# KING'S COLLEGE LONDON

## 4CCS1PPA PROGRAMMING PRACTICE AND APPLICATIONS

### Third "Simulation" Coursework (Feb 2025)

Project Name: The Simulation


Student Name: Mehmet Kutay Bozkurt
Student ID: 23162628

Student Name: Anas Ahmed
Student ID: 23171444

# 1 Introduction

This simulation project integrates multiple components to create a dynamic ecosystem that a researcher can use to study predator-prey-plant interactions, with the ability to add or remove any species or environmental factors easily by editing the given JSON file. Infact, almost every aspect of the simulation can be controlled in the JSON file. The simulation is designed to simulate without any grid-based restrictions, allowing entities to move freely in a continuous space. Specifically, the coordinates of the entities are stored as doubles from zero to the width and height of the field. The entities also now have their own genetics, which are inherited from parent(s) and may mutate, all of which are initialised in the JSON file.

The simulation smoothly runs at 60 frames per second by utilising a `QuadTree` to store entities, which allows the entity searching and collision detection to be highly optimised. In each simulation "step," every entity is updated by calling its `update()` method, which handles movement, reproduction, and hunger. Entities make decisions based on other entities in their vicinity (that is, entities that are located inside their `sight` radius) and the current state of the environment (weather and time of day). Additionally, there is a method to handle overcrowding in the simulation, for limiting the growth of entities from being unnaturally rapid.

# 2 Tasks Lists and Implementation Details

## 2.1 Base Tasks

**Diverse Species:** With how the simulation is implemented, adding new species is as easy as adding the species' behavioural data into a JSON file. Each entity species can be of type `Prey`, `Predator`, or `Plant`, and the data is added accordingly. For example, for the predator `Fox`, the following definitions are used:

```
{
  "name": "Fox",
  "multiplyingRate": [0.05, 0.15],
  "maxLitterSize": [1, 4],
  "maxAge": [80, 120],
  "matureAge": [40, 40],
  "mutationRate": [0.01, 0.05],
  "maxSpeed": [4, 6.5],
  "sight": [30, 50],
  "numberOfEntitiesAtStart": 12,
  "eats": ["Rabbit"],
  "size": [3, 6],
  "colour": [230, 20, 40],
  "overcrowdingThreshold": [8, 25],
  "overcrowdingRadius": [10, 15],
  "maxOffspringSpawnDistance": [3, 5]
}
```

The values that are arrays that contain two values (such as `sight`, `size`, or `maxSpeed`) represent the minimum and the maximum values that the entity can have for that genetic trait. The values that are not arrays are the fixed values for that trait. In addition, these values can mutate when the entity breeds/multiplies. `numberOfEntitiesAtStart` and `eats` are fixed values that are not subject to mutation, as they are not genetic traits and they defined what the entity is in the context of the simulation. Finally, we have the following entities in the simulation:

- `Rabbit` — Prey, eats grass.
- `Squirrel` — Prey, eats grass.
- `Wolf` — Predator, eats rabbit and squirrel.
- `Fox` — Predator, eats rabbit.
- `Bear` — Predator, eats wolf and fox.
- `Grass` — Plant.

**Two Predators Competes for the Same Food Source:** Since it is easy to add multiple species of predators, currently there are two predators, `Wolf` and `Fox`, that compete for the same food source, `Rabbit`, as well as a `Bear` that eats both wolves and foxes.

**Distinguishing Gender:** Each entity has their own gender, represented in their genetics, which affects reproduction mechanics. Only entities of opposite genders can reproduce. Specifically, gender is implemented as an Enum with two values, `MALE` and `FEMALE`.

**Tracking Time of Day:** An `Environment` class is used to track the time of day and the weather, which governs how the cycle impacts entity behaviour. During the night, entities will not move unless they are hungry or there is a predator nearby. Additionally, when sleeping, food consumption is reduced.

### 2.2 Challenge Tasks

**Adding Plants:** Plants have been added, featuring growth and reproduction dynamics. Plants die when they detect too many plants of the same species nearby (as determined by their overcrowding genetics: `overcrowdingThreshold` and `overcrowdingRadius`), which results in natural looking patches of grass.

**Adding Weather:** As mentioned, weather is added under the `Environment` class, wherein weather conditions influence behavior and visibility, increasing the realism in the simulation environment. There are 4 weather conditions:

- `Clear` - No effect on entities.
- `Raining` - Plants grow faster (by a defined factor in the plant genetics).
- `Windy` - Pushes entities in the wind direction, even when they are sleeping. Wind direction visualized for the ease of the user.
- `Stormy` - Slows down entities by some factor and has the effect of windy. Different to windy condition, stormy condition is more severe, in the sense that the wind changes directions much more rapidly.

**Genetics system:** As one of our self-admitted challenges, we wanted to implement a genetics system for our entities. As mentioned earlier in the first base task, when reproducing, animals combine their parents genetics to form their own, with a chance to mutate certain attributes. Plants reproduce asexually, so they inherit their parent genetics, but these can also mutate.

**JSON configuration file:** Almost every single aspect of the simulation is controlled from this file.

- `foodValueForAnimals` - Controls food value when entity eats an animal
- `foodValueForPlants` - Controls food value when entity eats a plant
- `animalHungerDrain` - Rate of hunger drain
- `animalBreedingCost` - Food required to breed (note food is a value in the range 0 to 1)
- `mutationFactor` - How drastic mutations are
- `entityAgeRate` - How fast entities age
- `fieldScaleFactor` - The size of the field, smaller value = more zoomed in.
- `doDayNightCycleSpeed` - How fast the day goes
- `weatherChangeProbability` - The probability of the weather changing each day
- `windStrength` - How strong the wind pushes entities
- `stormMovementSpeedFactor` - How much to slow entities during a storm
- `animalHungerThreshold` - The level of hunger when an animal is ”hungry”
- `animalDyingOfHungerThreshold` - The level when an animal is ”dying of hunger”

## 3 Code Quality Considerations

### 3.1 Coupling

### 3.2 Cohesion

### 3.3 Responsibility-Driven Design

### 3.4 Maintainability

## 4 Final Remarks