EVAC Assessment 1

Exam No: Y3868718

1 Methods

1.1 Changes to Provided Code

While the snake game discussed in this report is functionally the same as the initially supplied code provided with the assessment, some large changes have been made to the structure. Most notably, the snake game and neural network has been split into multiple separate scripts for ease of development. These files are detailed as follows:

1. `code/displayGame.py` - This file contains the `DisplayGame` class, the only change to this class is changing the snake body colour to be only green to remove any confusion on which way the snake is facing.

2. `code/runGame.py` - This file contains the `runGame` function, other than the code added to facilitate AI snake control into this, other changes have been made. A guard was added to the check of if the snake has eaten food to see if the snake has completed the game. Game completion is defined as there being no available spaces on the board to place food as the snake is taking up all available grid positions. There are also changes to the clean-up of the Python turtle to prevent Python kernel crash.

3. `code/snake.py` - This file contains the `snake` class. While sensing functions have been added to this class, the only change to initially provided functions has been to use `self.XSIZE` and `self.YSIZE` instead of the global `XSIZE, YSIZE`.

1.2 Chosen Evolutionary Method

Genetic Programming was used to evolve weights in a neural network. Input nodes to this neural network were the snake sensing functions, with four output nodes representing snake direction; left, right, up and down. A single hidden layer was used for the neural network with the number of hidden layer nodes corresponding to the sum of the input and output layer nodes.

1.3 Development of Different Snakes

Three differing snakes were developed for the snake game and will be referred to in this report as:

1. `local-sensing`

2. `space-aware-sensing`

3. `n+b-evolution`

Two directions of investigation were performed on the snake game. The first direction investigated the effect of differing collision sensing functions (`local-sensing` against `space-aware-sensing`) while the second investigated changing the generation selection method (`local-sensing` against `n+n-evolution`).

1.4 Sensing Functions

Sensing functions were created for the snake to use to learn about its environment, these can be classified into two groups; food sensing functions and collision sensing functions. All sensing assumes grid format of `[y,x]` and directions are relative to the grid, not the current direction of the snake.

1.4.1 Food Sensing

Food sensing was constant throughout all developed snakes. Multiple methods of food sensing were explored during development including passing the exact position of the food, calculating the manhattan distance between food and snake, and sensing the quadrant of the grid the food was in relative to the head of the snake. The chosen method instead simply checks if food is in the direction specified, for left and right the x axis of the snake is checked while up and down checks the right axis. The sensing function returns a boolean value; True if food is present and False if not.

If the Snake is directly in line with food in that direction, both directions on the axis will return False. For example, a snake head at position [4,4] and food at [6,6] would return True from `downFood` and `rightFood`, and False from `upFood` and `leftFood`. A snake head at position [4,4] and food at [6,4] would return True from `downFood` and False from all other sense directions.

1.4.2 Collision Sensing

Two methods of collision sensing were developed, the first method of collision sensing was used by both `local-sensing` and `n+n-evolution`. Each direction has two collision sensing functions that both check based on a given position; one used to sense if the position is within a wall, and the other used to sense if the position is within the snake body. Both of these sensing functions return True if a collision is detected, False otherwise. These positions are given for each direction by calculating the position of the snake head if the snake moved in the given direction. For example, if the snake head is at position [4,4], the position given to `senseWall` and `senseTail` would be [4,3] when `leftWall` and `leftTail` are called.

These wall and tail functions could be combined into a single function to detect if a collision has occurred, however this was not implemented as providing both collision detection components separately allows a snake to learn separate behaviors depending on collision type, for example chasing it's tail or circling the grid perimeter

2 Conclusions

2.1

3   Results

3.1