Protocol for the BI-ZUM and BI-PYT Semestral work

authored by Nikita Mortuzaiev (mortunik)

Task specification

This project is an implementation of the popular “Snake” game for FIT CTU in Prague.

The rules of the classic game are quite simple. The main character of the game is the Snake (which at the start consists of one or three square tiles) that is constantly moving forward in the snake-like manner (the body follows the head). The user is given the access to the controlling of it by pressing WASD or ARROWS buttons on his/her keyboard. The buttons change the direction of the snake so that it won’t die by collision with the wall or with its body. The snake will grow (the body elongates by one tile) if the snake’s head reaches the position of the spawned fruit. The fruit then spawns again at the new random position.

The application itself consists of several parts among which are:

- **Main Menu**

Gives the user the possibility to choose the mode of the game. Available modes are: Classic Snake or Snake AI.

- **Classic Snake**

In this mode the user’s input from the keyboard is accepted by the game giving him/her control over the Snake.

- **Snake AI**

This is the mode that runs the Genetic algorithm that is followed by the animation of the best individuals’ choices. More about this mode below.

The method of solving

In brief, the Genetic algorithm was chosen for this task as the main evolution technique for the Snake AI mode. In this mode, along with the animations of the Snake and the spawns of the new Fruits, the Evolutions (with the small input parameters as the population size, number of generations etc.) are run enabling the game to choose the best outcome for the Snake.

The best outcomes are defined by choosing the best individual from the evolved population of Snake Individuals. Every Snake Individual has its own brain represented by the Neural Network with 12 inputs, two hidden layers of size 8 and 3 outputs. To calculate its fitness, each Individual should determine the parameters of the Snake it is controlling at the moment. These parameters are the 12 inputs for the Neural Network and specified in this order: the distance from the snake’s head to the left wall, to the fruit to the left and to the first body tile to the left. Then the “Left” direction is changed by the “Up”, “Right” and “Down” directions. The three

outputs of the Individual's brains specify the direction that the Snake should choose for the next iteration of the game. The first output tells the Snake to turn left, the second – to move further forward and the third – to turn right.

However, not only the Individual's brains are used to determine its fitness. The distance between the new Snake's head and the Fruit is also computed and used in the code along with the ratio of the tiles reachable from the new head. The number of reachable tiles is computed with the help of Breadth-First Seacrh (BFS) algorithm.

The choice of the method of solving

The method of solving described above was chosen because it provides the best outcomes for the Snakes controlled by the Artificial Intelligence. The fitness function for the Snake Individuals is used to determine the best individual in the population, whose brains will provide the best next move for the current Snake's state. The BFS algorithm was chosen to count the reachable nodes as the best and most popular choice for solving this problem.

Experiments

The application was tested with different parameters for the game, the Genetic algorithm and the Neural Networks. All those parameters along with the game's UI parameters are currently saved and marked as constants in the utils module. More experiments with the Genetic algorithm were widely described in the task #4 for the BI-ZUM subject.

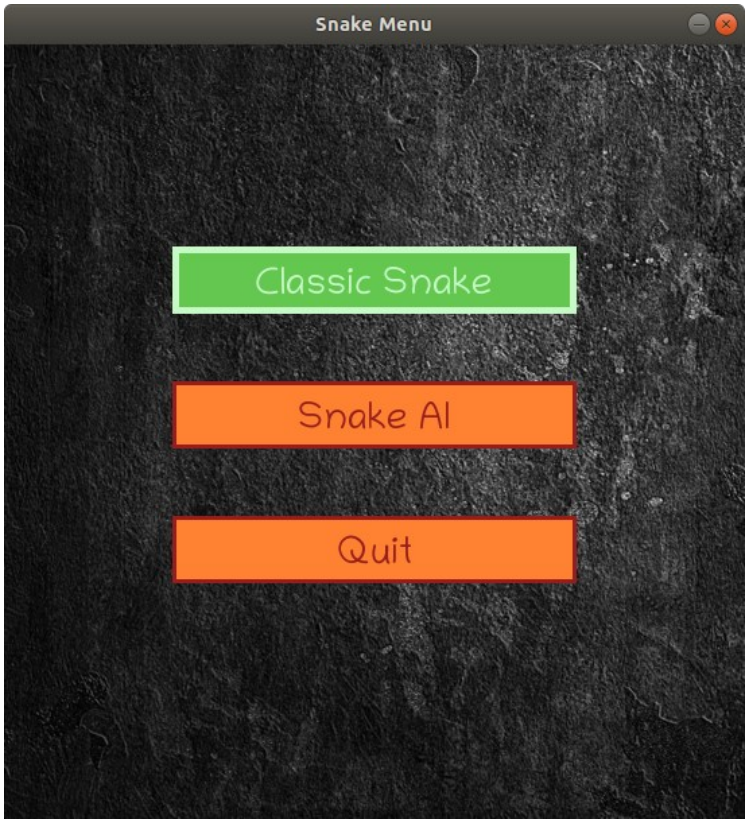
Some tests are also provided in the tests package to test the code with the help of pytest library.

Discussion

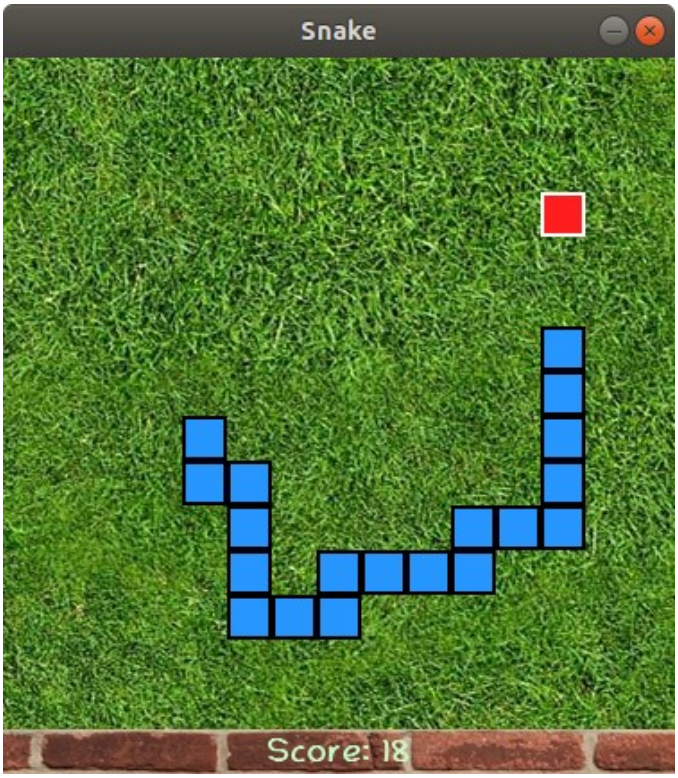
The problem solved in this task is neither new nor solved originally. There are a lot of other methods of creating the Artificial Intelligence for the Snake game and there are also methods of reaching the end of the game by filling the board with the Snake's body so that the new Fruit has no free nodes to spawn on. One of the solutions for creating the "perfect" game is to implement a Hamiltonian cycle for the game's board and follow it with the Snake. This provides a "perfect" game ending, but the task itself becomes a problem of finding a cycled graph rather than creating an Artificial Intelligence.

■ Examples

In this part, some screenshots and examples are provided. The screenshot of the Main menu:



The screenshot of the Snake AI game state:



Possible improvements

Experiments with the application showed that the Snake AI mode is running very slow for the boards larger than 20x20 tiles. So, the code might need some optimizations for the current method of solving or even implementing a better method to reach better results in consumed time and memory.

Conclusion

This project was my first game creating experience with Python. I learned the pygame library and improved my skills in OOP and Python programming.