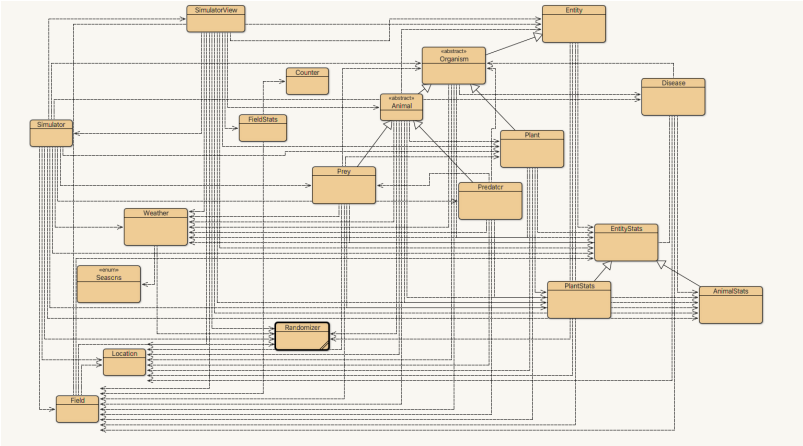
Assignment 3: Predator/Prey Simulation REPORT

By Cosmo Colman (K21090628) & Syraj Alkhalil (K21007329)

Simulation Description:



The Predator/Prey simulator is a full customisable simulation that gives the user full creativity when creating a simulation. The default state of the simulation gives the user 5 entities (2 predators, 2 prey and 1 plant) that they can configure to their liking. The survivability of each entity is dependent on their configuration and the food chain works where predators eat the prey and prey eat the plants. They all also need water to survive which they can get from weather simulation. There are also diseases which may affect the lifespan of an entity.

Species

Species	Behaviour	Interactions
Mouse	The mouse has 20% chance of spawning, with a breeding probability of 1 at the breeding age 1, can live up to age 200, has a hunger value of 200 and a max litter size of 59. Has the colour magenta	The mouse gets eaten by the wolf and the eagle and will eat grapes. Weather affects the water levels of this animal, disease may infect and kill this animal
Deer	The deer also has a 20% chance of spawning with a breeding probability of 1 at the breeding age 1, can live up to age 200, has a hunger value of 200 and a max litter size of 58. Has the colour yellow	The deer gets eaten by the wolf and the eagle and will eat grapes. Weather affects the water levels of this animal, disease may infect and kill this animal
Wolf	The wolf has a 1% chance of spawning with a breeding probability of 0.37 at the breeding age 15, can live up to age 130, has a hunger value of 20 and a max litter size of 2. Has the colour red	The wolf eats all of the prey below it, this includes the Mouse and the Deer. The wolf can't be eaten by any other animals and as a result it competes with the Eagle as the top predator in this food chain. Weather affects the water levels of this animal, disease may infect and kill this animal
Eagle	The wolf has a 1.4% chance of spawning with a breeding probability of 0.28 at the breeding age 16, can live up to age 150, has a hunger value of 16 and a max litter size of 2. Has the colour blue	The eagle, the same way as the wolf, eats all of the prey below it, this includes the mouse and the deer and as a result it competes with the wolf as the top predator in this food chain. Weather affects the water levels of this animal, disease may infect and kill this animal
Grapes	Grapes have a 1% chance of spawning with a breeding probability of 0.01 they offer a food value of 15 and can reach a maximum level of 5. Has the colour green. All of the above stats are adjustable by the user for all animals.	The only real interactions a plant has is with the weather, to be able to sustain itself using photosynthesis and downfall, and with the herbivores, in this case the mouse and the deer only. Additionally, disease may infect and kill the plant.

Base Tasks:

Your simulation should have at least five different kinds of acting species. At least two of these should be predators (they eat another species), and at least two of them should not be predators (they may eat plants). You can either treat plants as if they're always available or simulate their growth/death (see challenge task).

As outlined in the table above there are 5 total species in the simulation. These are Mouse, Deer, Wolf, Eagle, and grapes. There are two predators these being the Wolf and the Eagle, and there are two prey these being the

Mouse and the Deer. In addition to one plant that is the Grapes. The types of entity on the field are dependent on their stats (EntityStats) class which contains all the constants for that animal. Some of these values can be changed during the simulation by the user. The values which do change during the simulation (like the age or hunger of the entity) are stored alongside the stats in the Animal class. As outlined in the table above

At least two predators should compete for the same food source.

Since we only have two predators and two prey the simulation works where predators can eat any prey they seek and the prey will go after any plant. Since this is a pseudo-real simulation that isn't realistically accurate, this seemed like a reasonable compromise for how the food chain would operate. However, the code does allow predators to eat only certain prey and pray to eat specific types of plants with very few alterations.

Some or all of the species should distinguish male and female individuals. For these, the creatures can only propagate when a male and female individual meet. Meet means they need to be within a specified distance to each other, for example in a neighbouring cell. You will need to experiment with the parameters for breeding probability to create a stable Population.

All animal entities are assigned random sex on creation, 50% either way and will only be able to propagate with the opposite sex. Likewise, only female entities can become pregnant and create offspring and the pregnancy cycle is calculated with a cool down for when they can next become pregnant.

You should keep track of the time of day. At least some creatures should exhibit different behaviour at some time of the day (for example: they may sleep at night and not move around during that time).

Entities can be defined as nocturnal or not and this will affect their movement depending on the day. For instance, if the time is night and the entity is nocturnal then the entity may start moving, which means that the entity will not move during the day and vice-versa for all the entities that are not considered to be nocturnal. Time is being displayed using a cool clock in the bottom right corner. Additionally, time could be seen at the very top in a digital format. (just in case the clock in the bottom right corner isn't correctly interpreted).

Challenge Tasks:

Simulate plants. Plants grow at a given rate, but they do not move. Some creatures eat plants. They will die if they do not find their food plant.

Plants have been simulated to grow at a steady rate. Which is affected greatly by the weather. For example, if the current weather doesn't have enough visibility then the plants will not have enough sunlight to perform photosynthesis and this may result in their death. Similarly, if the weather doesn't contain enough downfall, then the plants will die due to thirst. (the weather is talked about in the weather section). Each time plants grow they have the opportunity to "level up" which simulates the actual growth that plants go through. This also allows the animals who feed on plants to eat specific levels of the plant. For example, if the plant has a current level of 5 then the next time the plant is eaten it will not die straight away but rather lose a level. When the total level of the plant reaches 0 the plant would be considered dead and removed from the field.

Simulate weather. Weather can change, and it influences the behaviour of some simulated aspects. For example, grass may not grow without rain, or predators cannot see well in fog.

The weather has been simulated with the aid of seasons. There are four seasons in total, these being: Winter, Spring, Summer, and Autumn. Each season contains a maximum and minimum value for visibility and the downfall. When it is time for a change in season a random function is used to generate a value for both visibility and downfall that falls between the maximum and minimum values for visibility and downfall for that season. This simulates reality a bit more as the same season may contain some variety to a certain extent. Weather affects the animals and the disease. For example, plants can't grow without enough sunlight or rain and the disease can't spread if there is a lot of rain.

The GUI representation of the weather, what is below the weather is the GUI representation of the clock, as seen a minecraft clock has been used.

Simulate disease. Some animals are occasionally infected. Infection can spread to other animals when they meet.

The disease must start with a host, this is because, in reality, all diseases have a "main" host. The disease will be able to only infect species that are the same as the original host, however, if the disease "contacts" other species over and over again it will eventually be able to "mutate" and be able to infect other species as well. (In this context contacts refers to if the disease is within appropriate proximity of other kinds of species). The disease is affected by the weather. For example, if there is too much rain the disease will have less of a chance to mutate and infect other kinds of species.

Others: (Our own challenges)

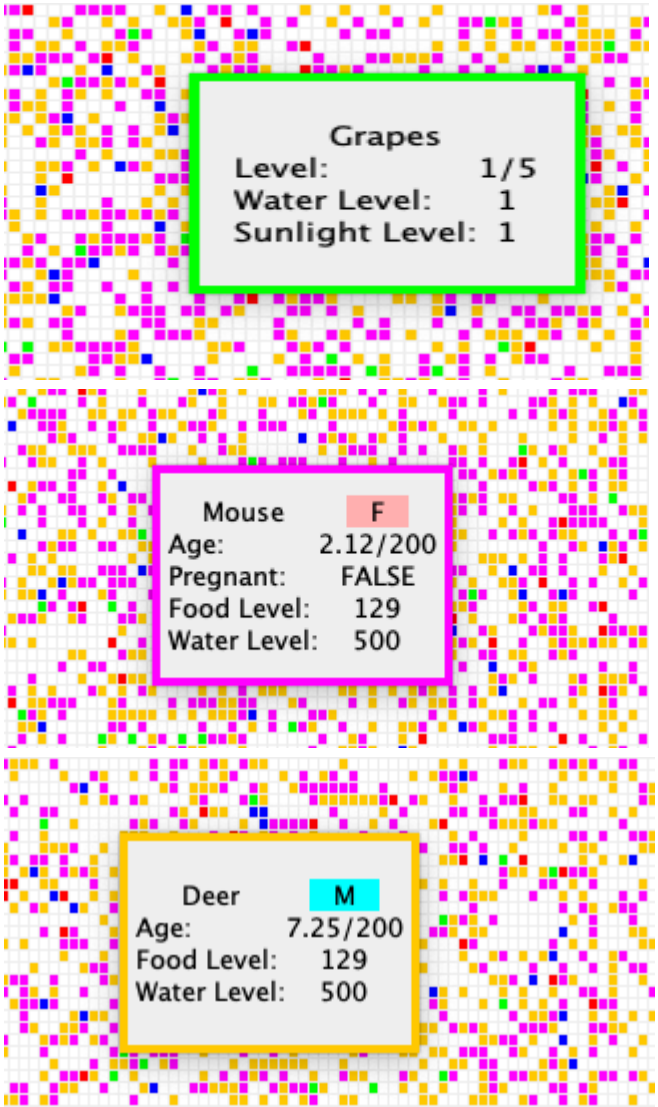
Buttons to control simulation rates.

On the top right of the screen, there are 4 buttons that can affect the rate at which the simulation runs, as well as reset it entirely. From left to right there is a play/pause button that will start and stop the simulation on press. The next button is a run speed incrementer when clicked the simulation will cycle through preset speeds at which the simulation will run. The next button is a single step button, the button will step through simulation once. The last button is a restart button, it will restart the simulation from the beginning at the speed which has been set.

Live display of the statistics of the each animal

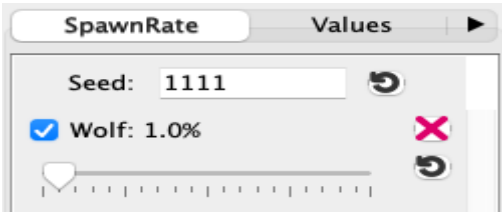
There is a live display of the statistics per animal, that is different to the number of each animal in total, by hovering over the animals on the simulation a box should appear that displays statistics unique to the animal, this includes the age of that animal, the sex, the food and water levels, and whether the animal is pregnant. The display should also show the colour of the animal in a neat way.

An example of what should be displayed when hovering over an animal:



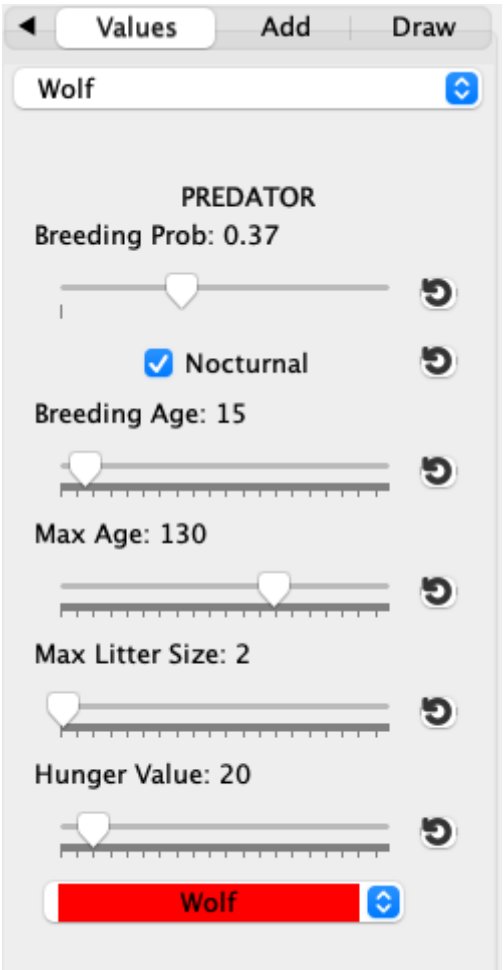
Manipulate starting state probability of the simulation.

The first tab of the options panel, "SpawnRate", controls the chances that an entity will be assigned to a square on simulation restart. The values are measured in percentage where it's the percentage chance that an entity will be assigned to a square. If all the entity percentages are below 100%, if the random chance happens to land on none of them, then the spot will be left empty (Ideal scenario). However if the user sets all the probabilities to equal 100% or greater, then the chances of the assignment will be selected among each entity using their probability as which has a better chance be compared to another, but no grid will be able to have a free spot on first creation, this isn't recommended to the user but the option is available. The spawn rates tab also lets the user temporarily disable an entity from spawning, or delete them altogether. But if they wish to get all the original entities back they can click the full reset button located at the bottom right.



Live change of the attributes of an entity.

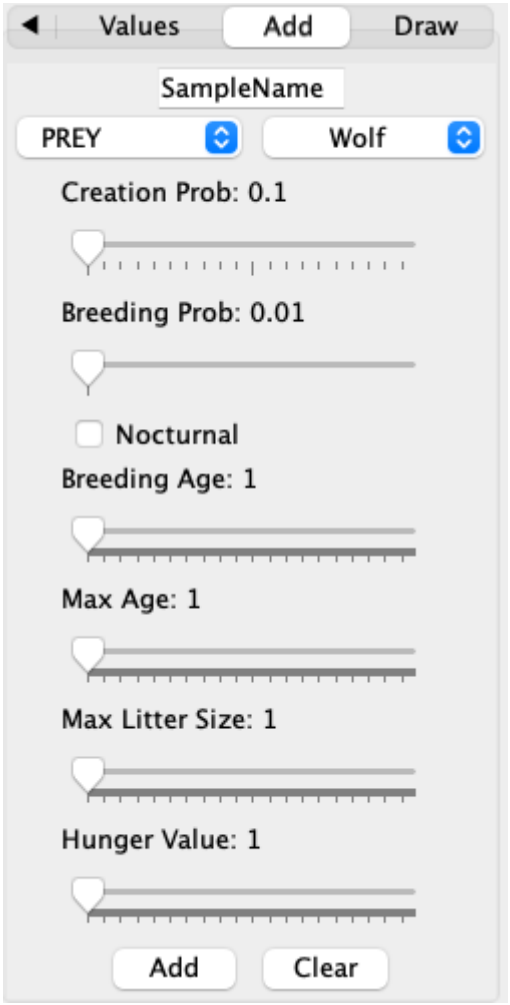
The second tab of the options panel, "Values", allows the user to select a currently existing entity and adjust its constants of it. Those values are like the max-age, the breeding rate, the litter size, colour, etc. Also, different attributes if the entity is a plant. These can also be changed while the simulator is running if the user would like to test.



Add custom entities to the simulation.

The third tab of the options panel, "Add", lets the user add whatever custom entity they want. They can assign each custom value, name, colour, type, etc and add it to the simulation. The entity will be added on the simulations restart (the 4th button along on the top right).

This is what the draw tab should look like:

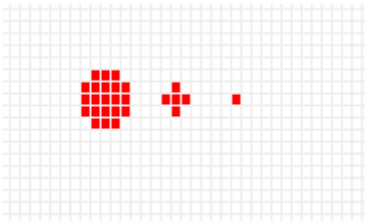


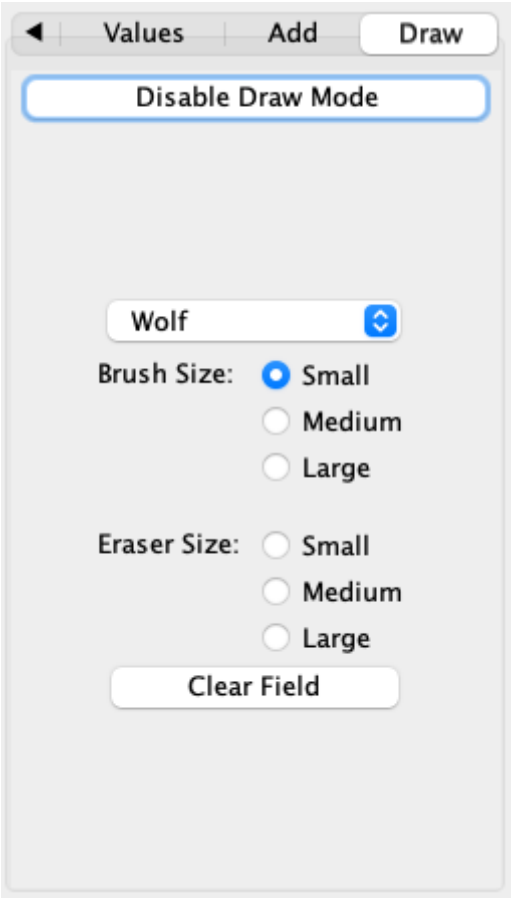
Draw and remove entities wherever you want on the grid.

The fourth tab of the options panel, "Draw", lets the user draw any entity they want onto the field. This can be done while paused or during simulation if they so wish. This panel lets you change the size of the brush you want to draw with, as well as select an eraser if you want to clear entities from the grid rather than draw them. You can also clear the field entirely if you want to fully draw your entire population.

What it looks like to add animals using the draw mode:

These pictures were taken after the clear field button was pressed for a better representation.





Code quality considerations:

Coupling:

The code was written with the objective of having as low coupling as possible so that changes in the classes don't impact other classes. This can be seen throughout the code, for example even though the Animals get infected with the disease, the disease isn't concerned about what an animal is or how they behave, in other words, the disease doesn't care about the mechanisms of the animals. Additionally, even though animals are placed in a field and they "move" throughout the field, the field isn't concerned with what it is holding, all the field does is allow the animals to move and change locations without a real understanding of which animals are which and whether the animals die or not.

Cohesion:

All methods throughout the entire program were designed in a way to ensure that they only do one thing and one thing only. The main example of this is the findMate method which allows the animal to find a suitable mate, in addition to the fine food method which allows the animal to find food. Both of these methods only do one thing and one thing only. An additional method would be the infect method in disease which allows the disease to infect other species. In the case of the method needing to do two things, this has always been split into two methods. As stated previously this spans across the entire code and isn't exclusive to the methods mentioned above.

Responsibility-driven design:
The philosophy of "A class that is responsible for holding some information should also be responsible for representing this information" has been adopted throughout the entire development process of this project, this can be seen with all the different toString method overrides in addition to all the different accessor and mutator methods.

Maintainability:
Despite having easy to read documentation, in addition to easy to read and understand variable and method names throughout the development of this project a core focus was how maintainable the simulation should be. The code as a result was highly generic that would allow for the development of any animal, which by extension allowed for the development of the draw tab aka tab4 and the add tab aka tab3