Game 10

Genetic Algorithm

Genetic Algorithm ~ Introduction

- The main task in relation to game 10 was to implement a genetic algorithm which through several generations improves itself until it can learn to safely land the spaceship in the indicated area.
- This type of algorithm utilises genes which compose the chromosome of each agent. Through breeding between the agents the genes change and improve overtime thus, allowing the agents to improve upon the previous generations.
- The main components required for the implementation of a genetic algorithm are a brain to control the agent, a generation controller to manage each generation and the DNA of each agent.

Genetic Algorithm ~ AI Explanation (1)

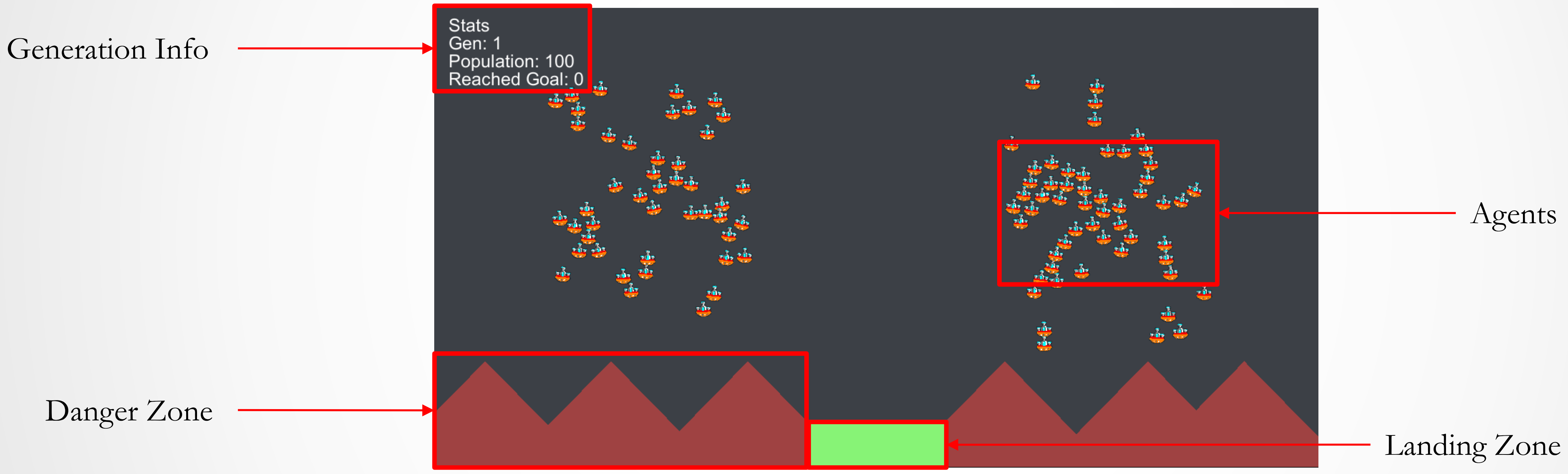
- The basis for any genetic algorithm is the evolutionary loop which goes as follows:
    1. The population is instantiated.
    2. The agents in said population compete so as to achieve their indicated goal
    3. The agents which come close to or achieve their goal are given a higher fitness than those that don't.
    4. These agents are then chosen to foster the next generation.
    5. Over time natural selection causes an overall rise in the fitness of the population.

Genetic Algorithm ~ AI Explanation (2)

- DNA – The DNA of the agent is composed of several genes. A gene is just a value which is used in some function. In the case of the spaceship it's DNA is composed of 14 genes, were each gene represents a thrust or rotation value. These are then called upon when required by the brain.
- Brain – The Brain serves as the agent controller, it consist of a series of checks which when triggered call upon a specific action to be carried out. For instance when the ship detects harmful ground it activates its thrusters to move itself upwards.
- Generation controller – This controller is crucial as its job is to breed the agents in the current generation so as to create a new and improved successor generation. This is achieved through some selection criteria in which the best agents are selected for breeding.
- **Note**: Further complexity can be added to the above scripts such as elitism and mutations.

Genetic Algorithm ~ Mini-Game Implementation (1)

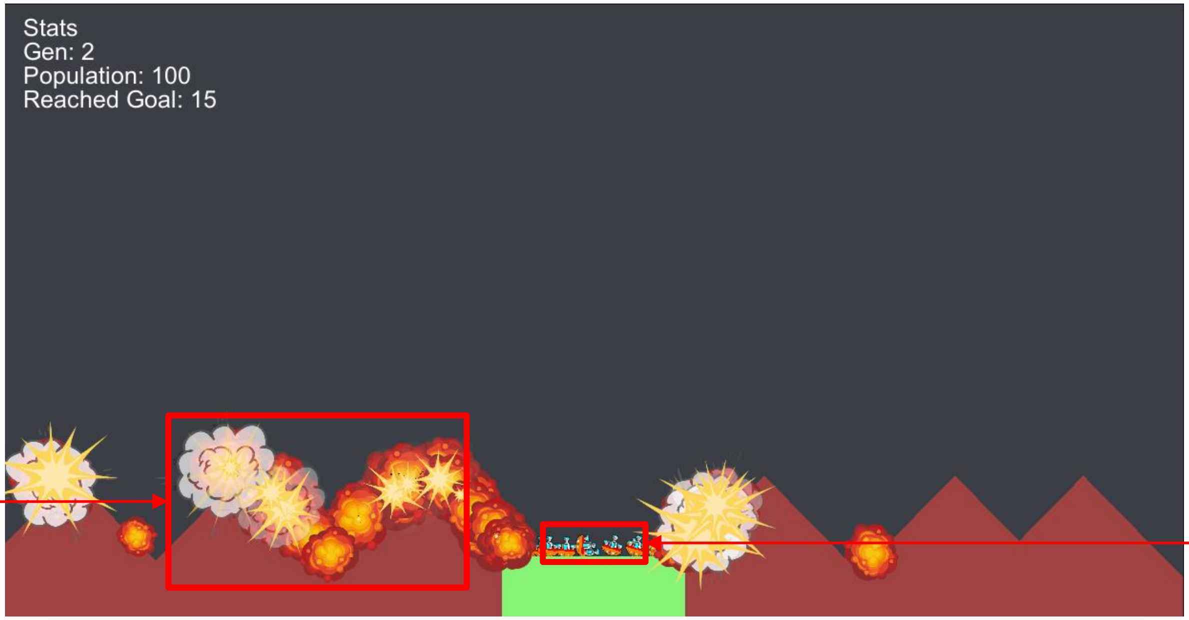
Playable Area:



Stats
Gen: 1
Population: 100
Reached Goal: 0

Genetic Algorithm ~ Mini-Game Implementation (2)

Playable Area:



```
Stats
Gen: 2
Population: 100
Reached Goal: 15
```

Crashed Ships

Successfully landed

Genetic Algorithm ~ Mini-Game Implementation (3)

- The implementation of the GA was inspired from [1] and utilised sprites from [2, 3]. It required three main scripts these being the Brain, DNA and Population.
- Brain – This script includes:
  - The Init() method initialises the agent by assigning it its spawn position, fuel amount and its DNA sequence.
  - The OnCollisionEnter2D() method is used to detect collisions which the environment so as to reward the agent based on its actions. For instance if the agent lands in the landing zone they get a positive reward whilst a negative reward is assigned based on how quickly the agent crashed.
  - The Update() method is used to generate the ray casts from the agent which allow it to detect its environment and thus react accordingly by activating specific statements which denote specific actions. For instance if the left boundary of the playable area is detected the agent will to turn and fly away. The turning amount and thrust level applied would then be based on the respective genes. This method also updates the time alive and fuel amount of each agent. The greater time alive the greater the positive reward whilst if the fuel hits zero the agent can no longer move.
  - The FixedUpdate() method is what applies the aforementioned actions onto the agent.

Genetic Algorithm ~ Mini-Game Implementation (4)

- DNA – This script includes:
  - The DNA() method initialises the DNA by specifying its max length and the maximum that each gene value is allowed to go. The SetRandom() is than called to determine the initial values of each gene.
  - The SetRandom() method generates random values for each gene based on the max value denoted in the DNA() method.
  - The Combine() method randomly chooses values from two passed DNA objects so as to create a new DNA sequence. This method is used to facilitate breeding between two agents.
  - The Mutate() method randomly changes a series of gene values in the specified DNA sequence.

Genetic Algorithm ~ Mini-Game Implementation (5)

- Population – This script includes:
  - The Start() method initialises the initial population.
  - The Breed() method generates an offspring based on the parent objects passed. This is achieved by using the Init() and Combine() methods, note however there is a slight chance that a DNA mutation will occur. (The chance can be increased/lowered by changing the mutationChance variable)
  - The BreedNewPopulation() method sorts the agents based on their reward value. This is than followed by the elitism component in which a percentage of the best agents are chosen and copied over to the next generation (This is achieved by breeding the agent with itself). The breeding component involves using a tournament selection in which the resultant agents are bred together to create offspring and finally a random parent selection is carried out so as to avoid converging to quickly.
  - The Update() method is used to call the BreedNewPopulation() method when all agents run out of fuel.
  - The onGUI() method is used to display useful information to the user such as the current generation, the population size and the number of agents that reached the goal.

Genetic Algorithm ~ Exercise (1)

Now it's your turn to Code ! – Let's implement the Genetic Algorithm ☺

Open the Population script and implement the following code in the Breed() method.

Note: This method serves to breed two agents together to create an offspring whilst also allowing for a chance for the offspring to mutate

1. Generate a random number between 0 and mutationChance and check if said value is equal to 1. This will result in a 1/mutationChance probability, if this is the case than call the Init method on b and pass the fuelTime parameter. Then call the Combine function on the two parents dna, so as to create an offspring. Then call the Mutate function on b. Finally do the same thing excluding the call to Mutate in the else statement.
   - (Hint: Use b.dna.Combine(), parent.GetComponent<Brain>().dna, b.dna.Mutate()
2. Return the offspring

Genetic Algorithm ~ Exercise (2)

Open the Population script and implement the following code in the BreedNewPopulation() method.

Note: This method serves to produce the next generation of agents

1. Loop according to the elitism rate and Breed the top agents with themselves, this is done so as to achieve elitism. The same result can also be achieved by copying the elite agents into the new population.
2. Loop according to the randomRate and Breed a select few random agents with eachother, this is done so as to reduce early convergence.
3. While the population count is less than the specified popualtion size, choose two indices one of which according to the elitismRate the other according to tournament selection. The latter can be achieved by generating a series of indices and choosing the fittest one. Once the two indices have been chosen breed them together and add them to the population.

Genetic Algorithm ~ Conclusion

- In conclusion genetic algorithms attempt to mimic evolution as seen in nature so as to implement a problem solution which might not be as obvious otherwise.
- The benefits of genetic algorithms include that they don’t require information about the environment at hand to achieve a satisfactory solution.
- A negative however is that the parameters assigned to said agents need to be thoroughly though out as otherwise this could lead to the agents learning a different behaviour instead of the intended one.

Genetic Algorithm ~ References

[1] – Prof. A. Dingli, ICS2211: "GENETIC ALGORITHMS IN GAMES" [Online]. Available: https://www.um.edu.mt/vle/pluginfile.php/1131102/mod_resource/content/1/Genetic%20Algorithms%20in%20Games.pdf [Accessed: 18-Mar-2023]

[2] Simple Spaceships, Unity Asset Store. [Online]. Available: https://assetstore.unity.com/packages/2d/textures-materials/simple-spaceships-81051[Accessed: 18-Mar-2023]

[3] 2D Flat Explosion, Unity Asset Store. [Online]. Available: https://assetstore.unity.com/packages/2d/textures-materials/2d-flat-explosion-66932[Accessed: 18-Mar-2023]