Predator and Prey – Coursework 3

Contributors' Names

'The names and student numbers of all students who worked on the submission'

- Person 1 – Hissan Omar; K23136072
- Person 2 – Kishan Prakash; K23153494

Description

'A description of your simulation, including the types of species that you are simulating, their behaviour and interactions.'

Overview

This program simulates the process of natural selection occurring over an ecosystem consisting of 6 different species:

- Plants
- Squirrels
- Deer
- Wild boars
- Wolves
- Bears

With some species being dedicated as predators whilst others are prey. The species all get subjected to external scenarios such as genetic illnesses, viral infections and even death by overcrowding, and the simulation shows the long term effects of this for the ecosystem generation by generation.

Species type and Observations

Species	Type	Behaviours and Observations
BEARS WOLVES	Predators	In this ecosystem, predators are carnivorous and they only eat prey. From several observations, the predators are always the first species to become extinct – typically around the 50th generation, with the wolves being the first to go. There are always a lot less predators than there are plants and prey at any time, even from our designed starting point. Food levels: - Bear = 15 | Wolf = 9
WILD BOARS DEER SQUIRREL	Prey	In this ecosystem, the prey are all herbivores that only feed on the grass. Given that whenever any species passes away, it is always replaced by a plant, there seems to be a stable food supply for all the prey at any time. Food levels: - Deer = 12

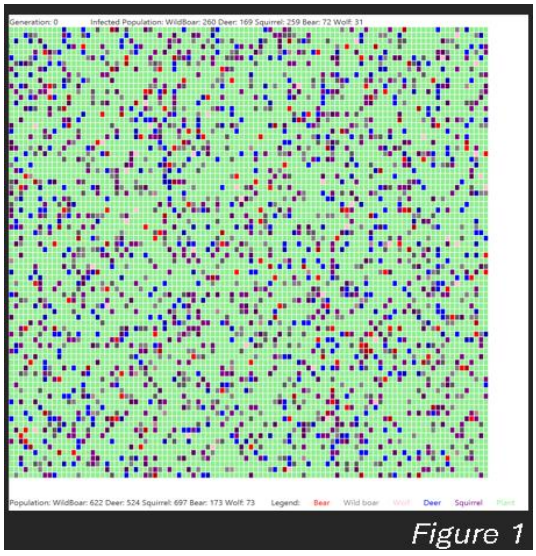
		- Wild Boars = 10 - Squirrels = 8

Technical Implementation

Task 1

'Your simulation should have at least five different acting animals. At least two of these should be predators (they eat another species), and at least two of them should be prey (they eat plants).'

We have successfully simulated 5 animals, with 3 prey and 2 predators, which can be visually seen in figure 1 through the 5 different colours, each colour representing an animal. The implementation we chose to do so was by creating an animal parent class with 5 child classes for each species that inherits a lot of base functionality. You can refer to those individual classes for the code.



Task 2

'Simulate plants. At the start of the simulation, plants should appear in any cell that isn't occupied by a predator or prey. Plant cells should be green. Plant cells are considered free cells and can be replaced by newborn predators or prey. Dead animals should be replaced with plants.'

We have successfully simulated plants and the ability for predators/prey to replace them as well. To demonstrate this, view this change from gen 0 (figure 2) to gen 1 (figure 3) below, from which it is visible how the animals take over the green free cells. The implementation for these can be seen in SimulatorView's method updateCanvas() as well as in Animal's setDead().

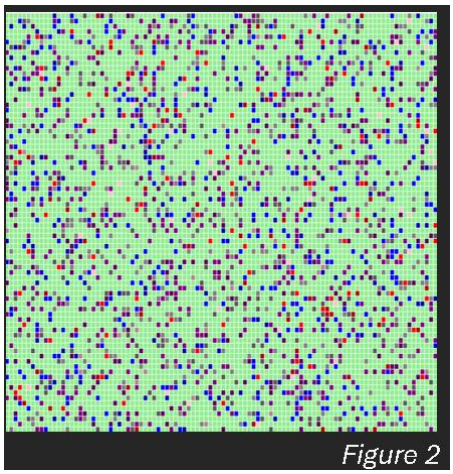


Figure 2

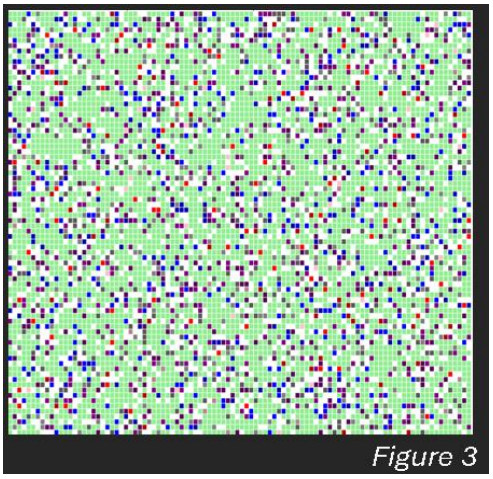


Figure 3

Task 3

'Predators should compete for the same food source, i.e., predators eat prey.'

We have successfully implemented this through the findFood methods in both the predators class, which dictates the food those species eat, and since it's the same for both, the predators will be competing for the same prey.

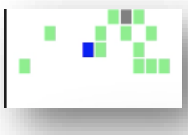
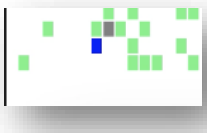
Task 4

'All prey eat plants and have a food value. For example, currently the food value of the rabbit is 9 units.'

We have successfully assigned food values to each prey:

- Deer has a food value of 15
- Wild Boar has a food value of 9
- Squirrel has a food value of 5

This can be seen in each of the prey's class through the method getTheFoodValue().


As you can see, the deer is the blue square in the first picture. In the second picture it moves right and replaces the position of the plant, then in the third picture it goes to a different location leaving an empty cell behind, indicating that the plant within that cell was consumed.

Task 5

'Simulate disease. Some animals occasionally become sick. Once sick, the animal should remain alive for a fixed number of steps (you decide how many time steps). Disease can spread to other animals of the same species. Implement disease as a boolean variable that becomes true when the animal is sick.'

We have successfully managed to implement disease into our ecosystem as well. At the beginning of each simulation there are always animals which pertain diseases.

This can be seen along the top key of our simulation, which shows how many animals in each species are currently alive and infected, also as shown in figure 4:

Figure 4

The infected population for our model generally seems to be around 30-40% of the population at the beginning and to compare that to 10 generations/steps later:

Figure 5

which is showing a great decrease in the infected population, proportionally with the actual population loss as wolves and bear decreased the most relative to their starting populations too. This shows that our disease modelling is working as it should be.

To see the implementation of this, look at the Animals class, in which we have the isSick Boolean variable, and multiple methods such as spreadDisease() related to it.

Task 6

'You will simulate natural selection in your project using elements of evolutionary algorithms. More specifically, you will add a genetic component to the project and implement a mating operation (with mutations)'

We have successfully simulated a basic natural selection process by encoding breeding age, lifespan, breeding probability, litter size and metabolism into each creatures (14 digit) genetic makeup. The implementation for this can be seen in the Animal class through generateRandomGene() and the restraints of which can be seen via parseGene().

We also took into account mutations and mating through the methods mutateGene + mateWith, found in the Animal class, alongside createOffSpring which is found in each of the species classes.