AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE
WYDZIAŁ FIZYKI I INFORMATYKI STOSOWANEJ

KATEDRA ODDZIAŁYWAŃ I DETEKCJI CZĄSTEK

Projekt dyplomowy

Ecosystem simulation under Unity game engine
Symulacja ekosystemu przy użyciu silnika Unity

Autor:	Konrad Jakub Walas
Kierunek studiów:	Informatyka Stosowana
Opiekun pracy:	dr inż. Bartłomiej Rachwał

Kraków, 2022

Contents

Chapter 1

Introduction

1.1 Purpose

The Purpose of this engineering project was to develop a system and tool to easily simulate the ecosystem and use it to conduct analysis of various scenarios.

Ecosystem consists of agents such as herbivorous or predatory species and elements like water or food objects. AI is relying on state machines and Machine Learning and agents are making decisions based on their needs and the surrounding environment. Agents also have a specific set of features that is inherited by offspring from their parents.

The analysis covers the state of the ecosystem after a specified period of time, in particular the population and the distribution of the features of individuals participating in the simulation.

1.2 Structure of the thesis

Thesis consists of five main parts, that span over the next chapters:

Part I : Chapters 2, 3

Concepts and theory that is related to the scope of this project.

Part II : Chapters 4, 5, 6, 7

General theoretical description of the ecosystem simulation system. Definition of models and relationships between them.

Part III : Chapters 8, 9

Technical side of project. Model preparing and implementation.

Part IV : Chapters 10

Main project part - run and analysis of the ecosystem simulation.

Part V : Chapters 11, 12

Conclusions and further development concepts.

Chapter 2

Theoretical background

2.1 Principles of ecosystem simulation

2.1.1 Ecosystem

In general we can consider a *System* as any phenomenon that has at least two separable components working together through some interaction. The system can be either structural or functional. Systems can consist of smaller systems, each component being system on their own thus any particular system is part of a hierarchy of other systems. Therefore there is need to define scope and limits in order to not miss any important environmental relationships but on the other hand not to be overwhelmed with irrelevant information.

Ecological systems are larger systems of nature. That systems can begin with individual organism and expand toward greater complexity such as *communities* or *populations*. Communities, together with non-organic elements form *ecosystems*. The definition is rather arbitrary, and there is no specific way of defining that system. In practice the boundaries are set by investigator, and all interactions and structures within that boundaries form an *ecosystem*. Generally it contains some species like trees or dogs and components of landscape like terrain, lakes or cities. Very large continental-sized ecosystems with strong biotic continuity are called *biomes*.

Ecosystems are controlled by external and internal factors, such as climate, topography material or cyclic life and competition between species present in that system. [3]

2.1.2 Modeling Ecosystem

Model is abstract representation or simplification of an system that is used for studying and simulations. Models are simpler than real systems and have just the most important functional attributes of the real system. Modeling can be used to better understand real ecosystems and aid the conceptualization and measurement of real complex systems. Another use is to predict the consequences of an action that would be expensive, devastating, or difficult to accomplish in the real ecosystem, or just to see "what if". Complexity of nature is often overwhelming, therefore modeling is needed to fully understand it, however models must be checked frequently against real world to ensure that their representation is accurate, at least in areas that we are concerned about.

Two major types of ecological models are *analytic models* and *simulation models*. Both approaches are intended to increase our understanding and prediction of ecosystems and their components, but in practice the two methods are used for completely different questions. With the analytical model we have more calculations and advanced mathematics, while with the simulation approach we use simpler mathematics, heuristic solutions and greater use of computers.

Analytic models are rather simple, often linear systems, that can be accurately described by a set of mathematical equations whose behavior is well-known. On the other hand, simulation models use numerical computations to solve problems that are impossible or impractical to solve with analytic approach. Simulation models are more widely used, and are generally considered more ecologically realistic, while analytic models are great for their explanatory power and conciseness.

2.1.3 Lotka-Volterra equations

Lotka-Volterra equations are also known as the predator-prey equations. They are a pair of first-order nonlinear differential equations. They are used to describe the dynamics of biological or ecological systems with two interacting species, one as predator and one as prey. The equations contains populations change through time and growth rates of the two populations.

$$\frac{dx}{dt} = \alpha x - \beta xy, \tag{2.1}$$

$$\frac{dy}{dt} = \delta xy - \gamma y, \tag{2.2}$$

where

- x is the number of prey;
- y is the number of predators;
- $\frac{dy}{dt}$ and $\frac{dx}{dt}$ represent the instantaneous change rates of the two populations;
- t represents time;
- $\alpha, \beta, \gamma, \delta$ are positive real parameters describing the interaction of the two species

Lotka-Volterra model limitations

As this model is analytical one, it has its limitations: [5]

1. The prey population is able to find food at all times.
2. Prey is the only food source for predators.
3. The rate of change of population is proportional to its size.
4. During the process, the environment does not change in favour of one species, and genetic adaptation is inconsequential.
5. Predators and are always willing to eat prey.

Prey equation explanation

Term αx represents growth in population. Growth is exponential, as prey has unlimited food supply. Term βxy on the other hand is the rate of predation upon prey. It is proportional to the rate at which they meet, and sizes of both populations, meaning that if either x or y is zero, then there will be no predation.

Interpretation of whole equation: the rate of change in prey's population is equal to its own growth rate minus rate at which it is prayed upon.

Predator equation explanation

Term δxy represents growth in population, as it is rate of predation (different constant is used, because population growth is not is not necessarily equal to the rate at which prey is eaten). Term γy corresponds to the loss rate of the predators, due to emigration or natural death of individuals. This leads to exponential decay when there is no pray to supply predators growth.

Interpretation of whole equation: the rate of change in predator's population is determined by rate it consumes prey minus death rate.

Solutions to the equations

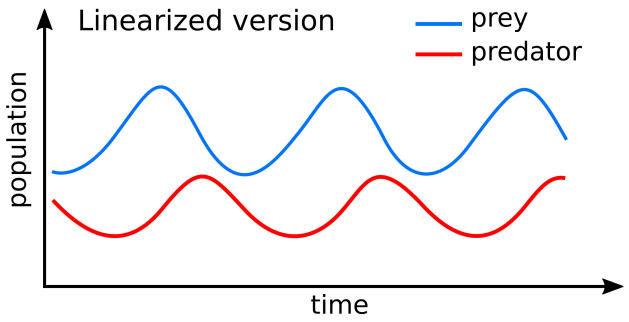

Figure 2.1: Draft of solutions to Lotka-Volterra equations [1]

Equations have periodic solutions with the population of predators trailing that of prey by $\pi/2$ in the cycle.

There are generalizations of the Lotka-Volterra equations, such as equations for n species or equations that include competition between those species [2].

2.2 Machine learning

Machine Learning is study of computer algorithms that are able to modify or adapt their actions through experience and use of data, so that these actions become more accurate. Machine learning is part of the broader topic of Artificial Intelligence.

When talking about machine learning it is good to define what learning actually is. The definition of *learning* used in this thesis is getting better at some task through practice. This leads to question: how computer knows if it is getting better at some

task. Answering that question provide a way to classify different algorithms int three main types:

1. Supervised learning
2. Unsupervised learning
3. Reinforcement learning [6]

Training using reinforcement learning was used in this thesis, so the focus is only on its description.

2.2.1 Reinforcement Learning

The goal of reinforcement learning is to learn a *policy*. Policy is a mapping from *observations* to *actions*. An observation is what the agent can measure from its environment (all its sensory inputs) and an action is change to the configuration of agent. In other words agent makes decision based on inputs it receives (observations) and does something (actions).

The main element of reinforcement learning is the reward signal. Rewards allow the algorithm to train its policies, as they give feedback on whether the actions taken by the agent were appropriate or not.

Training

Training this model involves taking some initially random action and then, based on the feedback from the reward, adjusting the probability of new actions so that they better fit the goal and receive a larger reward.

Figure 2.2: The reinforcement learning cycle. [14]

Curriculum learning

Curriculum learning is a way of training a machine learning model where more difficult aspects of a problem are gradually introduced in such a way that the model is always optimally challenged.

2.2.2 Observations and Sensors

For an agent to learn, observations should contain all the information the agent needs to perform its task. Without sufficient and relevant information, the agent may learn poorly or may not learn at all. A reasonable approach to determining what information should be included is to consider what would be needed to compute an analytical solution to the problem, or what a human might be expected to be able to use to solve the problem.

There can be many types of sensors, but the focus here is on two: *vector sensor* and *grid sensor*. A vector sensor is simply a vector of real values that are sent to the policy. The sensor uses a set of box queries in a grid shape and gives a top-down 2D view around the agent. During observations, the sensor detects the presence of detectable objects in each cell and encode that into one-hot representation. The collected information from each cell forms a 3D tensor observation and will be fed into the convolutional neural network of the agent policy (CNNs are a class of artificial neural networks that are most commonly used to analyze visual images). The observation for each grid cell is a one-hot encoding of the detected object. The total size of the created observations is $x_g \times y_g \times n$, where x_g and y_g are size of grid in each dimension and n is number of elements that the agent can recognize. [9]

One-hot encoding

In one-hot encoding, a separate bit of state is used for each state. It is called one-hot because only one bit is "hot" or TRUE at any time. [4]

2.2.3 The machine learning process

There are some common steps for solving given problem using machine learning.

1. Data Collection and Preparation
2. Feature Selection
3. Algorithm Choice
4. Parameter and Model Selection
5. Training
6. Evaluation [6]

2.3 Genetic algorithms

Genetic algorithms are metaheuristics inspired by process of natural selection. They are a part of Evolutionary Computation which is one of the three pillars of Computational Intelligence (CI). The CI is the theory, design, application and development of biologically and linguistically motivated computational paradigms. [15] Two other parts are Neural Networks and Fuzzy Systems. CI, and therefore genetic algorithms are used to develop systems, including games, in which agents behave intelligently.

There is no clear definition of genetic algorithms, but most methods that called GAs have following gears in common:

- Population of chromosomes
- Selection according to fitness
- Crossover to produce new offspring
- Random mutation of new offspring

2.3.1 Principles of Genetic Algorithms

In a genetic algorithm, a population of candidate solutions is evolved toward better solutions (more information on what it means to find a better solution can be found later in this chapter). These solutions are called individuals, creatures or phenotypes. Each candidate has a set of properties, called *genotype* or *chromosomes*, that can be altered or mutated in order to evolve into better solution. There are many ways to represent a chromosome, but one of the most popular representations is a binary string of 0s and 1s. For problems involving numbers, these can be converted to their binary representation so that they can be used in this representation.

The algorithm usually starts with a population of randomly generated individuals. For each iteration, called a generation, fitness is calculated and inferior solutions are replaced by new ones from the best individuals. The new generation is used in the next iteration of the algorithm. In most cases, the algorithm terminates when the maximum number of generations is reached or when a satisfactory fitness level is reached.

2.3.2 Fitness function

The goal of GA used to solve optimization problems is to find a set of parameter values that maximize some specified, often multi-parameter function. This function is called *fitness function*. In other words, the result of this function indicates how suitable a certain set of parameters, or *chromosome*, is for solving a given problem.

2.3.3 Genetic Operators

There are three common types of operators: selection, crossover, and mutation.

1. ***Selection*** This operator selects chromosomes in the population for reproduction. Chromosomes that has higher fitness are more likeky to be selected to reproduce (possibly several times).

2. ***Crossover*** In crossover one or more off-springs are produced using the genetic material of the parents. There are several types of crossover operator such as *One Point Crossover*, *Multi Point Crossover* or *Uniform Crossover*. The simplest and one of the most commonly used is One Point Crossover, where one random crossover point is selected and the tails of the two parents are swapped to produce new offspring.

3. ***Mutation*** This operator randomly flips some of the bits in a chromosome. For example, the string 00000100 can be mutated at its second position, giving 01000100. Mutation can occur at each bit position in a string with some, usually very small probability like 0.001).

2.3.4 Simple genetic algorithm

Most GAs (given a clearly defined problem and a bit string representation for candidate solutions) follow the following algorithm [7]:

1. Start with a randomly generated population of n l-bit chromosomes (candidate solutions to a problem).
2. Calculate the fitness $f(x)$ of each chromosome x in the population.
3. Repeat the following steps until n offspring have been created:
   (a) Select a pair of parent chromosomes from the current population, the probability of selection being an increasing function of fitness. Selection is done "with replacement", meaning that the same chromosome can be selected more than once to become a parent.
   (b) With probability p_c (the "crossover probability" or "crossover rate"), cross over the pair at a randomly chosen point (chosen with uniform probability) to form two offspring. If no crossover takes place, form two offspring that are exact copies of their respective parents. (Note that here the crossover rate is defined to be the probability that two parents will cross over in a single point. There are also "multi-point crossover" versions of the GA in which the crossover rate for a pair of parents is the number of points at which a crossover takes place.)
   (c) Mutate the two offspring at each locus with probability p_m (the mutation probability or mutation rate), and place the resulting chromosomes in the new population.
   (d) If n is odd, one new population member can be discarded at random.
4. Replace the current population with the new population.
5. Go to step 2.

Chapter 3

The project namespace

3.1 Logical items

Ecosystem - consists of environment together with actors who can interact with themselves and with this environment and evolve over time.

(Ecosystem) Component - the building block of ecosystem. Everything in the ecosystem is a component of the ecosystem.

Environment - set of components that agent is able to perceive and interact with.

Interaction - happens between two agents or other simulation object, is initiated by agent.

Action - result of decision made by agent.

Agent - entity that is able to move, make decisions and interact with its environment.

Food - stationary object that can actor can interact with.

Environment element - stationary object that agent can collide with.

3.2 Technical items

Academy - Singleton object which controls timing, reset, and training/inference settings of the environment.

Action - The carrying-out of a decision on the part of an agent within the environment.

Agent - Unity Component which produces observations and takes actions in the environment. Agents actions are determined by decisions produced by a Policy.

Decision - The specification produced by a Policy for an action to be carried out given an observation.

Editor - The Unity Editor, which may include any pane (e.g. Hierarchy, Scene, Inspector).

Experience - Corresponds to a tuple of [Agent observations, actions, rewards] of a single Agent obtained after a Step.

External Coordinator - ML-Agents class responsible for communication with outside processes (in this case, the Python API).

Frame - An instance of rendering the main camera for the display. Corresponds to each Update call of the game engine.

Observation - Partial information describing the state of the environment available to a given agent. (e.g. Vector, Visual)

Policy - The decision making mechanism for producing decisions from observations, typically a neural network model.

Reward - Signal provided at every step used to indicate desirability of an agent's action within the current state of the environment.

State - The underlying properties of the environment (including all agents within it) at a given time.

Step - Corresponds to an atomic change of the engine that happens between Agent decisions.

Trainer - Python class which is responsible for training a given group of Agents. [8]

Chapter 4

Ecosystem definition

The current and the following chapters define the ecosystem model used in the rest of the thesis. This chapter focuses on passive side of model, what builds an ecosystem, and relationships between these components. chapter 5 provides a dynamic description of the model and what happens to the ecosystem when the passage of time is taken into account.

4.1 Agents

Agents are heart of the ecosystem, they are components that can do something, interact with other components and change state of ecosystem.

There are two types of agents - prey and predators. Predators hunt prey, which feeds on plants.

Each agent have two sets of parameters - features and needs. Features alters agent's abilities and efficiency and needs influence decision making.

Species

Only agents of the same species can breed with each other. There are two species present in ecosystem:

1. Rabbit

   The rabbit is a representative of the prey. Rabbits eat carrots, drink water, reproduce with other rabbits and avoid the foxes that hunt them. They usually have about 6 offspring in a litter, but that number can vary depending on parents.

2. Fox

   The fox is a predator. They do not eat plants, but hunt rabbits, which are their main and only source of food. Foxes reproduce more slowly than rabbits and their typical number of cubs is 4.

A more detailed description of agents, including their features, needs, and how they make decisions can be found in chapter 6

Figure 4.1: Graphical (GameObject) representation of agents used in Scene: fox, male rabbit and female rabbit.

4.2 Environment

4.2.1 Terrain

The entire simulation takes place on a flat plane representing grass. There are two types of terrain in the environment - water and land. Agents cannot navigate through water, but they can satisfy their thirst near it. On land, individuals can move normally.

4.2.2 Food

The environment contains food that is consumed by an herbivorous species. In this ecosystem there is only one species of plant - carrots. Carrots grow around certain points called *Food Generators*. They appear at random points around the generator up to a certain number of plants, so new plants will only appear after the old ones have been eaten.

The coordinates of a point (x_f, y_f) in the Cartesian coordinate system relative to the generator position $(0, 0)$ are randomized and are represented by the following equation:

$$x_f = r_f \cdot cos(\alpha) \tag{4.1}$$

$$y_f = r_f \cdot sin(\alpha) \tag{4.2}$$

where

- r_f is random value in range $[0, r]$, r is specific parameter defined in implementation
- α is random angle in range $[0, 2\pi]$

Carrots grow over time, up to a certain maximum size. This is represented by growing in size. The longer a carrot lives, the larger it becomes and provides more energy to the individual who eats it. Carrots, unlike water, are consumed after being eaten by the rabbit, reducing its size. If the size is reduced below the minimum size, the carrot is considered to have been eaten whole and disappears from the ecosystem.

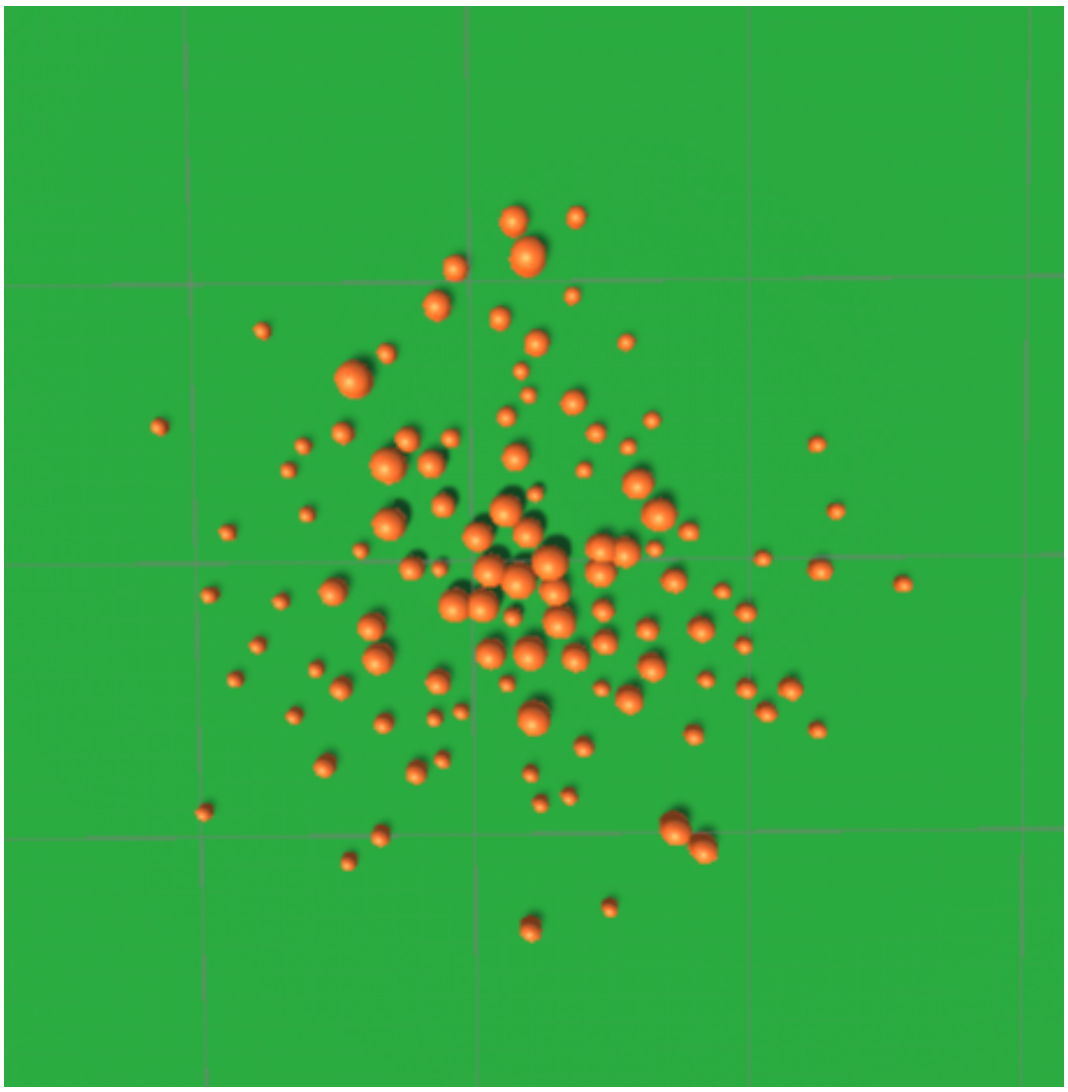

Figure 4.2: Carrots around the generator with big maximum number of plants.

4.2.3 Boundaries

The ecosystem has specific dimensions. It is a square, with impassable barriers along its sides, beyond which agents cannot enter. Every element is located inside the boundaries.

4.3 Interactions

Each interaction is initiated by an agent. There are two steps to initiate it. The agent must be ready to initiate the action, and then it must approach the object it wants to interact with. Interaction takes certain amount of time to complete and is different for each interaction, but interactions also can be instant.

Descriptions of each interaction

Below you will find definitions of interactions between agents. A more detailed specification of these interactions, e.g. their duration, is included in subsection 8.1.1. A visual representation of interactions between agents can be found in Figure 4.3.

Drinking Happens between agents and water. Agent spend time near water satisfying thirst and at the end of interaction it reduces thirst by an amount proportional to the time spent drinking.

Mating Happens between agents of the same species. Male agent need to find female partner that is willing to reproduce. If found partner do not have enough reproduction urge each interaction initiated by male agent will increase urge of its partner up to a point that mating can occur. New offspring are spawned at the end of interaction and new set of features is established through a genetic algorithm process. For more information, see chapter 7

Eating Carrot Happens between rabbit and carrot. Similar to drinking agent spend some time near carrot and then reduces hunger by eaten amount. Here however carrot are consumed after interaction.

Eating Rabbit Happens between fox and rabbit. Fox must find a rabbit and then eats it whole to reduce its hunger. Fox feeds one rabbit fully, reducing its hunger to zero.

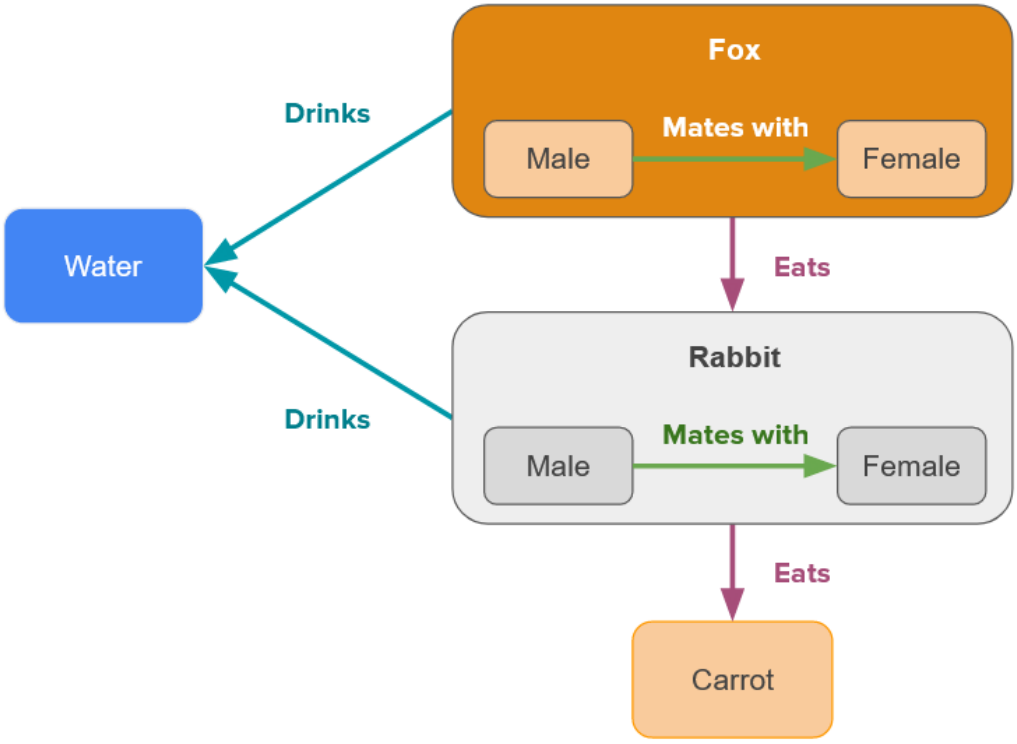

Figure 4.3: Diagram of interactions.

Chapter 5

Simulation

5.1 Simulation execution

The simulation involves the execution of actions by agents present in the ecosystem. They take actions based on current needs and analysis of their environment. Each agent has the ability to make decisions, move and interact with the environment. Each agent focuses its actions to satisfy its needs. Needs increase in time, so each agent must meet them periodically. Rate at which needs are growing also depends on features - higher values means a better adapted individual, but it comes at a price in terms of faster energy consumption.

5.2 Evolution of the simulation

The purpose of the simulation is to observe evolution. Evolution occurs in the process of reproduction of individuals. Actors pass on their features to their offspring in the process of reproduction, thus increasing the population and diversity of the species. What is evolved is the set of features of the individuals. The model of movement and decision-making is not changed in any way. Therefore, individuals differ between generations only in features. During the simulation, both prey and predators evolve, developing their traits in parallel and adapting to the environment in which they live.

5.3 Simulation metrics of interest

During simulation various metrics are being tracked. These information is saved to file with timestamps but also are available live during simulation. User can navigate through the environment and display the current values of those metrics. Metrics observed during simulation:

- Current population of rabbits
- Current population of foxes
- Median for each value of rabbits features
- Median for each value of foxes features

- Number of rabbits dead for the following reasons: starvation, thirst, and being eaten by a fox
- Number of foxes dead for the following reasons: starvation, and thirst

5.4 Advantages and disadvantages of the simulation model

As being said in section Modeling Ecosystem there are two ecological models that have different applications and restrictions. There are some advantages to using this model due to fact that it bypasses some of the limits from the analytical model described in Lotka-Volterra model limitations:

- Prey is not always able to find food, it is determined by position of agent in 3D world and agents can die of starvation.
- During this process, agents evolve, changing their features and adapting to the environment. It depends on the particular simulation whether this adaptation favours one species or whether they evolve in parallel, effectively mitigating the impact of these changes on other species.
- Predators are not always willing to eat prey. They have other needs such as thirst and if they are not hungry, they will not hunt for prey.

However simulation has its own restrictions:

- Simulation is non-deterministic. Due to the complexity of the models, interactions between them, and the randomness related to reproducing and inheritance, the state of the environment cannot be easily predicted. The outcome cannot be described by a function that depends only on time.
- Result does not necessarily have to be stable. Function of population over time will not always be periodical. It may happen that after several cycles one species suddenly goes extinct.

Chapter 6

The agent component details

This chapter is focused on a more detailed description of the agent. Agents are treated as a separate system with multiple components.

6.1 Description of agent's components

6.1.1 Features

Feature is an integer ranging from 0 to 100. Features are set at agent's birth and are later unchangeable. Features alter agents parameters and have an impact on its behaviour. There are 3 features for each actor:

Speed alters agent's movement speed.

$$s = (\sqrt[50]{2})^{v-50} \cdot a_s \tag{6.1}$$

where

- s is agents movement speed
- v is value of speed feature
- a_s is specific parameter defined in implementation

Sensory Range corresponds to agent ability to see other agents. For rabbits greater sensory range means that they can earlier begin to escape its predator.

$$r_s = \frac{r_{max} - r_{min}}{100} v + r_{min} \tag{6.2}$$

where

- r_s is agents sensory range
- r_{min} is minimal sensory range
- r_{max} is maximal sensory range
- v is value of sensory range feature

Fertility is responsible for controlling how fast agent regenerates after reproduction

$$m = \frac{2}{500}v + 0.8 \tag{6.3}$$

where

- m is modifier applied to growth rate of Reproduction urge of agent
- v is value of fertility feature

Fertility also regulates how many offspring a mother gives birth to.

$$n = \left\lfloor \frac{2d+1}{101}v \right\rfloor - d \tag{6.4}$$

where

- n is the number of offspring a mother gives birth to
- d is the maximum variation in the number of offspring by feature
- v is value of fertility feature

Genetic cost

The *genetic cost* parameter δ is related to features. The higher values are, the better agent is adapted to the environment, but it does involve increased energy consumption.

The genetic cost is simply the sum of all feature values.

$$\delta = \sum_{i=1}^{n} v_i \tag{6.5}$$

where

- v_i are value of i-th feature
- n is number of features, in this case $n = 3$

More useful, however, is the *relative genetic cost* δ_r, which is the ratio of the genetic cost to the sum of the maximum values of the traits.

$$\delta_r = \frac{\delta}{n \cdot v_{max}} \tag{6.6}$$

where

- v^i_{max} are maximal value of i-th feature, in this case $v_{max} = 100$
- n is number of features, in this case $n = 3$

6.1.2 Needs

Need is a real number between 0 and 100. Needs are constantly increasing during the simulation. There are needs that must be satisfied for an agent to live, such as hunger, and if this need reaches its maximum value, the agent dies. Needs do not change any of the agent's parameters, but they influence the agent's decisions in such a way that the agent strives to satisfy them. There are 3 needs for each actor:

Hunger obliges agent to look for food or hunt for prey. If an agent's hunger reaches 100% the agent dies.

Thirst compels agent to look for water source and drink. If an agent's thirst reaches 100% the agent dies.

Reproduction urge pushes agent to look for mate and extend the species.

Growth rate

Rate at which needs grow is influenced by relative genetic cost up to two times the normal growth and can be described by function:

$$f(t) = \alpha t(1 + \delta_r) \cdot m \tag{6.7}$$

where

- $f(t)$ is value of need at time t
- α is fixed need gain defined during simulation
- δ_r is relative genetic cost and $0 < \delta_r < 1$
- m is modifier that can be different for each need, but is equal to 1 in most cases

6.1.3 Body

Each agent has a body, which is responsible for the agent's physics. It has a collider and interacts with other objects in the game in a physical way.

6.1.4 State Machine

State Machine is component responsible for deciding what activity should agent perform at given time, and activate relevant policy based on the surrounding environment and agent's needs.

6.1.5 Machine Learning Agent

A *machine learning agent* is a subsystem of an agent that utilises machine learning. It is responsible for carrying out the activity determined by the state machine.

Sensors

The agent has two types of sensors:

1. **Vector sensor**

   A vector sensor of agent consists of two elements: x-coordinate and y-coordinate of the agent's velocity vector.

2. **Grid sensor**

   The agent has two grid sensors - one precise and the other long-range. The precise sensor is used by the agent to observe the agent's immediate surroundings and represents the agent's sense of sight, or hearing. The agent uses it to determine the exact location of environmental elements in its proximity. The long-range sensor, on the other hand, is responsible for the agent's general understanding of the environment and helps the agent determine the direction in which an environmental item is located without giving the agent its exact location. It represents the sense of smell, or again hearing.

   Detectable components: female fox, male fox, female rabbit, male rabbit, water, wall, food

Decision requester

Decision requester is component that request decision from policy automatically and periodically, at specified intervals. If decision is requested policy gathers input and maps it to actions. If the decision frequency is lower than the number of simulation frames, the last decision is repeated between frames.

6.2 Decision making process

The agent makes decisions based on observations of his environment, and his current needs. Observing the environment means, for example, spotting a predator, while deciding on the basis of current needs implies switching to food search mode when the agent is hungry.

There are several pre-trained models of agent behaviour. Each model corresponds to some specific activity that the agent performs, for example searching for food. The choice of a particular model depends on the state the agent is in and is managed by a state machine.

Because training such a general model would be demanding and difficult to achieve, several less complex machine learning models are used, instead of a single, complex one handling the whole agent behaviour. The agent adapts its behavioural model to the current situation.

Example

The rabbit is very hungry and a bit thirsty. So the food-seeking state is activated, and thus the policy responsible for this task is loaded. After a while, the rabbit has found and eaten a carrot, but in the meantime its thirst has increased even more, so the state machine will switch on the state of searching for water. However, in

the meantime, a fox has appeared next to the agent, so instead the state of running away from the predator will be activated. Until the rabbit either does not flee from the fox or is eaten this state will persist and the model responsible for fleeing will direct the agent's actions.

6.2.1 Switching between states

The states are selected on a ranking basis. The state machine constantly monitors all states in real time and selects the one with the highest rank. There are two types of states: *main state* and *special state*.

The main states have ranking functions that determine the rank for a given state returning a real value $y \in [0, 1]$. This function must be defined separately for each state. An example would be a linear function mapping the hunger value to the rank for the looking for food state.

Special states instead of a ranking function have an activation function returning the value $y \in \{0, 1\}$, and having a predetermined rank value from the range $[0, 2]$, which is multiplied by the value returned by the activation function.

6.2.2 States

Main States

1. **Chilling**

   Agents: all agents

   Description: This state represents agent's resting and is activated, when agent has nothing to do (all needs are met, there is no predator in sensory range).

   Ranking function:

   $$f(x) = a_c \tag{6.8}$$

   where a_c is specific parameter defined in implementation.

   ML Model: Basic, untrained model, which means that the agent makes small random movements, but stays in one place at the same time without standing completely still.

2. **Looking for food**

   Agents: rabbits

   Description: This state represents looking for carrots and is activated, when rabbit is hungry.

   Ranking function:

   $$f(x) = x \tag{6.9}$$

   where x is value of hunger.

   ML Model: Eathing Carrot Model

3. **Hunting**

   Agents: foxes

Description: This state represents looking for rabbits to eat and is activated, when fox is hungry.

Ranking function:

$$f(x) = x \tag{6.10}$$

where x is value of hunger.

ML Model: Hunting Model

4. **Looking for mate**

   Agents: all agents

   Description: This state represents looking for partner to reproduce and is activated, when agent has high reproduction urge.

   Ranking function:

$$f(x) = \frac{x}{100} a_m^{x-100} \tag{6.11}$$

   where x is value of reproduction urge and a_m is specific parameter defined in implementation.

   ML Model: Mating Model

5. **Looking for water**

   Agents: all agents

   Description: This state represents looking for water and is activated, when agent is thirsty.

   Ranking function:

$$f(x) = x \tag{6.12}$$

   where x is value of thirst.

   ML Model: Drinking Model

Special States

1. **Drinking**

   Agents: all agents

   Description: This state represents drinking and is active while agent is interacting with water.

   Activation function:

$$f(x) = \begin{cases} 0 & \text{if drinking interaction is not active} \\ 1 & \text{if drinking interaction is active} \end{cases} \tag{6.13}$$

   ML Model: None. Agent waits for drinking interaction to end.

2. **Eating**

   Agents: rabbits

   Description: This state represents eating and is active while agent is interacting with carrot.

Activation function:

$$f(x) = \begin{cases} 0 & \text{if eating carrot interaction is not active} \\ 1 & \text{if eating carrot interaction is active} \end{cases} \tag{6.14}$$

ML Model: None. Agent waits for eating carrot interaction to end.

3. **Escaping Predator**

   Agents: rabbit

   Description: This state represents running away from predator, when it is near.

   Activation function:

$$f(x) = \begin{cases} 0 & \text{if no predator is in sensory range} \\ 1 & \text{if predator is in sensory range} \end{cases} \tag{6.15}$$

   ML Model: Escaping Predator Model

6.2.3 Models

The agent uses machine learning to make decisions about specific activity. For each of these actions, there is a different model responsible for solving that task.

1. **Drinking**

   Description: Agent aims to find water as soon as possible and interact with it.

2. **Eating Carrot**

   Description: Agent aims to find carrot as soon as possible and interact with it.

3. **Mating**

   Description: Agent aims to find carrot as soon as possible and interact with it.

4. **Hunting**

   Description: Fox aims to find rabbit as soon as possible and interact with it.

5. **Escaping Predator**

   Description: Rabbit aims to escape predator and avoid being eaten.

6.2.4 Actions

Actions are results of agent decisions. Essentially actions are values returned by policy that tells agent how to behave. Actions can be discrete or continuous. In this simulation all agents have the same actions - three continuous actions and one discrete action.

Continuous actions

- value telling agent to move forward or backward,
- value telling agent to move right or left,
- value by which agent should rotate.

Each continuous action has value in range $[-1, 1]$.

Discrete actions

- boolean value to tell if agent want to initiate interaction.

Chapter 7

Genetic algorithms for evolution of agents

In the same way that a simulation model of an ecosystem differs from an analytical one, the different parts of the genetic algorithm used in this simulation will also differ slightly compared to standard genetic algorithms.

7.1 Selection

Firstly, there is no defined fitness function. It is not needed. During the simulation, the individuals match naturally. If an individual is less well adapted to its environment, there is a greater chance that it will be eaten or that it will not be able to satisfy its own needs and will die. The individuals who do mate, however, are those who have managed to overcome difficulties. They are able to pass on to their offspring the genes that have enabled them to get this far. Selection can thus be referred to as natural selection.

7.2 Chromosome representation

Naturally, a chromosome will correspond to the set of all traits an agent possesses and a single gene to a single trait.

7.3 Crossover

As the number of offspring is usually greater than 2, uniform crossover has been chosen as the crossover operator. During gene passing, it is randomly determined whether a child receives a feature from its father or its mother. The probabilities are each equal to 0.5.

7.4 Mutation

As chromosome contains integer values random resetting mutation method was chosen. For each gene (feature), there is a probability that it will be replaced by a new random integer value $v \in [0, 100]$.

Chapter 8

Implementation in Unity Engine

This chapter describes how the theoretical model described in the previous chapters has been implemented in practice, detailing the specific parameters defined previously.

In addition, due to its technical nature, the nomenclature from Unity engine [12] will be used, in particular:

GameObject - used to represent everything which can exists in scene. Every object in game is GameObject. GameObject contains components.

Prefab - The Prefab Asset acts as a template from which new Prefab instances (GameObjects) can be created in the Scene

Component - object that define properties and behaviour of GameObject it is attached to.

Project and the source code can be found in public repository on GitHub: `https://github.com/kerdamon/Engineering-Thesis-Simulating-Ecosystem`

8.1 Components

Below is the list of custom components written for the project:

1. State machine - handling state change described in subsection 6.2.1
2. Interactions - handling interactions described in section 4.3
3. Interaction managers - handling the initiation of the corresponding interaction when an agent wants to initiate it and the appropriate conditions are met
4. Agent component - handling training and inference of ML model
5. Simulation Control - gathering statistics during simulation
6. Training Area managers - controlling training environment, for example, provides the ability to place agents in a random position in a training environment
7. Food managing - components used to control plant growth

8. Needs - managing needs of agent

9. Features - managing features of agent

As the thesis does not focus on a strictly technical approach to the problem, it is not necessary to describe most of the components, except for the description interactions and values of parameters for features, needs and sates in order to make their functioning more precise.

8.1.1 Interactions

Below are specific values of parameters regulating the functioning of the interactions defined in section 4.3

Drinking Interaction

Max drinking time - after this time agent regenerates max thirst: 10 s

Step time - determines the minimum time to record drinking progress: 0.5s

Regeneration per step - amount by which thirst changes in each step: 5

Mating Interaction

Duration time : 5 s

Eating Carrot Interaction

Max eating time - after this time agent regenerates max hunger: 10 s

Step time - determines the minimum time to record eating progress: 1s

Regeneration per step - amount by which hunger changes in each step: 10

8.1.2 Features, needs and states

Below are the specific values of the parameters described in the definition of features, needs and states (subsections 6.1.1, 6.1.2 and 6.2.2).

Name	Symbol	Value
Speed coefficient	a_s	1
Minimal sensory range	r_{min}	15
Maximal sensory range	r_{max}	27
Offspring variation by feature	d	Rabbits: 3 Foxes: 2
Growth rate	α	0.5
Chilling coefficient	a_c	0.6
Mating coefficient	a_m	1.01

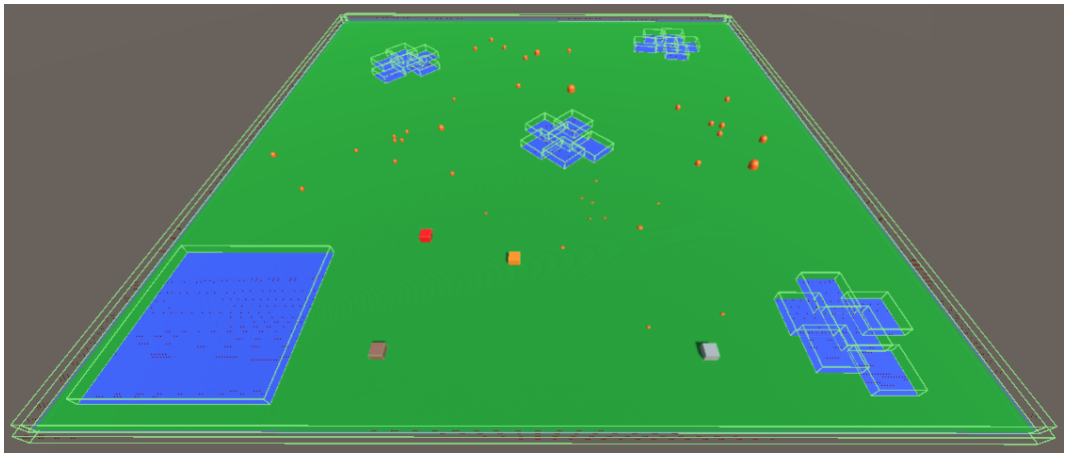

Figure 8.1: Simulation area in Unity editor.

8.2 Prefabs

Simulation and training Areas

Areas contain agents together with environment. Each area contains terrain - plane, boundaries, water, food generators, rabbits and predators.

Agents

An agent is just an implementation of the agent defined in chapter 6. The agent's body is a cube with side length 1. The agent also has a cubic collider which is used to initiate interactions with side length 1.5, so it does not need to directly touch the body of the object it wants to interact with.

Precision grid sensor has dimensions of 37 x 40 and a cell size of 1.5. Long-range grid sensor has dimensions of 20 x 20 and a cell size of 40. The sensor has such a large range because, due to technical limitations, the smallest possible size of this sensor is 20x20, and the cell size had to be chosen so as not to give the agent the exact position of the object, only its general location.

The rabbit can give birth to a base of 6 offspring, the discrepancy due to fecundity is 3, while randomness can modify the number of offspring by 2. Thus, the final number of offspring the rabbit can give birth to is 1-11. For the fox, these numbers are 4, 2, and 1, respectively, giving a final number of offspring of 1 - 7. The probability of mutation is 0.005.

Plant Generator and Carrots

The carrot is a capsule of size 1, although it is modified according to the growth level of the plant. Its minimum size is 0.5, maximum is 1.5, and during growth it changes by 0.1 every 5 seconds. For the Plant Generator, the spawn range is 8, the spawn radius is equal to 6, and the plant limit is 15, but these parameters may vary depending on the simulation.

Water

Water is just a plane with a collider placed above its surface to allow sensors to detect it and to prevent agents from moving through the water. The base size of the water element is 5x5, while the sea element is 20x20. The lake element consists of 6 merged water elements.

8.3 Scenes

The project was divided into 6 scenes. Five of them are training scenes and one is a simulation scene. The training scenes are slightly modified versions of the simulation scene for purposes of training ML models. Each training scene contains multiple training areas specific to each model in order to train it better and faster. The base size of the plane is 200x200.

Each scene contain main camera, lighting, event system (used for inputting keys) and area with environment and agents.

Simulation scene layout

Area present in this scene contains agents with with normal setup and behaviour with all states, models and interactions defined in chapter 4 and chapter 6 and further specified above in this chapter. Simulation scene also includes simulation controller GameObject responsible for measuring statistics during simulations and logging it, as well as canvas GameObject for displaying it on screen.

Training scene layout

The area present in this scene contains agents with modified states, interactions and simplified behaviours tailored to the specific model being trained. For example, in the mating training scene, the agents use a simplified mating interaction in which the male only needs to find a female and interact with her, after which the female is moved to a new random position and the male continues to search for a mate. There are also several copies of areas in the scenes to speed up the training of the model.

The individual training scenes are described in more detail in chapter 9

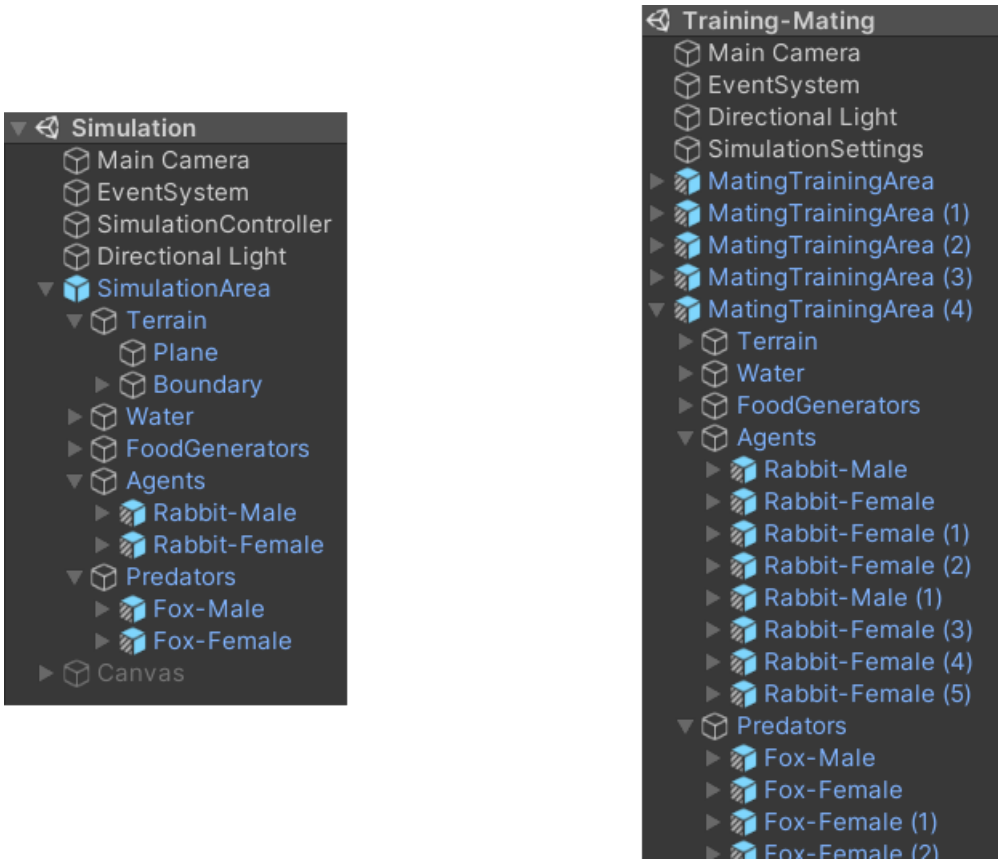

Figure 8.2: Simulation and training scene layouts.

Chapter 9

Training Models

Training was done in following order: Drinking, Eatting Carrot, Mating, Hunting, Escaping Predator. Subsequent training scenarios are more and more advanced and use models that have been trained before. In each case, the agents' features are randomised so that the trained model could cope with the changing features during the simulation. Each scenario also had specific simplified interactions to aid the training process. Additionally, in the scenarios, the actors receive negative rewards that increase over time, accumulating to specific values intended to encourage the agents to complete the task as quickly as possible and not to sit idle.

Curriculum learning

In order to train the models better and faster, a *curriculum learning* process was used, which means that the parameters of the scenario were changed during learning process. By default, each scenario consisted of 4 lessons changing the size of the training area and thus the density of the distribution of elements in it. The scale of the area is 0.5, 1, 2 and 3 in consecutive lessons respectively. Only the plane and borders are resized, not other elements.

Results and configuration

Results and configuration files can be found in Appendix A

9.1 Drinking

In this scenario, all agents trained the same drinking model. They aimed to find a source of water and interact with it.

Number of agents 4 - 1 male rabbit, 1 female rabbit, 1 male fox, 1 female fox

Number of training areas 7

Interaction When interacting with water the agent receives a reward, and is moved to a random location in the training environment.

Rewards +1 for interaction with water, -0.2 for bumping into wall or carrot, -1 by default in each episode.

Curriculum No additional changes other than the size of the environment.

9.2 Eating Carrot

In this scenario, the rabbits train to find food while the foxes wander around in the background, using a model already trained to search for water.

Number of agents 10 - 4 male rabbits, 4 female rabbits, 1 male fox, 1 female fox

Number of training areas 6

Interaction When an agent interacts with a carrot, the carrot disappears and the agent gets a reward.

Rewards +1 for interaction with water, -0.2 for bumping into wall or water, -1 by default in each episode.

Curriculum No additional changes other than the size of the environment.

9.3 Mating - rabbit

In this scenario, male rabbits train to find female rabbits and interact with them.

Number of agents 12 - 3 male rabbit, 9 female rabbit

Number of training areas 5

Interaction When the male rabbit interacts with the female rabbit, he gets a reward and the female is moved to a random location in the environment. Females are randomly distributed in the environment and are in a chilling state. Females are not involved in the training of the model, they are only part of the environment with which males can interact.

Rewards +1 for interaction with female, -0.01 for bumping into wall, carrot or water, -1 by default in each episode.

Curriculum Water was deactivated in the first two lessons and added in the third and fourth.

9.4 Mating - fox

In this scenario, male foxes train to find female foxes and interact with them.

Number of agents 12 - 3 male fox, 9 female fox

Number of training areas 5

Interaction When the male fox interacts with the male fox, he gets a reward and the female is moved to a random location in the environment. Females are randomly distributed in the environment and are in a chilling state. Females are not involved in the training of the model, they are only part of the environment with which males can interact.

Rewards +1 for interaction with female, -0.01 for bumping into wall, carrot or water, -1 by default in each episode.

Curriculum Water was deactivated in the first two lessons and added in the third and fourth.

9.5 Hunting

In this scenario, foxes train to find rabbits and eat them.

Number of agents 12 - 4 male rabbit, 4 female rabbit, 2 male fox, 2 female fox

Number of training areas 6

Interaction When the fox interacts with the rabbit, it gets a reward and the rabbit is moved to a random location in the environment. Rabbits are randomly distributed in the environment and are in a chilling state. Rabbits are not involved in the training of the model, they are only part of the environment with which foxes can interact.

Rewards +1 for interaction with rabbit, -0.01 for bumping into wall, carrot or water, -1 by default in each episode

Curriculum Water was deactivated in the first two lessons and added in the third and fourth. After some time, the previous size from the second lesson was reverted, due to the low average reward achieved by the agents in each episode.

9.6 Escaping Predator

In this scenario, rabbits train to escape foxes that are trying to eat them.

Number of agents 12 - 4 male rabbit, 4 female rabbit, 2 male fox, 2 female fox

Number of training areas 6

Interaction When the fox interacts with the rabbit, the rabbit receives a penalty and is moved to a random location in the environment.

Rewards -1 for the rabbit when the fox interacts with it, -0.01 for bumping into wall, carrot or water, +1 by default in each episode

Curriculum There was only one lesson in a small area

Chapter 10

Analysis of performed simulations

The simulations described in this chapter were run in a program prepared for the thesis and the resulting data are available in the project code repository [10].

10.1 Simulation 1

In this scenario there were only rabbits and plenty of food. The rabbits therefore had ideal conditions to thrive, without the threat of predators and with good availability of food and water.

10.1.1 Initial state of the environment

Number of agents : 4 (2 male rabbits, 2 female rabbits)

Number of food generators : 6

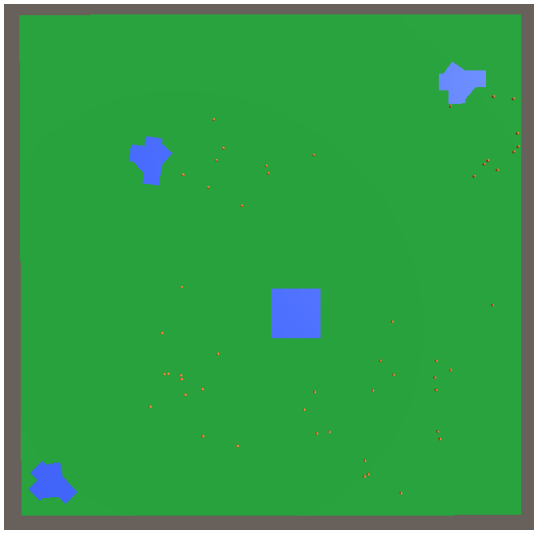

Figure 10.1: Distribution of environmental components for Simulation 1.

10.1.2 Results

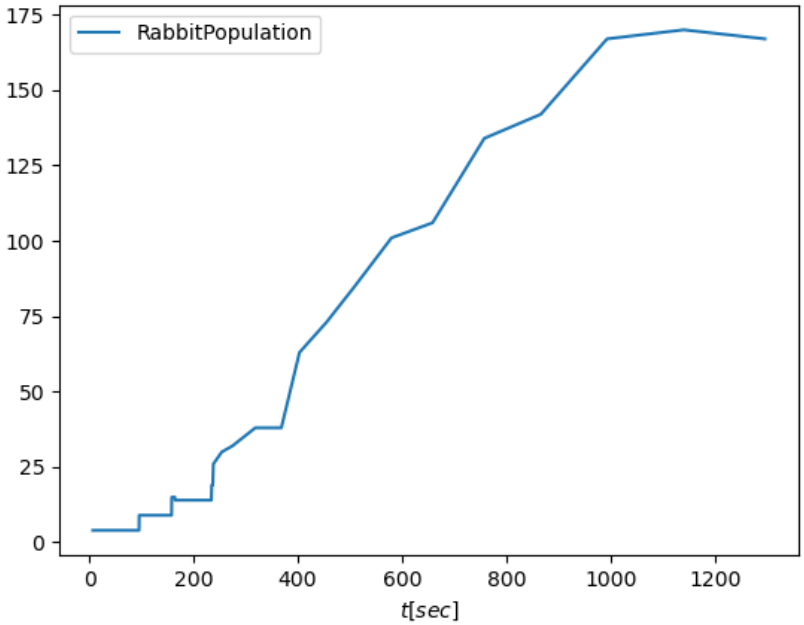

Figure 10.2: Population of rabbits over time.

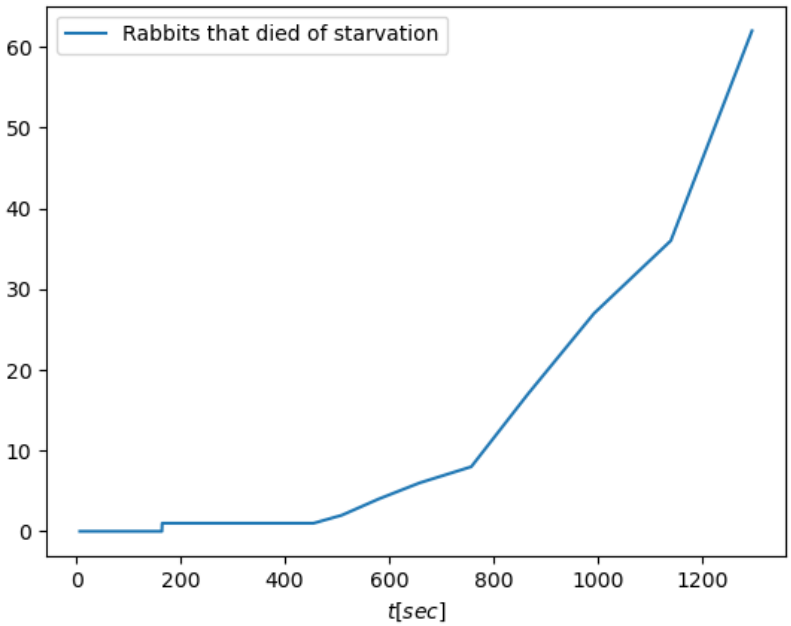

Figure 10.3: Number of rabbits that died of starvation over time.

10.1.3 Analysis

Population

As might be expected, in an environment ideal for rabbits to thrive, their numbers grew rapidly until the number of carrots in the environment was no longer sufficient for the rabbits to survive, then their numbers remained constant.

Features

The graphs of the features over time were not very extensive, their values did not change, so only their final values are worth analysing. The median features of the rabbits at the end of simulation were as follows:

- Speed: 74
- Sensory range: 85
- Fertility: 36

As we can see, the value of the fertility feature is not high, from which we can conclude that rapid reproduction was not crucial in this scenario. The speed was quite high, the faster the rabbits could get to the carrots the better chance they had that no other individual would eat it before him. It is interesting to note that the size of the sensor range is really large, despite the fact that in this case the rabbits did not derive any benefit from it because there were no foxes in the environment. The reason could be that there was not much generation of agents in such a one, old rabbits that had an unfavourable trait value in these conditions survived anyway, passing on their genes to their offspring.

10.2 Simulation 2

In this scenario, as in the previous one, there are no predators, but the amount of food is much lower, so that the rabbits will have to compete among themselves for resources in order to survive.

10.2.1 Initial state of the environment

Number of agents : 6 (3 male rabbits, 3 female rabbits)

Number of food generators : 1, but with changed parameters:

- spawn interval: 10, instead of 8
- spawning radius: 15, instead of 10
- plant limit 20: instead of 10

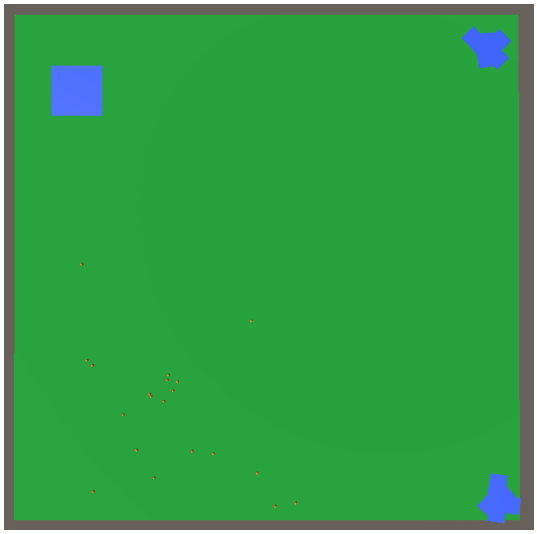

Figure 10.4: Distribution of environmental components for Simulation 2.

10.2.2 Results

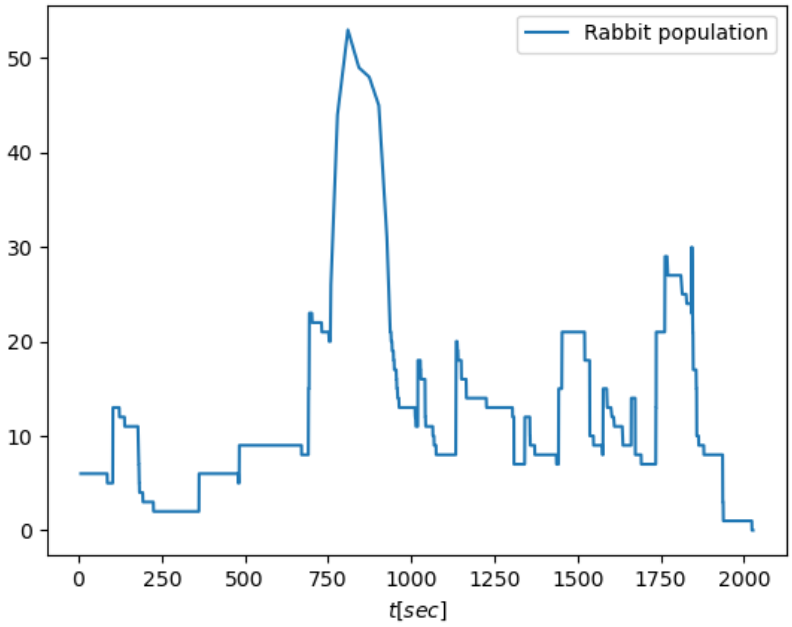

Figure 10.5: Population of rabbits over time.

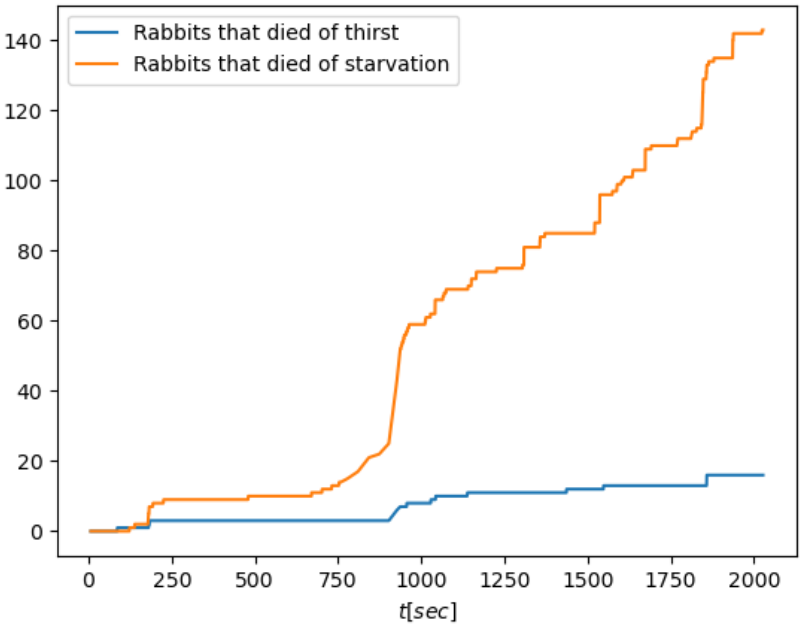

Figure 10.6: Death cause.

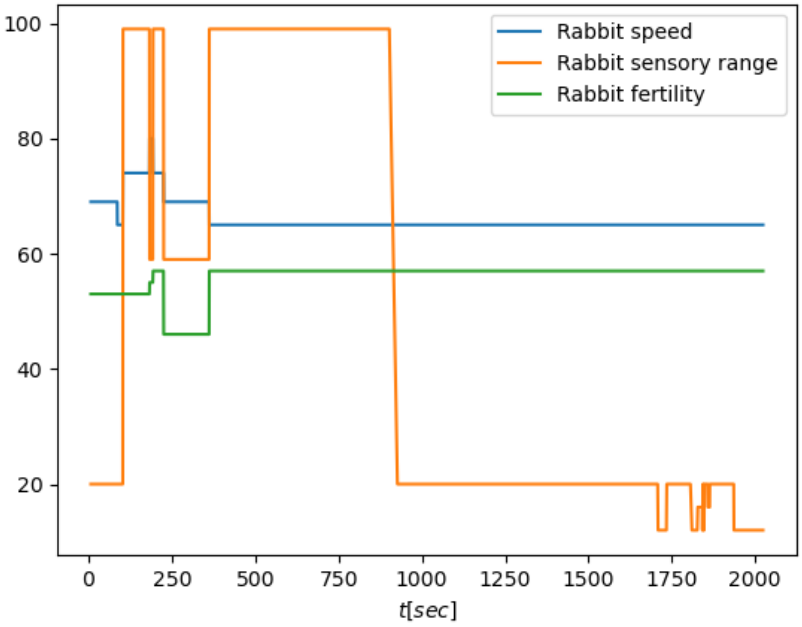

Figure 10.7: Median value of features over time.

10.2.3 Analysis

Population

In this scenario, initially the rabbits also reproduced rapidly, as can be seen with a maximum at around 800 seconds. At this point, however, the food ran out and the rabbits began to die out just as quickly, as can also be clearly seen in Figure 10.6. Thereafter, their population fluctuated erratically until there were enough rabbits to eat almost all the food again, leading to the extinction of the entire species.

Death causes

As might be expected, most deaths were due to starvation. It is possible to note a sharp rise around the 900th second, corresponding to the extinction of a large proportion of rabbits, after the supply of carrots was no longer sufficient for the growing rabbit population. Thereafter, the function of deceased rabbits over time was linear, corresponding to a period of fluctuation in the rabbit population until the whole species became extinct.

Features

Unlike the previous scenario, in this case the rotation of individuals was much higher, allowing for better evolution of traits. Their final values were as follows:

- Speed: 65
- Sensory range: 12
- Fertility: 57

The values for speed and fertility seem to be a quite reasonable compromise, reaching values sufficient for survival in the environment, but not too high to justify the high energy consumption that a high value for these traits brings. Whereas the sensory range value, as expected, is low, because in this scenario it does not bring any benefit to the rabbits, but only results in higher energy consumption.

10.3 Simulation 3

In this scenario, the amount of food is at a comfortable level for the rabbits, but there are predators in the environment that prey on them.

10.3.1 Initial state of the environment

Number of agents : 10 (4 male rabbits, 4 female rabbits, 1 male fox, 1 female fox)

Number of food generators : 5

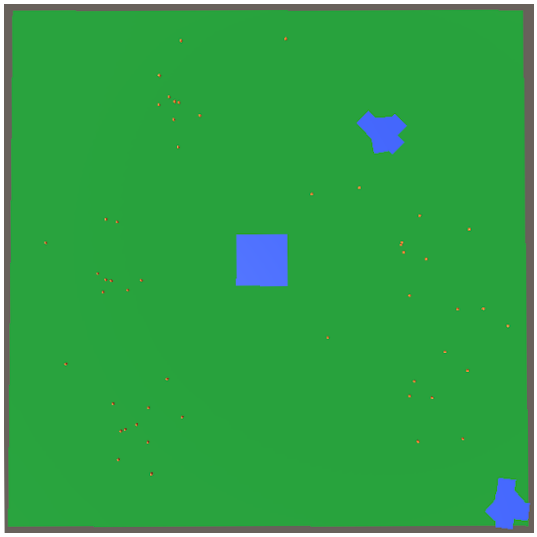

Figure 10.8: Distribution of environmental components for Simulation 3 and 4.

10.3.2 Results

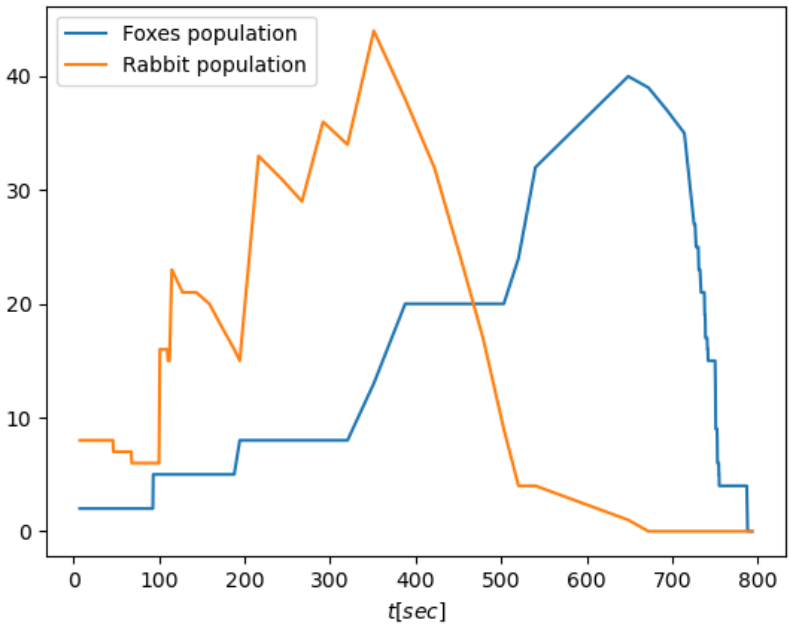

Figure 10.9: Populations of agents over time.

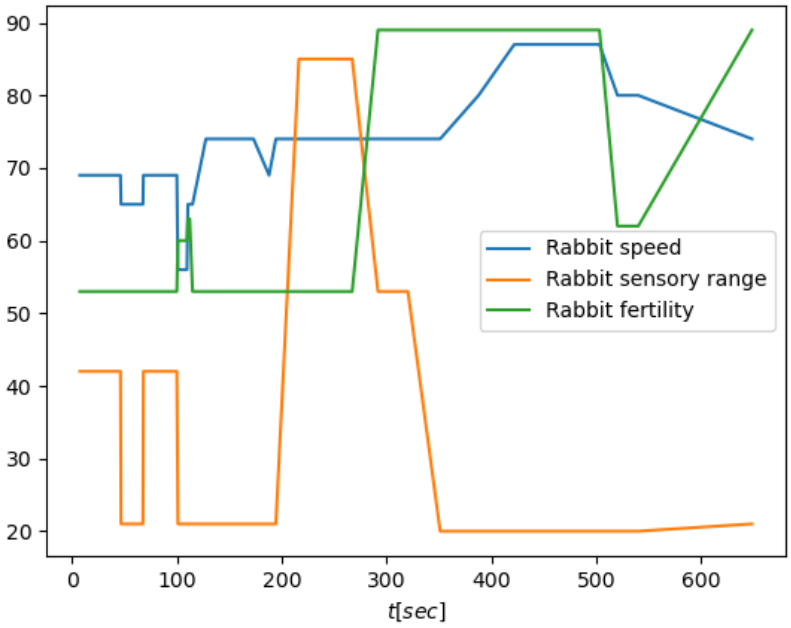

Figure 10.10: Median value of rabbit features over time.

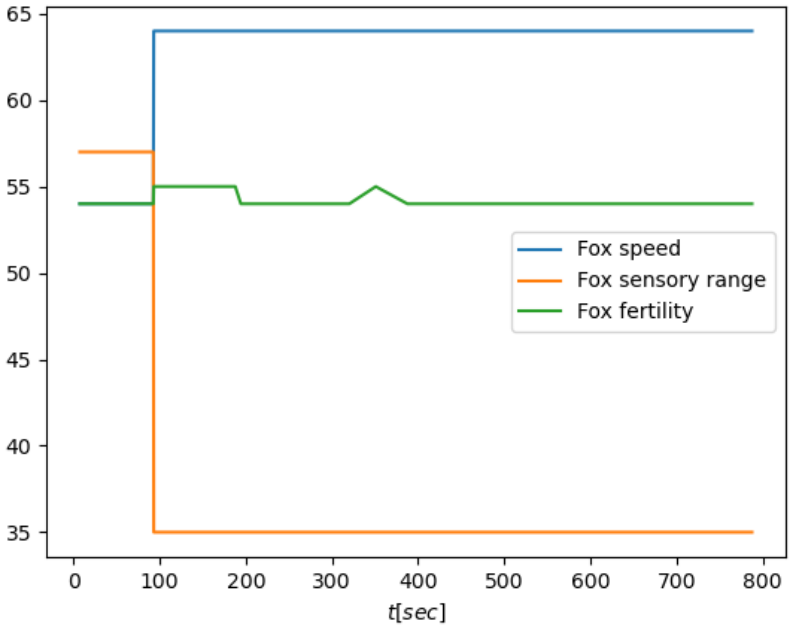

Figure 10.11: Median value of fox features over time.

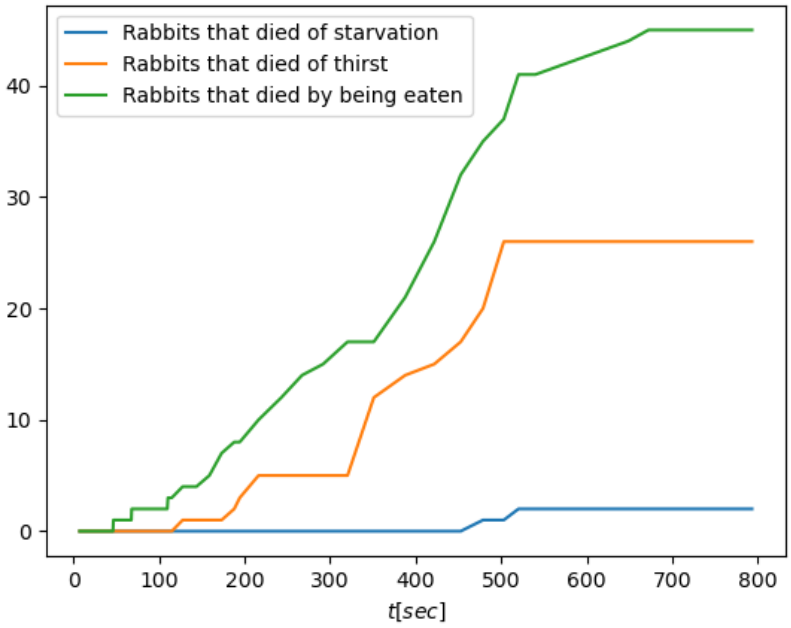

Figure 10.12: Rabbits death cause.

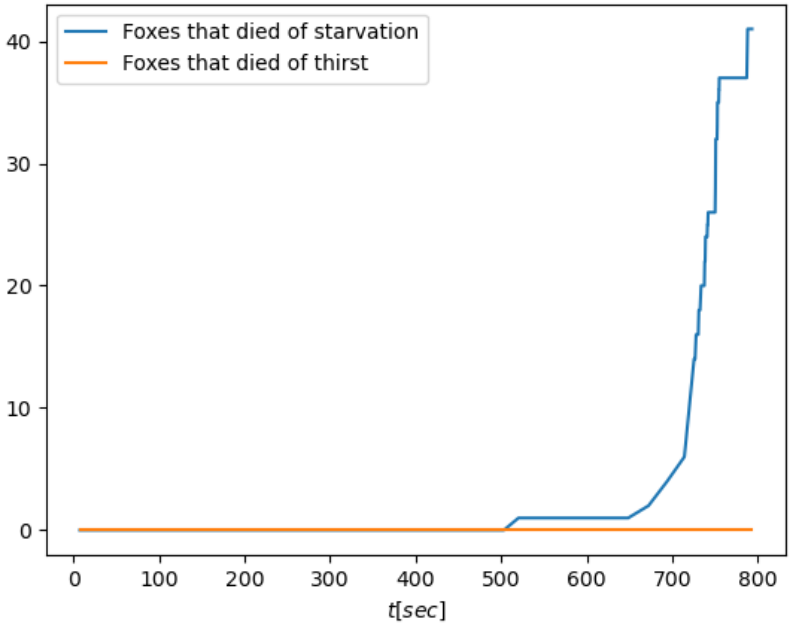

Figure 10.13: Foxes death cause.

10.3.3 Analysis

Population

As in previous scenarios, the rabbit population initially shot up. But then, due to the fact that it was an ideal opportunity for the foxes to thrive as they had food and drink in abundance, the situation reversed. At around second 350, the fox population started to rise strongly, while the rabbit population fell very sharply. However, with this rapid growth, the foxes sealed their own fate, because it led to the extinction of the entire rabbit population. For a while, foxes with their needs satisfied multiplied even more, until they ran out of food and they, too, began to die out rapidly. So a strong correlation can be seen between the two species and how prosperity can easily turn into inhospitable conditions.

Death causes

Most rabbits were obviously eaten by foxes, but quite a few deaths were also caused by thirst. On the other hand, no fox died of thirst. This was most probably due to the fact that the foxes, by staying at the watering holes, scared away the rabbits, which were afraid to approach the predators. Therefore, the foxes not only actively contributed to the extinction of the rabbits by eating them, but also passively cut off their access to water.

Features

In the case of rabbits, we see dynamically changing feature values, actively striving towards the most favourable values for rabbits in this environment. The features that were more valuable to the rabbits were fertility, and speed, as can be seen when the values of these features reached very high scores in the later stages of the simulation. Interestingly, sensory range apparently did not give the rabbits enough of an advantage to make it profitable to bear its genetic cost.

In contrast, for foxes, the values remained virtually the same throughout most of the simulations. This may be due to the fact that individuals of this species only started dying out towards the end, so the same genes were in the gene pool all the time. For the less adapted individuals, the environment was favourable enough for them to survive anyway, giving them the opportunity to pass on their weak genes to their offspring.

10.4 Simulation 4

This scenario is no different from the previous one, but the simulation went differently.

10.4.1 Initial state of the environment

The environment is exactly the same as the previous one.

10.4.2 Results

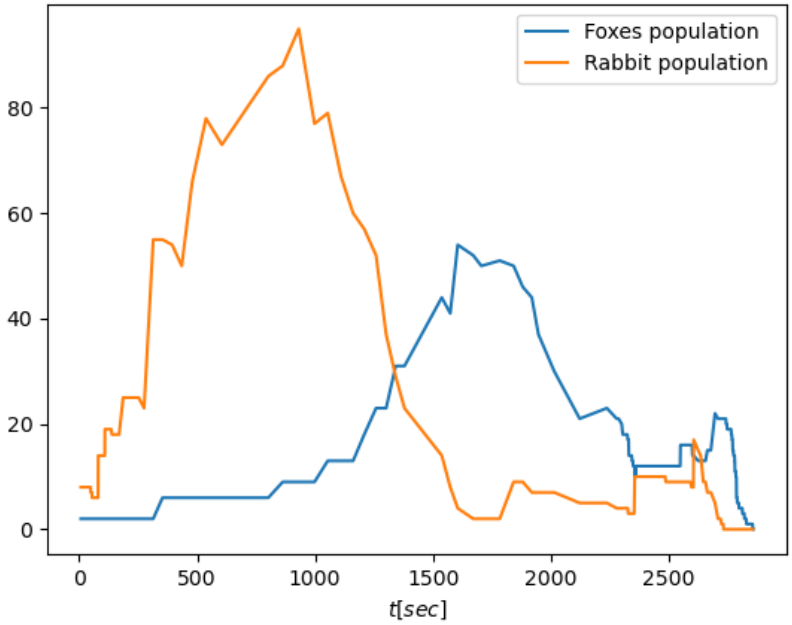

Figure 10.14: Populations of agents over time.

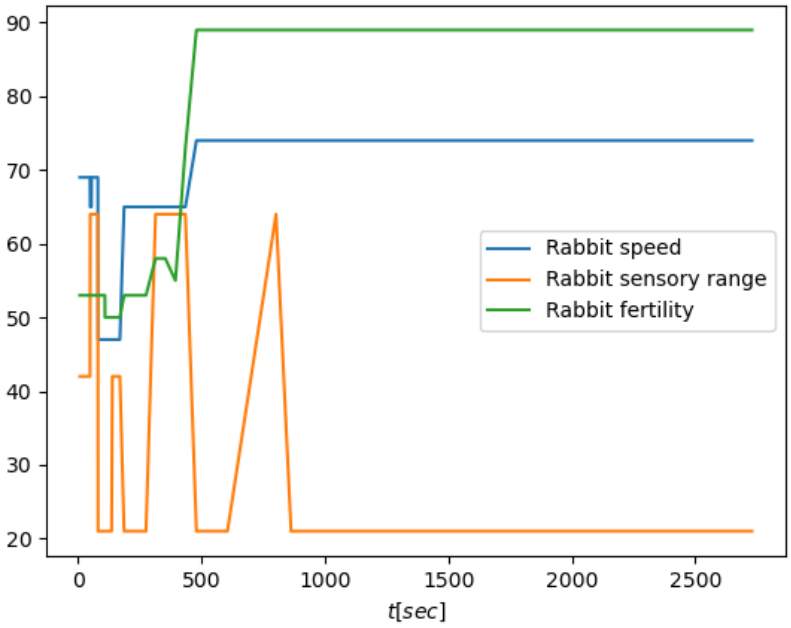

Figure 10.15: Median value of rabbit features over time.

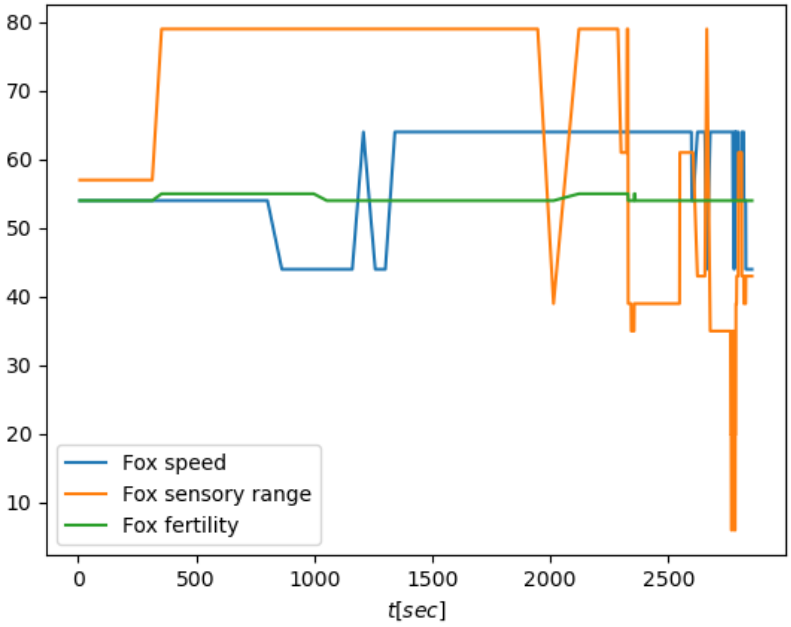

Figure 10.16: Median value of fox features over time.

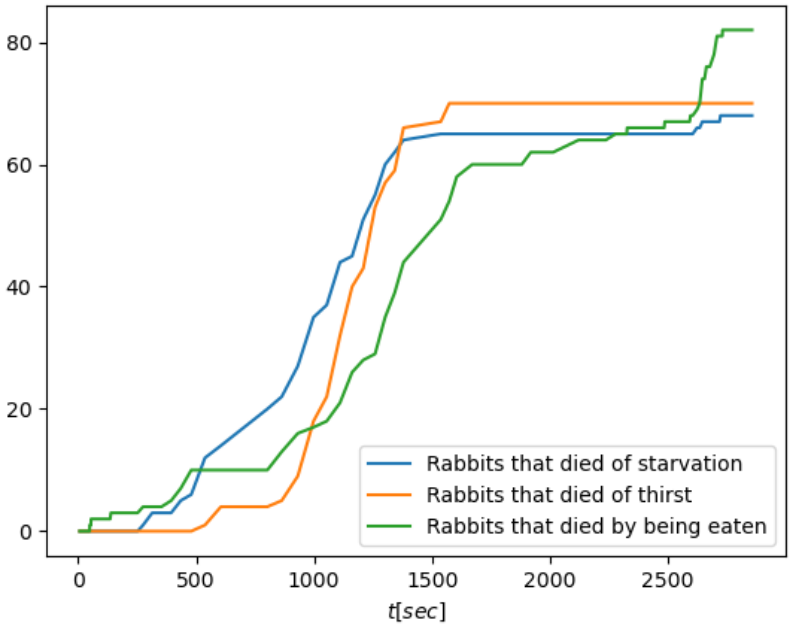

Figure 10.17: Rabbits death cause.

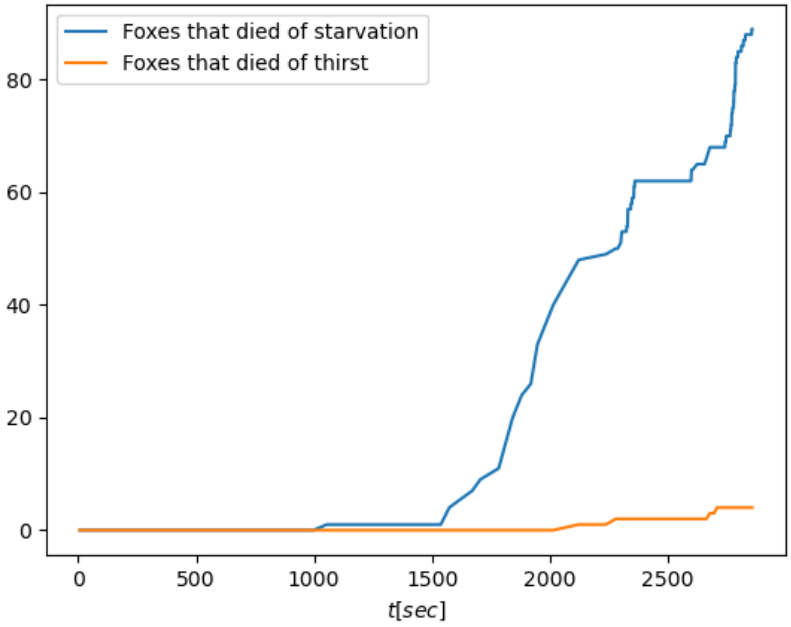

Figure 10.18: Foxes death cause.

10.4.3 Analysis

Population

In this scenario, as in the previous one, once the foxes had eaten a large proportion of the rabbits, they themselves began to starve to death. However, this time a few rabbits managed to survive and multiply again, but the foxes quickly dealt with them, this time finally. However, a certain cyclicality can be observed, if the rabbits had started to breed later, more foxes would have died out and they could have rebuilt their population more by reaching the same environmental state as originally (however with different, more adapted trait values nonetheless).

Death causes

Because the leaves started reproducing later than in the previous simulation, this led to more deaths from starvation, which were previously much lower. This time it had values similar to death from thirst and being eaten by a predator. In this simulation the availability of carrots started to play a role again, there were not enough carrots in the environment to satisfy such a growing population of rabbits.

Features

The dominant feature values for rabbits were established at the very beginning, where the weakest individuals were immediately eaten by the foxes. This led to a state where rabbits had a genetic advantage over foxes for most of the simulations, which was most likely the reason that foxes reproduced more slowly than in the previous simulation.

The features of foxes began to develop rapidly towards the end of the simulation, when foxes began to die out and only strong individuals could survive in this difficult environment. The final values of the features are as follows:

- Speed: 44
- Sensory range: 43
- Fertility: 54

Thus, it can be seen that for foxes each of the traits was important, but they could not afford their high values.

Chapter 11

Conclusions

The results of the simulations performed largely meet the expectations and agree with the logic and analytical models. An example is the fact that in a favorable environment, agents experienced significant growth, while the population vary under more extreme conditions. Interestingly, it was the harsh conditions that drove the genetic development of the agents, because it was then that weaker individuals were rejected by natural selection. The conclusion is that the conditions for reproduction must be ensured, as well as the danger of discarding the weakest individuals, for genetic development to proceed.

On the other hand, the simulation has some puzzling shortcomings, for example the final values of the fox traits in Simulation 4. In this case, it is surprising that a higher speed value, for example, would not be beneficial for the foxes. This may, of course, occurred due to the quality of the trained models. For example, in most of the simulations, the agents had a problem with collisions with water, and avoiding them was problematic for them. The second reason for weaknesses in the simulation may be that while the number of agents in the simulation was sufficient to observe interesting behavior, it was not large due to technical reasons. With so many agents, the simulation was already heavily loaded and it was demanding on hardware.

Nevertheless, the simulations carried out can definitely be considered successful. Its very unpredictability caused by randomness makes it possible to observe interesting behaviors that cannot be obtained in analytical models. Additionally, the presented model of the environment can be freely developed, increasing its possibilities, which is described in the chapter 12.

Chapter 12

Further development possibilities

12.1 Possible fields of application

- Computer games - fauna, realistic backgrounds that enhance gameplay and immersion
- More advanced simulations that need agents who realistically make decisions based on their needs
- Background for various types of animation
- Scientific research - modelling the actual ecosystem

12.2 Technicalities and improvement to implementation

Fuzzy systems for switching states - Instead of the current ranking system, fuzzy logic could be used, which seems a very good solution for this type of task. The state switching system itself would also become clearer and more consistent.

12.3 New features

Size - Influences strength and noticeability. Reduces the risk of attack by birds (they will not be able to carry the prey), but increases by foxes (larger animals give more energy).

12.4 New mechanics

Pregnancy - add a pregnancy period after mating interaction that offspring is spawned after. This will add another layer of complexity, as a pregnant rabbit may be eaten before its offspring are born.

Aging of agents - This is linked to the pregnancy mechanic, the agents would have different stages of development. Right after birth, young versions of agents would be spawned, with temporarily lowered trait values, unable to reproduce, or maybe even using slightly less trained policies to reflect their lesser experience in all activities.

Camoflage - extension of the spotting system, could significantly enhance the simulation and mechanics of hunting.

Attack sytem - another extension to hunting mechanic. Agents would have health points, and predators would have to first kill animal to be able to eat it.

Dying of plants - After a while the plant would wither and die. It would be possible to get rid of the parameter of the maximum number of plants around the generator, because the maximum number of plants would stabilise and remain naturally at a certain level, when the frequency of death of old plants would equal the frequency of appearance of new ones.

Better plant growth model - plants will grow differently depending on their position in the environment, for example they will grow better near water.

12.5 New States

Better chilling - add some agent behavior to chilling state

Sleeping - add sleeping state that agent reduces tiredness and regenerates itself

Dead - instead of disappearing immediately after death, the bodies of agents could remain in the environment for some time

12.6 New Species

Hawk - eats rabbits. Reproduces through laying down eggs that can be eaten by foxes (but it would give much less energy than rabbit, so foxes would rather hunt for them), so reproduction for hawks would be more challenging than for rabbits, for example. Hawks could only eat small rabbits, so this feature would be more relevant. This species, however, would have their own unique characteristics that could give them an advantage in certain environments, such as flying (their movement would not be restricted by environmental elements and they could spot prey more easily and the prey would be less likely to escape).

Scavenging species - eats dead agents

Omnivorous species - eats other agents and plants

Bibliography

[1] AspidistraK. https://commons.wikimedia.org/w/index.php?curid=56484640. Own work, CC BY-SA 4.0.

[2] S.C. Bhargava. Generalized lotka-volterra equations and the mechanism of technological substitution. *Technological Forecasting and Social Change*, 35(4):319–326, 1989.

[3] Charles Hall and John W. Day. *Ecosystem modelling in theory and practice: An introduction with case histories*. University of Colorado, Department of Fine Arts, 1977. ISBN: 9780870812163.

[4] David Harris and Sarah Harris. *Digital Design and Computer Architecture, 2nd Edition*.

[5] Lotka-volterra equations. https://en.wikipedia.org/wiki/Lotka\OT4\textendashVolterra_equations.

[6] Stephen Marsland. *Machine Learning An Algorithmic Perspective*. 2 edition, 2015. ISBN: 9781466583337.

[7] Melanie Mitchell. *An introduction to genetic algorithms*. MIT, 1998. ISBN: 0262631857.

[8] Ml-agents toolkit glossary. https://github.com/Unity-Technologies/ml-agents/blob/main/docs/Glossary.md.

[9] Ml-agents observations ans sensors. https://github.com/Unity-Technologies/ml-agents/blob/main/docs/Learning-Environment-Design-Agents.md.

[10] Simulations results. https://github.com/kerdamon/Engineering-Thesis-Simulating-Ecosystem/tree/simulation/EcosystemSimulation/SimulationResults.

[11] Tensorboard tool. https://github.com/Unity-Technologies/ml-agents/blob/main/docs/Using-Tensorboard.md.

[12] Unity glossary. https://docs.unity3d.com/Manual/Glossary.html.

[13] Ml-agents toolkig configuration file description. https://github.com/Unity-Technologies/ml-agents/blob/main/docs/Training-Configuration-File.md.

[14] Joe Ward. The reinforcement learning cycle. https://github.com/Unity-Technologies/ml-agents/blob/main/docs/images/rl_cycle.png.

[15] What is computational intelligence? https://cis.ieee.org/about/what-is-ci.

Appendix A

Training files

A.1 Configuration Files

The following are the configuration in yaml format used in the ML-Agents toolkit.

The individual files differ only in the values of the rewards and the size of the test environment (in hunting and escaping, other models use one from common configuration), the other parameters are the same for each test environment and are shown once in common section below.

The meaning of each parameter is described in the documentation for the ML-Agents toolkit [13].

A.1.1 Common Configuration

```
behaviors:
  AgentMovement:
    trainer_type: ppo
    hyperparameters:
      batch_size: 1024
      buffer_size: 10240
      learning_rate: 0.0003
      beta: 0.005
      epsilon: 0.2
      lambd: 0.95
      num_epoch: 3
      learning_rate_schedule: linear
    network_settings:
      normalize: false
      hidden_units: 256
      num_layers: 1
      vis_encode_type: simple
    reward_signals:
      extrinsic:
        gamma: 0.99
        strength: 1.0
    keep_checkpoints: 5
    max_steps: 1000000
    time_horizon: 64
    summary_freq: 10000
environment_parameters:
```

```
  is_training: 1

  # features
  agent_speed:
    sampler_type: uniform
    sampler_parameters:
      min_value: 0
      max_value: 100
  agent_sensory_range:
    sampler_type: uniform
    sampler_parameters:
      min_value: 0
      max_value: 100

  # training area size
  training_area_size:
    curriculum:
      - name: SmallAreaSize
        completion_criteria:
          measure: progress
          behavior: AgentMovement
          min_lesson_length: 5
          threshold: 0.1
        value: 0.5
      - name: MediumAreaSize
        completion_criteria:
          measure: progress
          behavior: AgentMovement
          min_lesson_length: 5
          threshold: 0.4
        value: 1.0
      - name: LargeAreaSize
        completion_criteria:
          measure: progress
          behavior: AgentMovement
          min_lesson_length: 5
          threshold: 0.7
        value: 2.0
      - name: ExtraLargeAreaSize
        value: 3.0
```

A.1.2 Drinking Configuration

```
environment_parameters:
  rabbit_each_episode_fixed: -1.0
  fox_each_episode_fixed: -1.0
  agent_bump_into_wall: -0.2
  agent_bump_into_food: -0.2
  agent_drink_reward: 1.0
```

A.1.3 Eating Carrot Configuration

```
environment_parameters:
  rabbit_eating_carrot_reward: 1
  rabbit_each_episode_fixed: -1
  agent_bump_into_wall: -0.2
```

```
    agent_bump_into_water: -0.2
```

A.1.4 Mating Configuration

```
environment_parameters:
  rabbit_each_episode_fixed: -1
  fox_each_episode_fixed: -1
  agent_bump_into_wall: -0.01
  agent_bump_into_water: -0.01
  agent_bump_into_food: -0.01
  rabbit_mating_reward: 1
  fox_mating_reward: 1
```

A.1.5 Hunting Configuration

```
environment_parameters:
  # rewards
  fox_eating_rabbit_reward: 1
  fox_each_episode_fixed: -1
  agent_bump_into_wall: -0.01
  agent_bump_into_water: -0.01
  agent_bump_into_food: -0.01

  # training area size
  training_area_size:
    curriculum:
      - name: SmallAreaSize
        completion_criteria:
          measure: progress
          behavior: FoxMovement
          min_lesson_length: 5
          threshold: 0.15
        value: 1.0
      - name: MediumAreaSize
        completion_criteria:
          measure: progress
          behavior: FoxMovement
          min_lesson_length: 5
          threshold: 0.5
        value: 1.0
      - name: LargeAreaSize
        completion_criteria:
          measure: progress
          behavior: FoxMovement
          min_lesson_length: 5
          threshold: 0.75
        value: 1.0
      - name: ExtraLargeAreaSize
        value: 1.0
```

A.1.6 Escaping Configuration

```
environment_parameters:
  # rewards
```

```
  rabbit_each_episode_fixed: 1
  rabbit_on_eaten: -1
  agent_bump_into_wall: -0.01
  agent_bump_into_water: -0.01
  agent_bump_into_food: -0.01

  # training area size
  training_area_size: 1.0
```

A.2 Results

The graphs were created using the TensorBoard tool. [11]

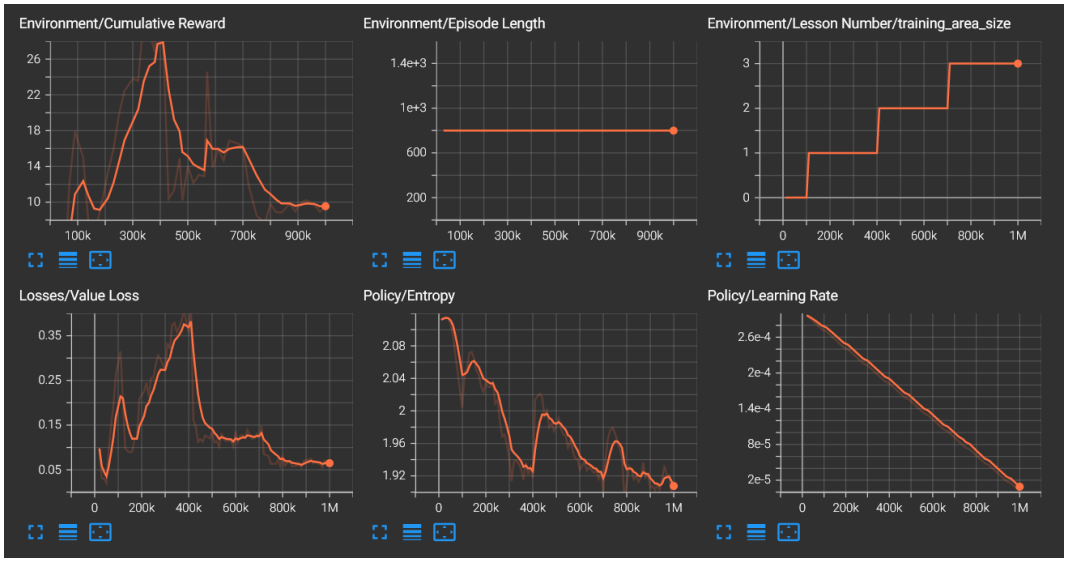

Figure A.1: Results of training Drinking model.

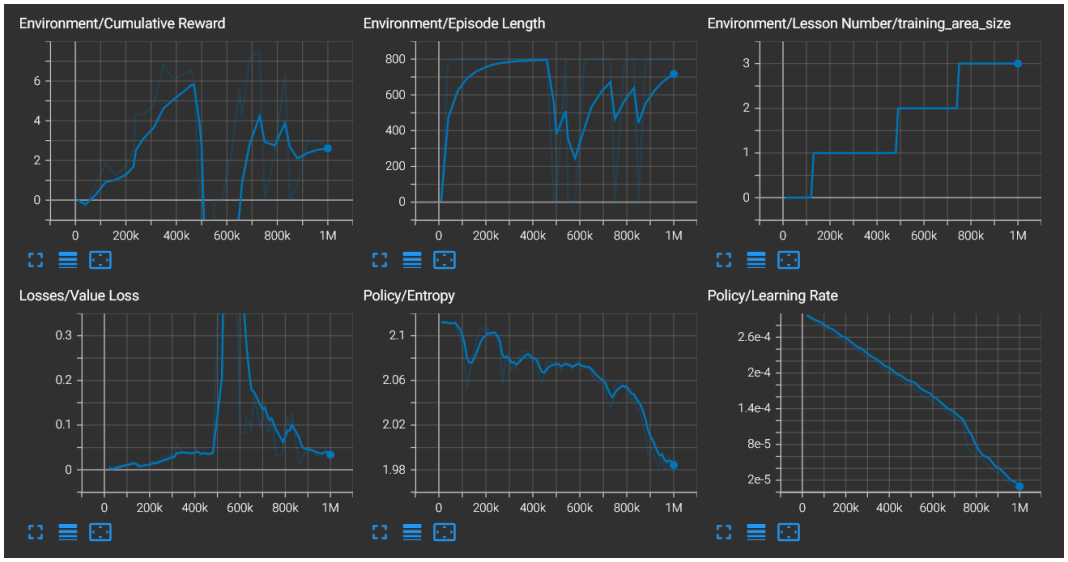

Figure A.2: Results of training EatingCarrot model.

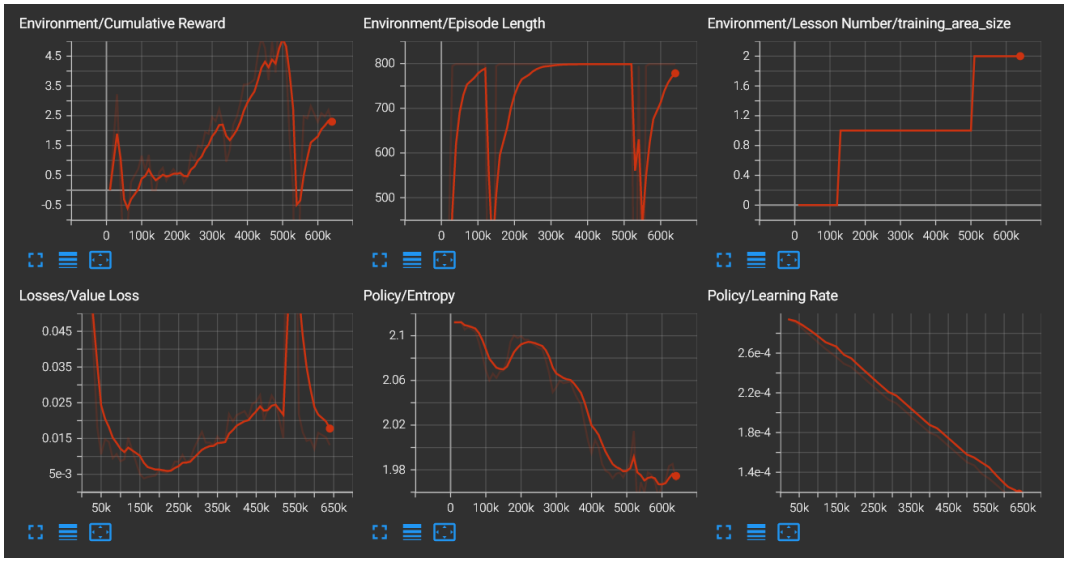

Figure A.3: Results of training Mating-Rabbit model.

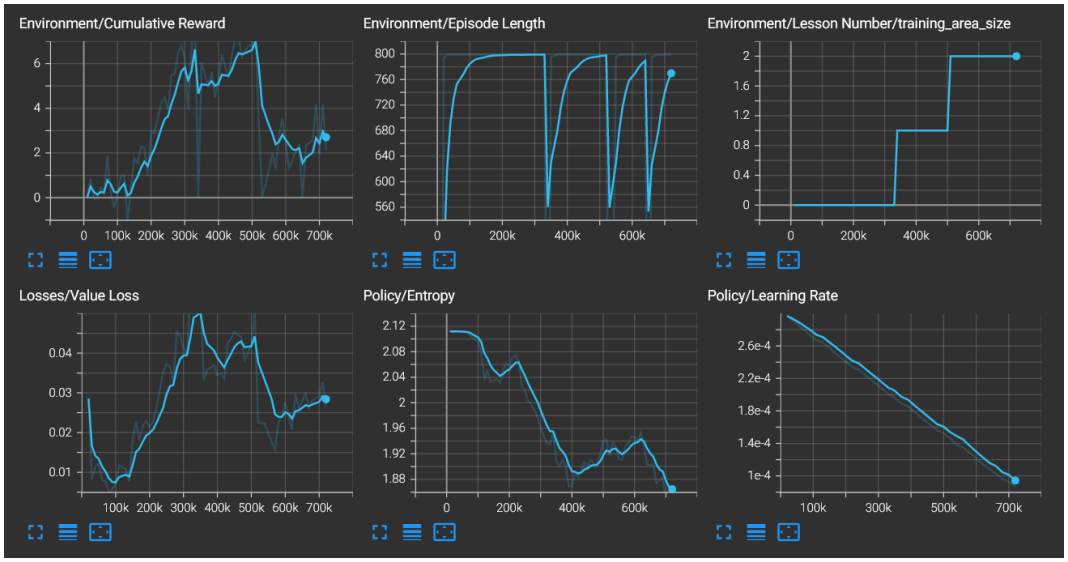

Figure A.4: Results of training Mating-Fox model.

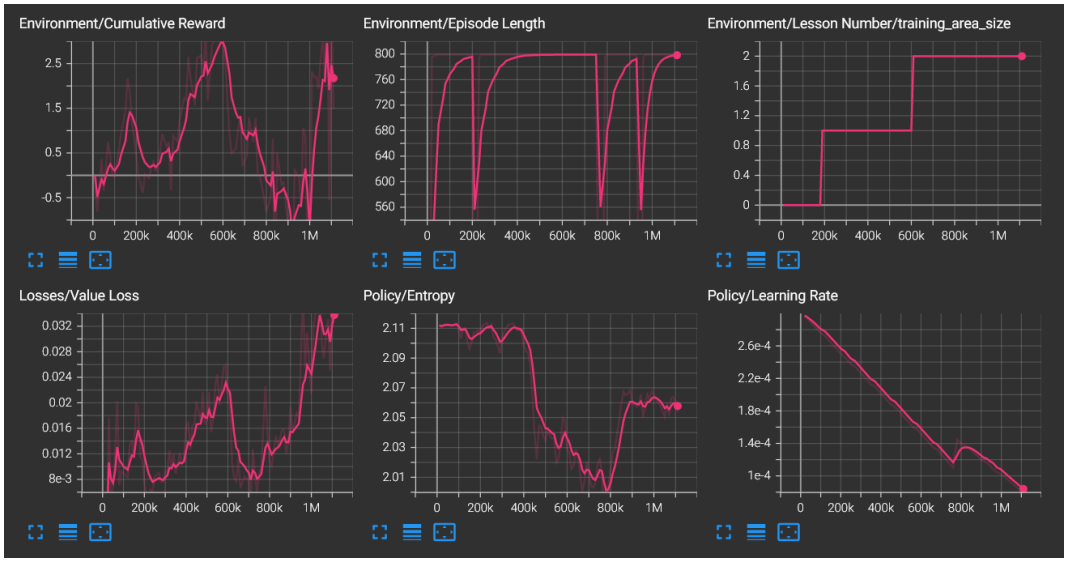

Figure A.5: Results of training Hunting model.

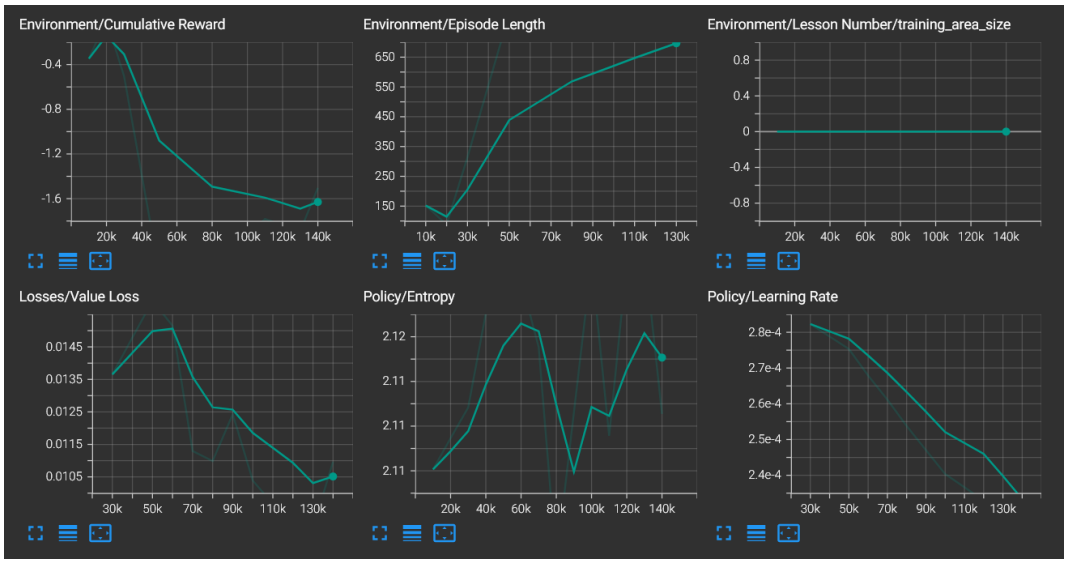

Figure A.6: Results of training Escaping model.

Appendix B

Packages and Versions

This section describes tools and packages used to create the project.

Unity Editor 2020.3.25f1

`https://unity.com/download`

Unity ML-Agents Toolkit 2.0.1

`https://github.com/Unity-Technologies/ml-agents`

Grid Sensors for Unity ML-Agents by mbaske - version 2.0

`https://github.com/mbaske/grid-sensor`

NaughtyAttributes 2.1.1

`https://assetstore.unity.com/packages/tools/utilities/naughtyattributes-129996`

Free Fly Camera 1.2

`https://assetstore.unity.com/packages/tools/camera/free-fly-camera-140739`