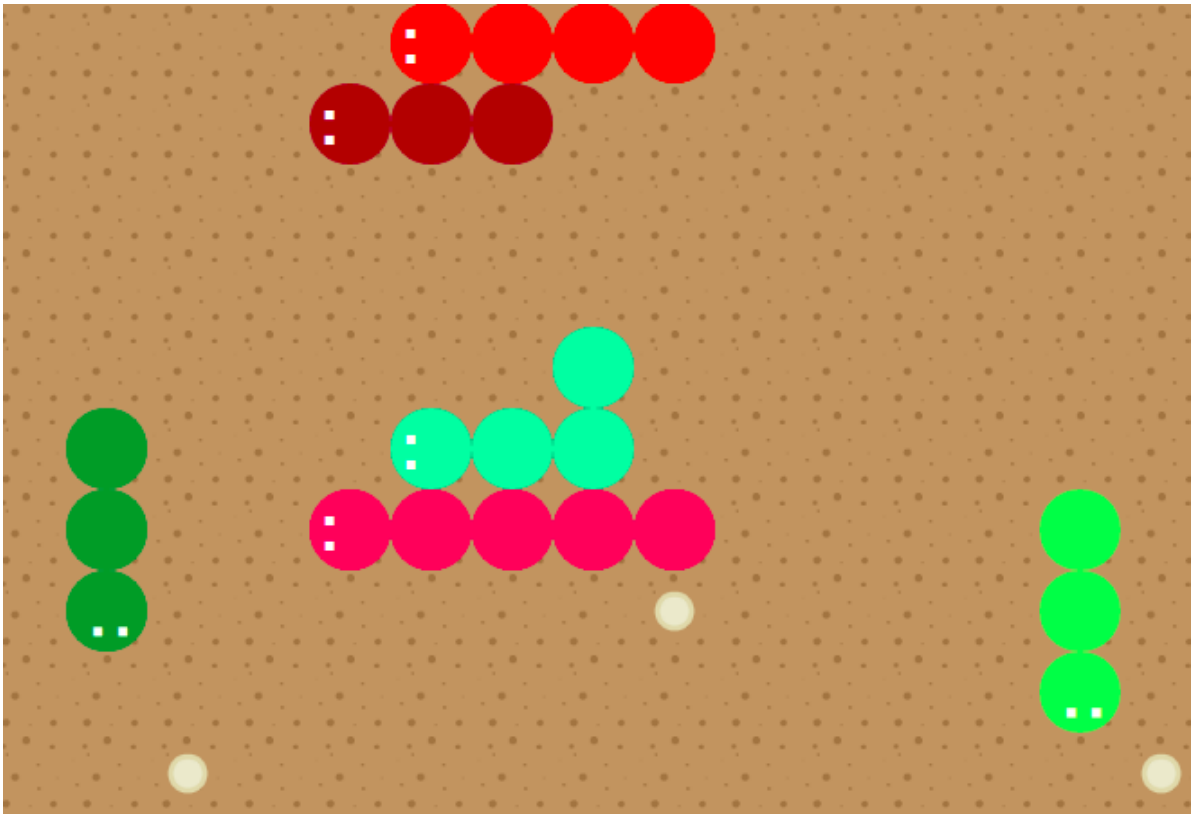
Project: Snakes 3v3

> CS410: Artificial Intelligence
> Shanghai Jiao Tong University, Fall 2021
> Due Time: 2021.12.29 23:59 (for presentation slides) / 2022.01.02 23:59 (for all materials)

Introduction



In this project you will play Snakes 3v3, which is a multi-agent version of the traditional Snake game. The game is played by controlling the direction of the snakes' head to eat beans, thus making the snakes longer and longer.

Rules

1. There are **two** parties in this game, and each player manipulates **three snakes** in a grid of $n \times m$. n is the vertical height and m is the horizontal width.
2. A snake is a finite, non-repeated and sequential sequence composed of a series of coordinates. The adjacent coordinates in the sequence are adjacent, that is, the x-axis or y-axis coordinates of the two coordinates are the same, and the first coordinate in the sequence represents the snake head. The player can only control the snake by controlling the **head's orientation** (up, down, left and right).
3. The snake moves forward at a **constant speed** (forwarding means inserting the head of the sequence into the next coordinate in the direction in which the head of the snake points, and deleting the coordinate at the end of the sequence). The initial position of the snake is a random position in the grid, and the initial length is **3**. The length of each bean eaten increases by **1**. In the game, the number of beans remains constant (5), that is, if they are eaten, the same number of beans will be randomly generated.
4. Snake's heads in the body of its own (the sequence is repeated), and the body of other snakes (with the same coordinates as other sequences) are judged to be **dead**. When an

illegal direction is entered (such as moving to the left, you cannot choose to the right), a legal direction is randomly selected. Unlike traditional games, when the snake head crosses the boundary, it can cross to a symmetrical position, which is not counted as death. When the snake dies, it can be reborn according to the initial settings.

5. The game ends when the specified number of steps is reached. After the game is over, the party with the **larger sum of the length of the snakes wins**.
6. Observation is a dictionary with key from 1 to 7 and `board_width`, `board_height`, `last_direction`, `controlled_snake_index`. Here, 1 represents beans, while 2 to 7 represent snakes. The value of the dictionary is a list with $[h, w]$ coordinates as elements, representing positions of beans or snakes. h represents the vertical distance from the origin in the upper left corner, w represents the horizontal width from the origin. Among the values corresponding to 2 to 7, the list elements from left to right represent the position of the head to the tail of the corresponding snake; The value of `board_width` is the width of the board; The value of `board_height` is the height of the board; The value of `last_direction` are directions of each snake in the last step; The value of `controlled_snake_index` is the controlled snake index.
7. The `action_space` is a list with length of `n_action_dim`. Here, `n_action_dim=1` and every element in it is a Discrete object in Gym (Discrete Link), like `[Discrete(4)]`.
8. For more details, please refer to the source code below.

Source

Clone or download the repository Competition_3v3snakes to get started.

Evaluation Guide

1. The evaluation platform is now available. Please **sign up the competition** with:
    1. Team Name: Group xx (please make sure it is in accord with that in the sheet)
    2. Organization: SJTU CS410
    3. Member1: Jidi account of member 1
    4. Member2: Jidi account of member 2
2. To evaluate your algorithm, we open several warm-up rounds. At the end of each round, your submissions during this round will be evaluated against the baseline opponents. We will periodically setup several baseline opponents (with ever-improving performance) on the platform for you to evaluate performance and improve your algorithm.

    Detail:

    - Warm-up 1st Round (Submission Deadline): 2021.11.24 23:59
    - Warm-up 2nd Round (Submission Deadline): 2021.12.03 23:59
    - Warm-up 3rd Round (Submission Deadline): 2021.12.04 23:59
    - Warm-up 4th Round (Submission Deadline): 2021.12.07 23:59
    - Warm-up 5th Round (Submission Deadline): 2021.12.16 23:59
    - Warm-up 6th Round (Submission Deadline): 2021.12.17 23:59
    - Warm-up 7th Round (Submission Deadline): 2021.12.21 23:59
    - Warm-up 8th Round (Submission Deadline): 2021.12.25 23:59
    - Warm-up 9th Round (Submission Deadline): 2021.12.29 23:59
    - Warm-up 10th Round (Submission Deadline): 2021.12.30 23:59
    - Warm-up 11th Round (Submission Deadline): 2021.12.31 23:59
    - Final Round (Submission Deadline): 2022.01.02 23:59

Important: It is not compulsory to participate in warm-ups. Submissions for warm-up rounds are not evaluated immediately. If you want to obtain more feedbacks, you can submit via the environment channel, which provides immediate feedbacks for competitions with random user opponents.

3. During verification and evaluation, the platform runs the user code on the single-core **CPU** (**GPU** is not supported temporarily), and limits the time for the user to return action in each step to no more than **2s** and memory to no more than **500M**.
4. **Only the final submission counts towards your performance score of this project**. Specifically, we will rank your algorithms based on the **last submission** during **Final Round** (2021.12.31 23:59 -> 2022.01.02 23:59). The performance score for this project will only depend on the final rank of all the teams. The final evaluation period will adopt the Round-robin Tournament format to rank teams.
5. To submit your algorithm to the platform, you only need to provide one single file named `submission.py` containing `my_controller()`. Here is an example.

Important: The deadline of final submission is **2022.01.02 23:59** and late submissions will **NOT** be accepted.