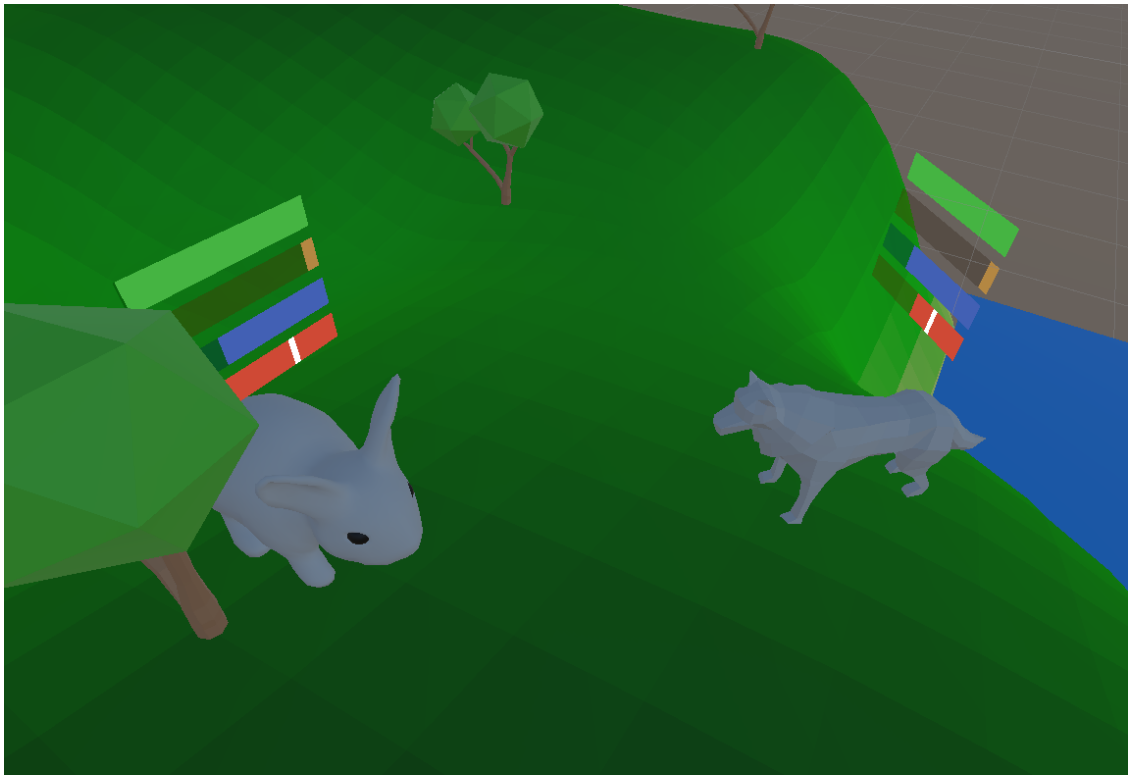


Simulating an Ecosystem

Exploring the possibility of generating terrain-dependent non-player character behaviour by using an evolutionary-based fuzzy cognitive map

Bachelor of Science Thesis in Computer Science and Engineering

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

Bachelor's thesis 2020

Simulating an Ecosystem

Exploring the possibility of generating terrain-dependent non-player character behaviour by using an evolutionary-based fuzzy cognitive map

BEN HABAGIL
JONATHAN KÖRE
ISAK SCHWARTZ
ALEXANDER SOLBERG
AUGUST SÖLVELAND
ROBERT ZETTERLUND

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
DATX02-089
Chalmers University of Technology
Gothenburg, Sweden 2020

Simulating an Ecosystem
Exploring the possibility of generating terrain-dependent non-player character behaviour by using an evolutionary-based fuzzy cognitive map Bachelor of Science Thesis in Computer Science and Engineering
Ben Habagil, Jonathan Köre, Isak Schwartz, Alexander Solberg, August Sölveland, Robert Zetterlund

Supervisor: Michael Heron, Department of Computer Science and Engineering
Examiner: Mohammad Obaid, Department of Computer Science and Engineering

Bachelor's Thesis 2020
Department of Computer Science and Engineering
DATX02-089
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: A picture of a prey and a fox in the ecosystem.

Gothenburg, Sweden 2020

Abstract

Providing immersive experiences in video games often proves to be difficult as rudimentary Non-Playable Characters (NPCs) can appear repetitive to players. In an attempt to address this issue, we used the game engine Unity to implement a behavioral model using evolving Fuzzy Cognitive Maps (FCMs). An FCM is a weighted directed graph where nodes represent concepts and edges determine their causality. Through simulations in a procedural generated environment and usage of genetic algorithms, we sought a correlation between the decision-making of animals and their environment. As the animals' FCMs processed input from the environment, their decision making developed throughout several generations. Simulations, done with three different environments, have shown that animals simulated in hostile environments evolved to prioritise certain actions such as escaping as opposed to those in peaceful environments. Evolved FCMs consistently performed better than their predecessors and we concluded that our model could be used to successfully create environment dependent decision making of NPCs in a video game setting.

Keywords: Ecosystem, Fuzzy Cognitive Map, Genetic Algorithms, Procedural content generation, Non playable characters, Open world.

Sammandrag

Att tillhandahålla uppslukande upplevelser i datorspel visar sig ofta vara svårt eftersom rudimentära icke-spelbara karaktärer (NPC:er) kan verka repetitiva för spelare. I ett försök att lösa detta problem implementerade vi en evolverande beteendemodell i spelmotorn Unity med Fuzzy Cognitive Maps (FCM:er) som grund. En FCM är en riktad och viktad graf där noder representerar koncept och kanter bestämmer dess kausalitet. Genom simuleringar i en procedurgenererad miljö och genom genetiska algoritmer sökte vi en korrelation mellan djurens beteende och deras miljö. När djurens FCM bearbetade indata från miljön över flera generationer utvecklades deras beslutsfattande. Simuleringar, utförda i tre olika miljöer, visar att djur som simuleras i fientliga miljöer utvecklas till att prioritera vissa åtgärder som att fly i motsats till dem i fredliga miljöer. Evolverade FCM:er presterade konsekvent bättre än sina föregångare och vi drog slutsatsen att vår modell skulle kunna användas för att framgångsrikt skapa omgivningsberoende beslutsfattande av NPC:er i en datorspelsmiljö.

Acknowledgements

We would like to extend our gratitude to our supervisor Michael Heron and his ability to come up with interesting ideas and for being available for discussions.

Glossary

AI – Artificial Intelligence

CA – Cellular Automaton

CPU – Central Processing Unit

FCM – Fuzzy Cognitive Map

FPS – Frames Per Second

Fuzzification – The process of converting a value to a fuzzy variable

Fuzzy variable – A real number bound by $[0, 1]$ that determine the degree of truth.

GA – Genetic Algorithm

NPC – Non-Playable Character

PCG – Procedural Content Generation

Contents

1

Introduction

The majority of game studios employ techniques for: generating environments, creating complex behaviour models and enabling emergence of behaviour [1] [2]. As advances in hardware and software allow personal computers to run more complex computations, more open-world games are created than ever before [3]. In *The Legend of Zelda: Breath of the Wild*, players traverse various environments throughout the open-world, exploring new biomes and interacting with different types of non-playable characters (NPCs) [4]. Automatically generating these types of vast environments with near infinitely many variations is a technique deployed in many games today, such as Minecraft and Terraria. By combining these techniques with an evolving behavioral model, games could provide more diverse NPCs, potentially improving the experiences for players.

The following sections briefly present an overview of relevant history and knowledge within the areas of ecosystem modelling, evolutionary algorithms and procedural content generation (PCG), introducing the reader to the context of this project.

1.1 Background

NPC behaviour in Game Development

NPCs have been a critical part of video games since their creation. The first NPC algorithm was created before there were powerful enough computers able execute the program. It was a chess algorithm, created by Alan Turing, that could be used to calculate the best move algorithmically [5]. With improvements to both hardware and software, more complex NPCs have since been developed making them much more advanced. These NPCs exhibit Artificial Intelligence (AI) which mimics thought and actions that are rational [6] [5].

There are a couple of reasons why AI is an important part of video games, the most obvious being that it allows for one person to play a game as if playing with real humans. In games where this is desirable there is a clear benefit from an AI being as advanced (in this case human-like) as possible. Although this not always the case since it can also be desirable to have an AI which behaves nonhuman, for example in order to increase immersion. For example if the player is tasked to overcome a

particular challenge, the level of difficulty should always be the same so that when it i mastered, the challenge becomes easy [5].

A key part of human intelligence is to adapt to a situation which is one of the biggest challenges in AI. A program is generally specified to do one thing and one thing only, for example a chess computer could never be used to autopilot a plane [5]. As most NPCs use predefined algorithms they become predictable given enough experience or insight [7]. Scripted behaviour conflicts with what is deemed important in video games according to one study done on adults aged 19-39 [8] and another study done on 1397 consumers across more than half a million rounds of gameplay [9]. The first study showed that the most important feature that a game was to have "lots of variety", whilst the second study showed that the consumers generally preferred variety as a part of games.

To compensate for AIs lack of adaptability, evolution could be implemented in video games since it could provide more dynamic interactions, as in an ecosystem. This could create an abundant range of interesting behaviours in NPCs that differ depending on their environment, which in turn makes them seem more realistic. This is particularly relevant in games where random terrain generation is used because the environment that the NPC finds itself in is different each time [1]. Exploration games such as Minecraft might benefit from evolving NPCs by providing more content to discover. However, in games such as Halo, it is important that NPCs behave in a predictable manner so that players can form a strategy around their experiences [10]. Note however that evolving behaviours are not inherently unpredictable. Although behaviour may differ across different environments, they could simultaneously be predictable inside each environment. Similar to how a rabbit in Sweden behaves differently from a rabbit living in a desert.

Simulating Ecosystems

Simulating living entities has long been a subject of research [11]. Ranging from state-based models such as Lotka-Volterra [12], where populations of predators and preys are represented as differential equations, to individual-based models attempting to treat animals as independent beings [11] [13]. The populations of the Lotka-Volterra model can be seen in Figure 1.1.

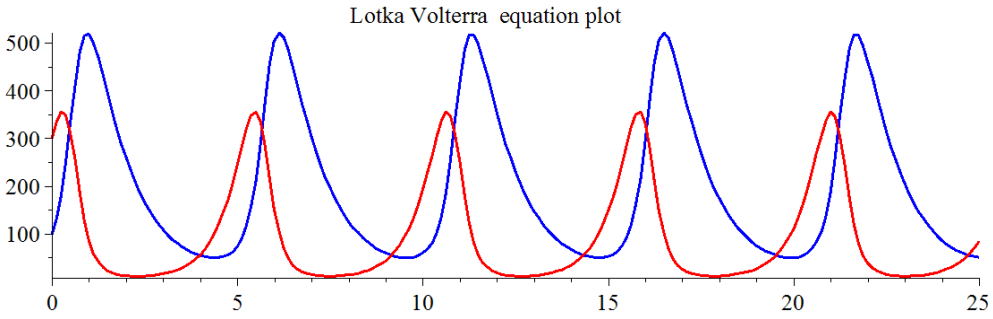


Figure 1.1: Population of predators (blue) and preys (red) as a function of time, as represented by the Lotka-Volterra model.

Source: [14]

Common approaches to simulating ecosystems

An example of a discrete simulation model is a *Cellular Automaton* (CA). The basic concept of CAs consist of having a 2d-matrix of square cells which assume various states dependent on the surrounding environment along with rules set in advance. Conway's Game of Life [15] [16] is one of the most well-known CAs where two states (on or off) exist. Preexisting rules determine the outcome of a cell, depending on the eight cells surrounding it. Figure 1.2 shows interactions inside a Game of Life CA in consecutive snapshots.

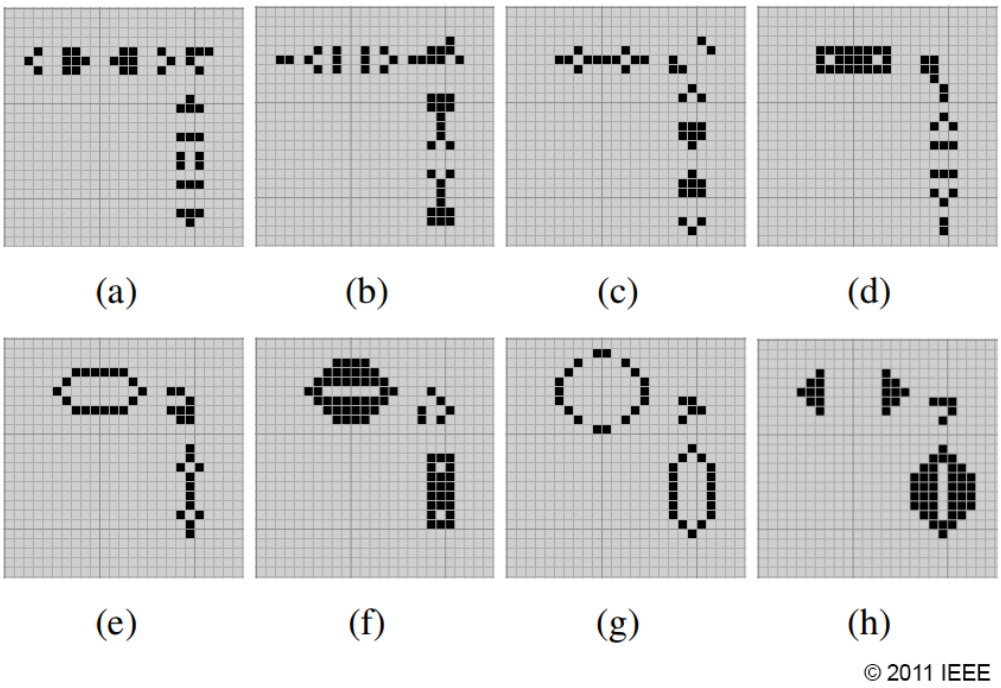


Figure 1.2: Eight consecutive frames in an instance of Game of Life.

Source: [17]

Discrete models such as CA can be used in theoretical biology to analyze interactions in ecosystems with regard to spatial surroundings [15] [18]. One such CA used for simulating animals could be configured by defining rules such as a cell with a rabbit surrounded by two cells with foxes should cease to exist and become grass. Discrete models provided simpler and more computationally efficient simulations and this simplicity resulted in CA becoming one of the most common models for simulating ecosystems [18].

Another approach to simulating evolving organisms is to use *Decision Tree Modelling* which are also relevant in recent research with applications such as diagnosing cancer and Alzheimer's disease [19] [20]. Decision trees are hierarchical structures used in machine learning to predict values that depend on some variables by creating a model using data for said variables. Decision trees can therefore be used to model populations and habitats, as explained by M. Debeljak and S. Džeroski [21]. Figure 1.3 demonstrates how a decision tree can be used to estimate the number of anecic earthworms in soil given a set of variables including pH value.

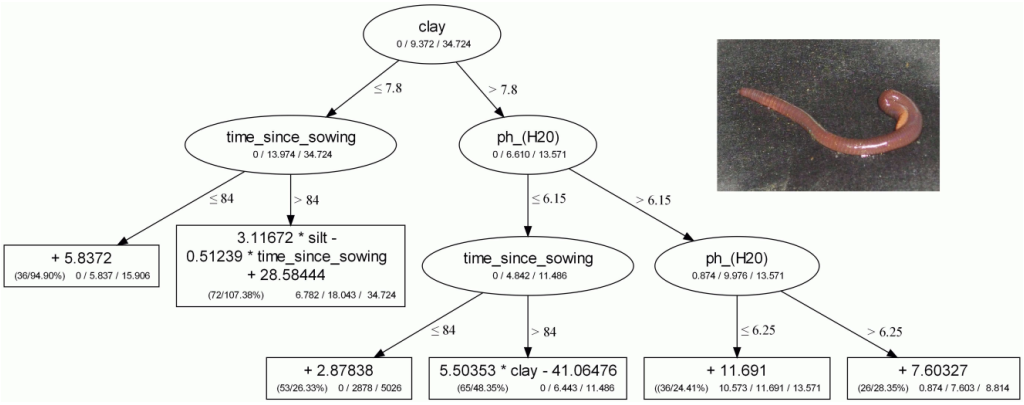


Figure 1.3: A decision tree for estimating the presence of anecic earthworms. Each node contains the corresponding minimum, mean and max biomass of worms and the leaves also contain their number of examples and relative root mean square error (Debeljak et al, 2007). The photo in the upper right shows an Epigeic earthwork Eisenia fetida (Lumbbricidae). Courtesy of Paul Henning Krogh.

Source: [22]

The advantages of decision trees highlighted by M. Debeljak and S. Džeroski include the capability to not only model interactions between variables and outcomes but also interactions between variables as well as having a high efficiency in terms of generating models and making the actual predictions using the models. Another advantage is that no underlying knowledge of the domain or the interactions between variables is needed since the decision trees model these automatically. This does however come with an inherent downside which is emphasized by M. Debeljak and S. Džeroski; a lot of data (or highly accurate data) is needed to create the models [21].

Procedural Content Generation

The general idea of *Procedural Content Generation* (PCG) is to use algorithms to generate content, rather than having content be created by hand [23]. This can be used as an advantage both during game development and in real time as a game is played. During development, it becomes possible to generate assets with an algorithm and having designer refine them, opposed to manually creating it [1]. During runtime, PCG allows for a lot of unique content to be generated quickly allowing for unique experiences every time new content is generated. The downside to real time PCG is that the content generated will not be as refined as if a designer had worked on it. An early example of PCG used in real time is in the game *Rogue* (1980), where every time the player climbs down a set of stairs a new unique level is generated [24]. A more recent example is in Minecraft (2009), where the terrain is continuously generated as the player explores the environment [25]. Combining an emerging terrain-dependent behavioural system with PCG could allow for a great variety of content, not only in the terrain as in Minecraft, but also in the behaviour of the NPCs.

1.2 Purpose

The purpose of this project is to explore the combination of behaviour models and procedurally generated environments in order to create dynamic ecosystems. This can serve as a starting point for developers that are trying to create open world environments filled with different types of NPCs. Our purpose can be divided into three objectives, more specifically:

1. The modelling of animals equipped with senses to perceive their surroundings and use a behaviour model to make decisions.

2. The creation of a behaviour model capable of evolving over time.

3. The ability to generate environments that are different enough to affect the evolution and the resulting behaviour.

For the first objective, a successful decision making model is deemed a model that is able to take multiple aspects into account in real-time. For the second objective, a successful evolution of behaviour is one where the NPCs ability to survive in their environment is increased naturally without any intervention. For the final objective, a successful creation of environments is the ability to algorithmically create several coherent environments which partially represents a real world biome.

1.3 Scope

The animals will be restricted to two species, rabbits and foxes, being the minimum amount of species needed in order to create a predator-prey simulation. This should make it easier to observe how two species' behaviours impact each other. Further-

more, each simulation will take place in one type of terrain at the time, so that resulting behaviours can be tied to a specific terrain. Only two types of terrain will be used, more specifically plains and a mountainous region.

The main focus will not be towards perfecting the procedural generation. While its important that the generated terrain is decent enough to impact its inhabitants, a relatively small focus will be directed towards making it aesthetically pleasing or diverse.

In order to fully focus on the objectives mentioned in the Purpose and to save time, this project will be created in an already existing Game Engine, namely Unity.

The actions performed by each species will be hard-coded and thus be non susceptible to change. The behaviour will instead be based on when to perform actions, rather than what the actions will do.

1.4 Thesis outline

The second chapter introduces the reader to relevant theory behind fuzzy cognitive maps, procedural content generation and evolutionary algorithms needed to understand the paper. Chapter 3 presents the methodology and creation of the ecosystem, detailing the implementation of the ecosystem. Results of the simulation are presented in Chapter 4 and discussion is carried out in Chapter 5. Finally, conclusions are drawn in Chapter 6.

2

Theory

In this section we present the theory behind fuzzy cognitive maps, terrain generation and genetic algorithms.

2.1 Fuzzy Cognitive Maps

In 1986, B. Kosko presented fuzzy cognitive maps (FCMs) [26] which model individual reasoning by using a fuzzy-graph structure building on the concept of fuzzy logic introduced by L.A. Zadeh in 1965 [27] and the extension of cognitive maps made by Axelrod in 1976 [28].

Fuzzy-logic is used to represent granularity and uncertainty of an element's membership to a group, traditionally within the unit interval I ($[0, 1]$), 0 representing no membership and 1 absolute membership [27]. Zadeh exemplifies using the group known as animals; where humans, birds, rocks and plants clearly belong or do not belong to the group whereas uncertainty arise around organisms such as bacteria and starfish [27]. These organism's memberships are *fuzzy*, and the extent of the membership can be determined via *fuzzification* into a fuzzy variable ranging from 0 to 1. Hence, fuzzification can be viewed as function with the co-domain $[0, 1]$ and with a domain of a predetermined set. Usages such as representing distance as close or far [13], representing water temperature as either hot or cold [11] and using short or tall to classify people's height [29] are examples of fuzzification. Membership functions (MF) are used for fuzzification, both linear and non-linear (sigmoid, gaussian) exists, with non-linear being deemed the most accurate within modelling of ecosystems [11]. Figure 2.1 presents different ways of determining distance as a membership of close and far. A crisp MF determines membership of *either* close or far, whereas the fuzzification performed by the linear and the sigmoid MFs allows for partial membership of both close and far.

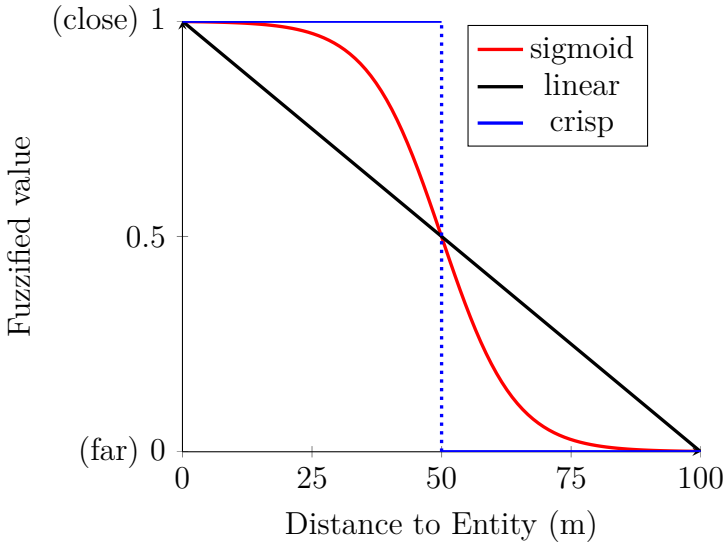


Figure 2.1: Three different membership functions processing distance as being close or far away.

Cognitive maps are used to model processing and evaluation of information, and in individuals they are also used as a way to store and memorize information. Cognitive maps were originally used for navigation and remembering geographical knowledge, however, in this thesis the concept of an cognitive map is the one adopted by researchers in Operations Research (OR), where optimal decision-making is studied [28]. In OR a cognitive map is seen as a model for processing inputs and then being able to extract an optimal output.

The uncertainty and softness of cognitive mapping and the popularity of fuzzy-logic during the 80s led to the creation of the Fuzzy Cognitive Map. An FCM is a graph structure of directed weighted edges and nodes. Nodes are denoted as concepts and a weighted edge represent the causality between two concepts. A concept can have unlimited connected edges and the degree of causality can be modified or set at any time. The simplistic graphical structure and the concepts being represented by words or sentences makes FCMs simple to model and to understand. The creation of FCMs can be supported in various ways, either by questionnaires to experts, performing content analysis on text, using available data, in depth interviewing process or by intuition [30]. FCMs are multi-purpose and can be applied to various problems, four categories are highlighted in particular [31]:

- Explanatory. Which is modelling *a posteriori*, attempting to reason about events after they have occurred.
- Prediction. Which is modelling *a priori*, attempting to reason about events before they occur.
- Reflective. Which is modelling under an occurring situation, highlighting aspects and prompting discussions about causality between concepts.

- Strategic. Which revolves around acquiring knowledge around complex situations.

Reflective FCMs can act as a decision support system, where optimal decisions are made based on the available surroundings. The fuzzy variables handle the concept of partial truths, and are *inferred* and then *defuzzified*. An FCM is inferred onto some behaviour and later defuzzified, in order to decide what intensity to apply the inferred behaviour. For example, in 2009 an ecosystem was simulated using an FCM [13], where an agent behaved differently based on the distance to a foe, the idea of the logic shown in Figure 2.2. Distance was implemented as a fuzzy variable along with sensitivity concepts such as `FoeClose` and `FoeFar`. These would affect the affected concept variable `Fear` which would - given the exceeding of a set threshold - invoke the action of `Evasion` where defuzzification would determine the speed of which the animal would evade.

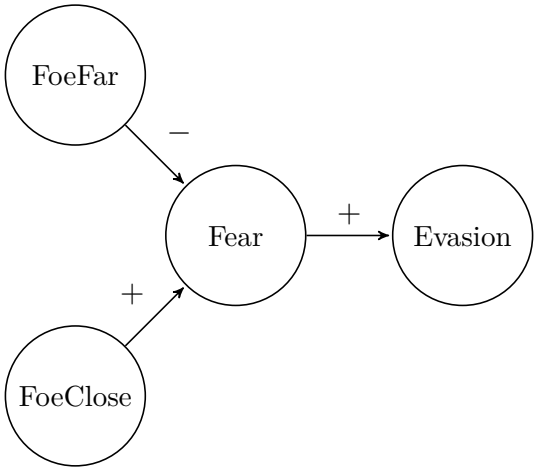


Figure 2.2: A small FCM showing an intuitive correlation between the closeness of foes, fear and the act of evasion, recreated from [13]

Illustrated below in Figure 2.3 is a general example of an FCM consisting of 6 concepts and 7 weighted edges. There are two different types of concepts in an FCM, causal and affected concepts. For example, C_1 is a causal concept with respect to the affected concept C_3, due to edge $W_{1,3}$. Hence, the affected concept C_m is affected by the causal concept C_n by the weight of $W_{n,m}$ bound by $-1 \leq W_{n,m} \leq 1$. Every concept C_n has a fuzzy variable denoted A_n, bound by $0 \leq A_n \leq 1$, which represents the value of the concept C_n [32]. This value, A_n, called the activation level which - together with outgoing weights from C_n - determine the strength to affect other concepts. Hence, computation of A_3 whould be $A_3 \mathrel{+}= A_1\dot{W}_{1,3}$ [13].

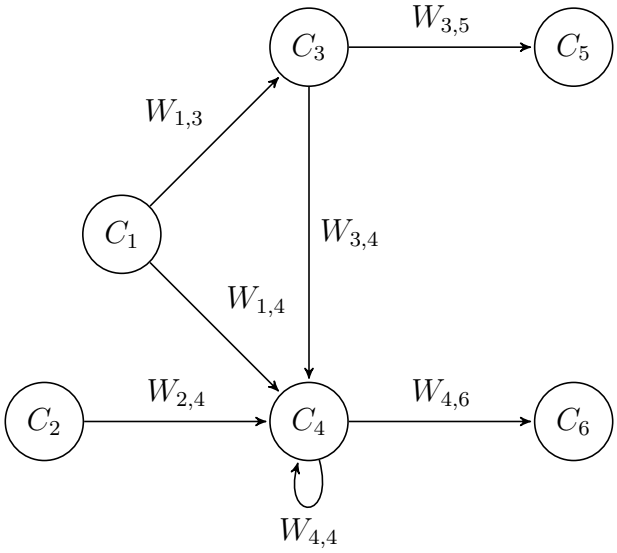


Figure 2.3: An FCM with 6 concepts C_n and 7 weighted edges W_{nm}.

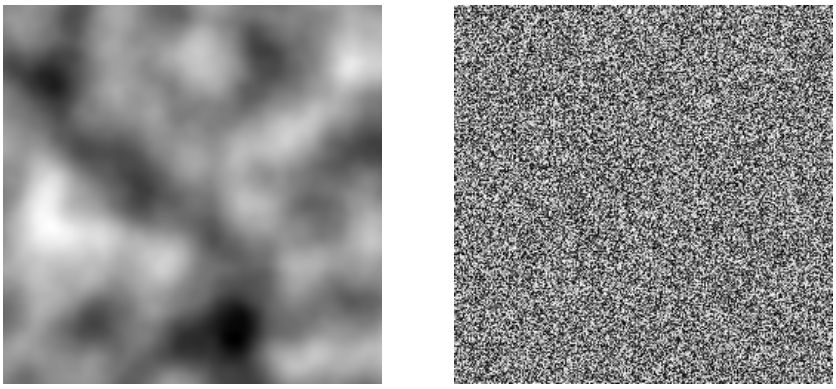
2.2 Procedural Content Generation

Procedural generation is a broad term which includes several different concepts and techniques. Some examples of techniques used are Perlin noise, L-systems and Voronoi diagrams. The completeness of an algorithm which generate content varies, where some techniques require a lot of human modification post generation in order to produce anything of value, whilst others are entirely non-assisted. A non-assisted technique is one that produces a result, algorithmically, and is sufficient without any post modification. These techniques require less effort in producing a large amount of different content, however it becomes harder to ensure coherence. Using an assisted technique instead makes it easier to limit randomness at the cost of more effort. A concept or technique is not assisted or non-assisted on its own but rather dependent on context and how it is used. The problem becomes balancing the amount of effort required, due to the limited amount of time, with the number of different environments generated [1] [33].

2.2.1 Perlin Noise

One technique that can be used to generate a world is via the use of a noise map. A noise map is simply an array of values where each value ranges from 0 to 1. Each value in a noise map can be generated randomly or pseudo-randomly. A completely random noise map gives rise to a incoherent world which does not resemble any real landscape, as seen in Figure 2.4b. To circumvent this, a pseudo-random noise map generator is commonly used. The pseudo-random noise map looks more coherent and the worlds generated exhibit more realistic characteristics. A type of pseudo-random noise is Perlin noise which can be generated in any dimension. Two dimensional Perlin noise works by having an two dimensional array of integers where each integer has a predefined vector with the length 1. To calculate the value at any point, four

new vectors are defined; one from each corner of the square the point exists in, to the new point. Each of these new vectors are multiplied by that points predefined vector which gives four numbers, one for each corner. Finally, each of these numbers contribution is weighted and interpolated along one axis by a so called "smooth" function (the exact function differ but commonly a third degree polynomial) and these values are linearly interpolated along the other axis to give the final value of the point. To improve the replication of a natural looking world it is possible to layer Perlin noise on top of each other where each layer has subsequently lower amplitude which represent smaller and smaller details of the terrain [34].



(a) Five times layered Perlin noise

(b) Random noise

Figure 2.4: Different noise generated in Unity

2.2.2 Voronoi Diagrams

One method of spatial data interpolation of points into polygons, invented more than 100 years ago, is *Voronoi diagrams*. These polygons correspond to the area that is closer to its corresponding point than any other, see Figure 2.5. Voronoi diagrams have several applications, especially when it comes to geoinformatics, but also within computer graphics [35]. The polygons, known as Voronoi regions, are created by identifying where its sides are located. Each point on a side is located exactly in between the two (or more) closest points. In other words, the side can be found by looking at a ray cast radially from a point and when the ray is as close to any other point as the original, a side of the polygon is found. Mathematically a Voronoi region can be defined as: let S $\subseteq \mathbb{R}^2$ be a set of n points, then the Voronoi region of $p \in$ S is the set of points $x \in \mathbb{R}^2$ that are at the same distance or closer to p as any other point in S, see equation (2.1) [36].

$$Vp = \{x \in \mathbb{R}^2 : \|x - p\| \leq \|x - q\|, \forall q \in S\}. \tag{2.1}$$

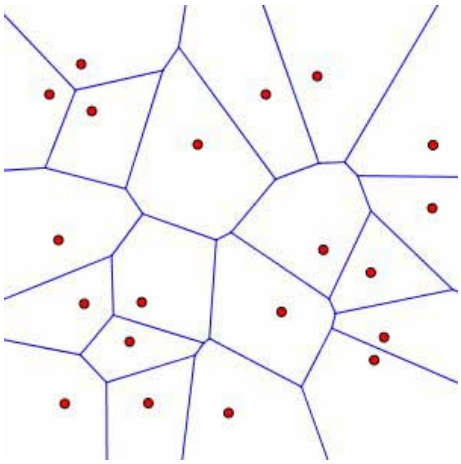
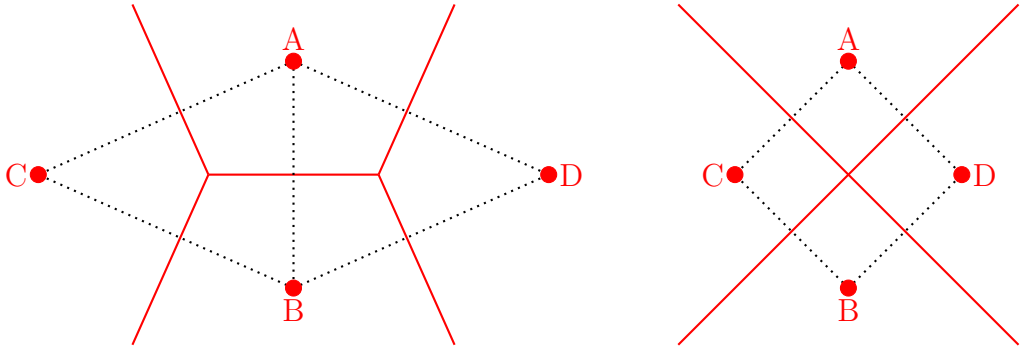


Figure 2.5: Voronoi diagram, points in red surrounded by their Voronoi regions. Source: [37]

As previously mentioned, Voronoi diagrams can be used in combination with other techniques such as Perlin noise when generating terrain. The combination helps break the monotonous patterns by having the Voronoi regions influence the noise map in a different way. For example, the noise could be superpositioned with a value dependent on how close the point is to the center of a region, creating a mountain like effect [36].

2.2.3 Delaunay Triangulation

A concept which is closely related to Voronoi diagrams is *Delaunay Triangulation*. This is because a Delaunay Triangulation can be found by drawing an edge between each set of points that share an edge between their Voronoi regions [36] [35]. As illustrated by Figure Figure 2.6a, when a line is drawn from each region to each bordering region a triangle is created. However in rare cases, four or more lines will meet at the same point as in Figure 2.6b. In this case when a line is drawn between each bordering region a quadrangle or a higher order polygon is created. It is normal to assume the absence of such cases although if encountered they are usually solved at the end of the process by either perturbation or case analysis. In Figure 2.7, a complete map of triangles (black) derived from a Voronoi diagram (red) can be seen. Delaunay Triangulation is a common concept within computational geometry and is especially useful when generating meshes for simulations of physical processes [36] [35].



(a) The most common case where three lines meet and all regions A, B and C connect to each other which form a triangle.

(b) A unusual case where four lines meet which causes A and B to not connect to each other but connect to C and D and a quadrangle forms instead of two triangles.

Figure 2.6: Different cases of Delaunay triangulation applied on a Voronoi diagram. The case in Figure 2.6a is the most common case where only three lines meet and the case in Figure 2.6b is a unusual case where four lines meet.

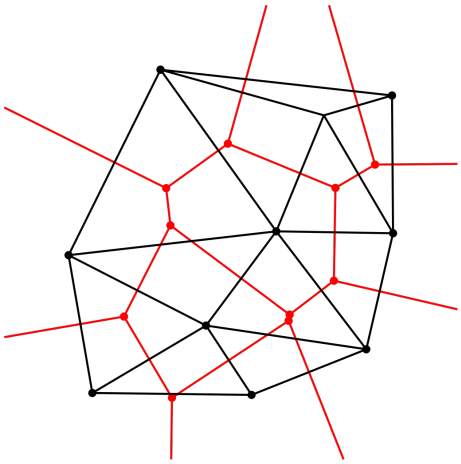


Figure 2.7: Delaunay edges in black and Voronoi edges in red.

Source: [38]

2.3 Genetic Algorithms

Genetic Algorithms (GAs) present a method of solving optimisation problems and they work by continuously modifying parameters until an acceptable solution is reached. The underlying theory behind GAs was first published by John Holland in *Outline for a Logical Theory of Adaptive Systems* (1962) [39]. GAs is one of

three main subsets of Evolutionary Algorithms with the other two being Evolutionary Strategies and Evolutionary Programming [40]. GAs are based on the theory of *evolution by natural selection* that occurs continuously in nature and was first proposed by Charles Darwin in his book "On the origin of species" in 1859 [41].

Darwin's theory of evolution states that organisms continuously evolve in nature and adapt to the environment that they live in. Organisms possess a set of *genes* that defines which features the organism should have. As the organism survives in its environment, it has the possibility of passing on its genes by reproducing. The genes of a successful organisms remain within the species, while the genes of unsuccessful organisms that die or does not manage to reproduce do not. This process is also known as *survival of the fittest* [41].

GAs are built around Darwin's theory and uses words and concepts similarly, such as *mutation*, *population* and *generation*. The fundamental idea of GAs is to continuously create better solutions to an optimisation problem by capitalising on solutions that works above average compared to others [42]. Frequent improvements through recombination and breeding by adjusting parameters are similar to how horse breeding works, where breeders try to breed fast horses together with the hope of getting an even faster foal.

In GAs, a solution is known as a *chromosome* which contains a set of *genes* [43]. An example of a simple chromosome of a human could for instance contain the genes *gender*, *eye colour*, *hair colour*, *muscularity* and *body height* and from these genes a human can be constructed. In this case, the human or the "realisation" of the chromosome is in GAs known as an *individual*. The genes in the chromosome can be represented in different ways. A gene could for example be seen as a binary number, indicating whether the gene is active or not, or as a real value, indicating the strength of that gene [43]. An example of a binary gene could be vision, which could be represented as a number within an interval, where a higher number would indicate a better and more accurate vision.

GAs contain a series of steps in order to improve parameters, shown in the flow chart diagram in Figure 2.8. The first step is to create the *initial population*, consisting of randomised individuals, thus creating a diverse population. The individuals are then evaluated based on how well they solved a certain optimisation problem or task. In nature, this task is to survive and reproduce, while it in GAs can be anything as long as a concrete value can be assigned to how well each individual in the population performed the task. Succeeding this evaluation, GAs undergo a series of steps which aims to create a new generation with a new population that in general performs better than the previous one. These steps are namely *selection*, *crossover* and *mutation* and they are explained more in depth in sections below. Finally, a new generation is created and the process repeats from the evaluation step. The process is terminated once a good enough solution has been found [43].

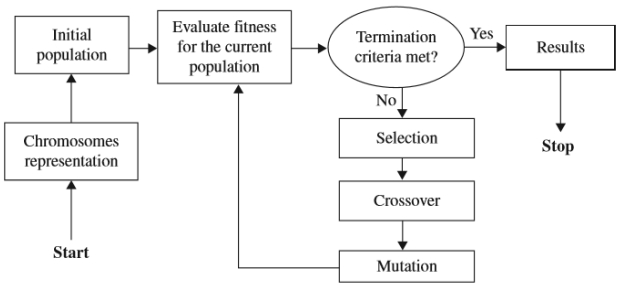


Figure 2.8: The steps of Genetic Algorithms

Source: [43]

2.3.1 Selection

In the selection phase a number of individuals are chosen to be the parents of the new generation. The goal of the selection operator is to select individuals that perform well at a given task as parents to the next generation. The selection operator selects individuals based on a value that reflects how well they performed, their *fitness*. This value is given by a *fitness function*. By evaluating the individual through this function, a number can be calculated to indicate how fit the underlying chromosome was [44].

Durin selection, it is necessary balance *exploitation* and *exploration* [44]. The selection algorithm should both exploit fit individuals, as well as exploring other solutions by selecting individuals with with lower fitness. Thus creating diversity among individuals, potentially improving overall performance of the species by exploring other local maxima. For an example, assume two approaches to human movement, crawling and running. As shown in Figure 2.9 the fastest possible runner is faster than the fastest crawler. If an entire population is crawling, it becomes crucial for genes to explore other options to reach the global maximum, which is running, although the fitness of the gene might decrease initially.

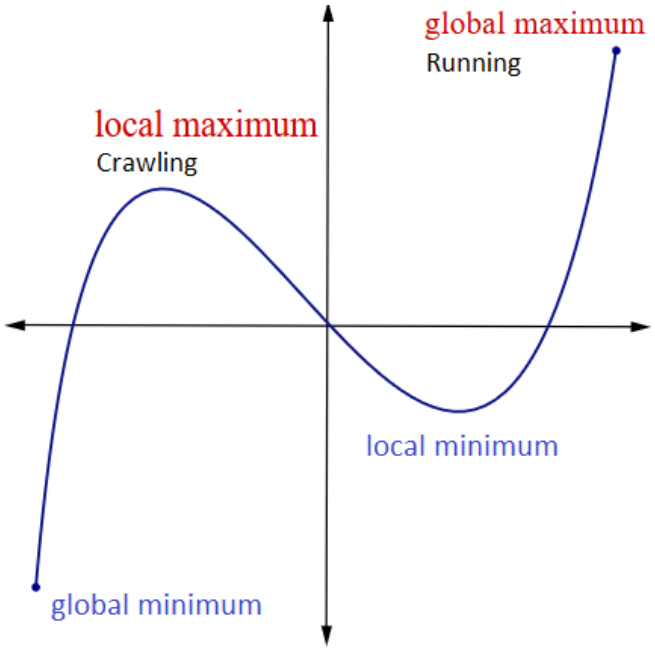


Figure 2.9: An evaluation of fitness based on movement techniques. The X-axis represents the technique used, the Y-axis the fitness of the gene

Source: Adapted from [45]

A common selection method is *Roulette Wheel Selection* which gives each individual in the population a probability of being selected proportional to that individual's fitness. Thus rendering individuals with higher fitness more likely to be selected than worse performing ones. The probability of an individual i being selected is given in equation (2.2), where n denotes the total number of individuals in the population and where $f(i)$ denotes the fitness of an individual [44].

$$p(i) = \frac{f(i)}{\sum_{j=1}^{n} f(j)} \tag{2.2}$$

Another example of a selection method is *Elitism*, in which a select amount of the best performing individuals are directly transferred to the new generation without getting altered. Elitism prevents altering of successful genes, from potentially being destroyed by further mutation and crossover [44].

2.3.2 Crossover

The crossover phase aims to decide what specific genes are to be carried over from each of the parents to the child [46]. Simple crossover methods include *single point crossover*, where a crossover point is selected and the child's genes consist of continuous parts from either side of the crossover as can be seen in Figure 2.10. Another crossover method is *uniform crossover* in which the probability of getting parent

A's or parent B's gene is 0.5. This means that the child's chromosome will consist of a combination of the parents genes, shown in Figure 2.11.

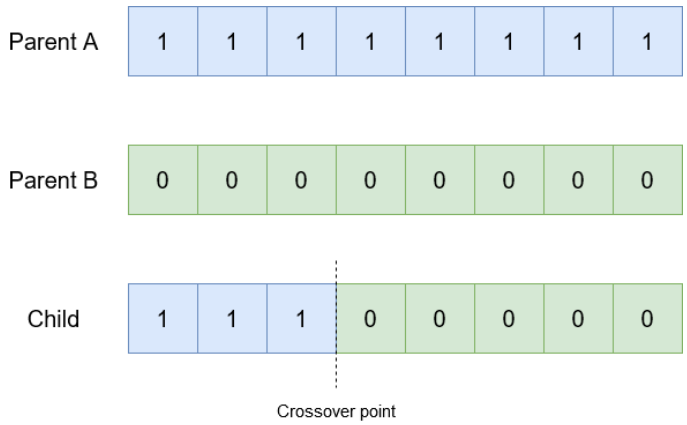


Figure 2.10: Single Point Crossover

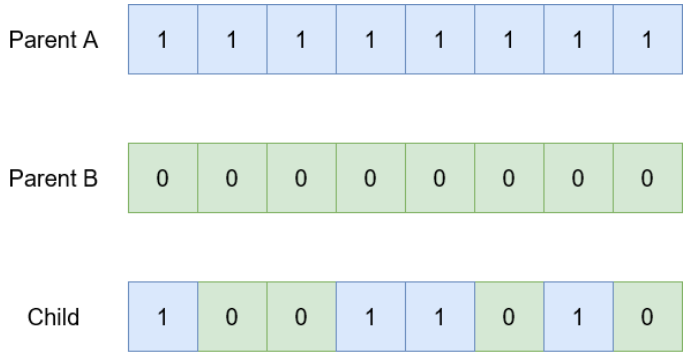


Figure 2.11: Uniform Crossover

A slightly more complex method is *BLX-α* which acts both as a crossover method and a mutation method. Each gene in the child is randomized uniformly inside an interval that is calculated by taking the range between the parents' genes and then expanding it by the difference between the two. The range is increased by multiplying the difference with a positive real number α. A property of *BLX-α* is that the range will become relatively small for parents whose genes are similar [43].

2.3.3 Mutation

While the crossover phase aims to create new constellations of already existing genes, the mutation phase aims to mutate and change them further. The purpose of mutation in GAs is to prevent the solutions from getting stuck in local maximums by further mutating genes and thus exploring new solutions [43].

The probability for a mutation to happen is called the *mutation chance* and research show that this value needs to be chosen carefully. GAs with high mutation chances

behaves similarily to a random search that does not converge. Those with low mutation chance can result in premature convergence at local maximums or minimums [47]. Research show that values such as 0.001 performs well. [48].

If the genes in the chromosome are represented as binary digits then a simple bit flip can be used as a mutation operator, seen in Figure 2.12. If the values are however represented as real numbers then other methods have to be used. One of these methods is Gaussian mutation, where the gene's new value is chosen from a Gaussian distribution which can give any real number. As opposed to *BLX-α*, Gaussian mutation has the same distribution in the range of possible values for each gene regardless of what the gene is for [49].

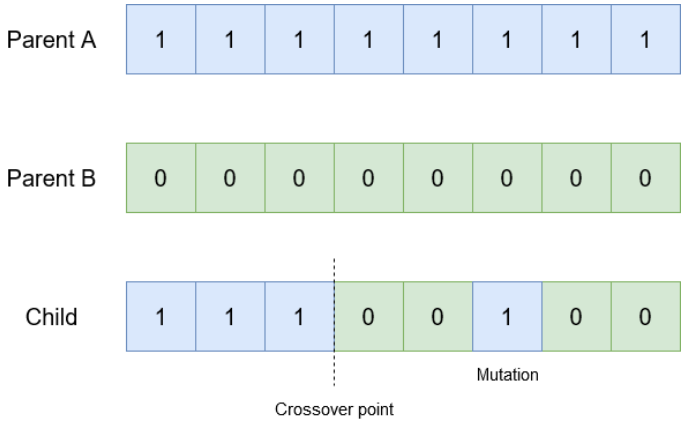


Figure 2.12: Bitflip Mutation.

3

Method

This section presents the design and implementation of the four main features needed to investigate the possibility of terrain dependent behaviour. These are dynamic terrain, evolutionary mechanisms, animals and a behavioural model implemented as an FCM. These features were the foundation of an ecosystem in which multiple simulations were performed to collect data of how the animals evolved.

3.1 Simulations of the Ecosystem

The following section details how the simulations were performed. In the first section, the goal and conditions of the simulations are presented. The second section show the environment, detailing the terrain, vegetation and the presence of predators. In the third section a fitness function is provided. The usage of the fitness function, the implementation of mutation and the evolution of genes are presented in the final section.

3.1.1 The goal of the simulation

The goal of the simulations was to see if the animals would evolve and adapt to their current environment, and if different environments would create animals with noticeably different behaviours. In addition to this, the frames per second (FPS) and load on the CPU was measured during the simulations in order to see whether the program and solutions are feasible to use in games. The simulations were performed on a HP computer with Intel Core i5-6300HQ CPU, NVIDIA GeForce GTX 960M Graphics Card and 16GB RAM.

Each simulation was run with the goal of reaching a point where the ecosystem was completely sustainable in its simulated environment, which entails preventing the rabbit-species from becoming extinct. This goal was set as rabbits had evolve a versatile behaviour incorporating reproduction, eating and drinking, in order to make the ecosystem thrive,. Escaping was not essential for survival but it would allow rabbits to avoid getting eaten. The simulation would reset if no animals were alive so multiple rounds were run allowing animals to evolve and reach self-sustainability. A sustainable species was defined as being extant one hour in-game time after the population was introduced, meaning that the species had to survive

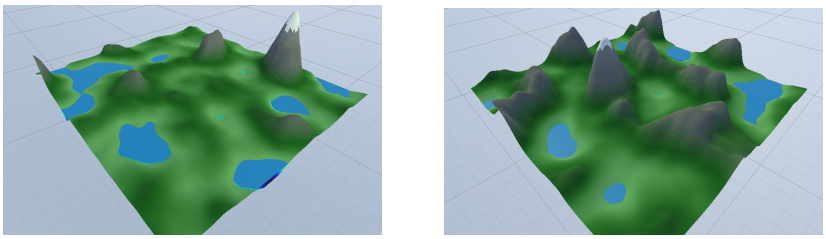
for at least 14 generations, as an individual lives for a maximum of 275 seconds. One hour of life does not guarantee that the species will never die out, but it served as a good indication that the animals were highly adapted to their environment. The simulation would be terminated early if the animals could not reach this point. This termination point varied between the simulations but generally did not exceed 50 hours in game time (10 hours real-time)

3.1.2 The environments in the simulation

Three main simulations were set up with the goal of investigating whether or not Fuzzy Cognitive Maps could be used to obtain environment-dependent behaviour. Two vastly different environments generated by PCG were categorised as: `Mountains` - with low vegetation, few water resources and hills restricting the sight of the animals - and `Plains` - with high vegetation and an abundance of water and very few hills. The two environments can be seen in Figures 3.1a and 3.1b. These two environments differ in the main aspects for the survival of rabbits: Accessibility to food and the presence of water. Additionally, the two environments were simulated as either `Hostile` or `Peaceful`, allowing for analysis of the presence of predators for the different environments. The hostile scenario added three foxes to the environment, whose only objective was to kill rabbits. Foxes could not die and were slower than the rabbits, so that the rabbits could escape them if they chose to. Collectively there were three unique simulations:

1. `Mountains, Peaceful (MP)`
2. `Plains, Peaceful (PP)`
3. `Plains, Hostile (PH)`

Each simulation round was always initialized with 20 rabbits. The amount of plants differed between simulations, for example one ecosystem was simulated in `Plains Peaceful` with 20 plants and was therefore named: (`PP20`). Other configurations that were simulated were `PP20`, `PP40`, `MP20`, and `PH20`. Two instances of `PH40` were simulated, denoted `PH40-1` and `PH40-2`. This was done to compare if these two would evolve in a similar way.



(a) `Plains` (b) `Mountain`

Figure 3.1: The two environments used in the simulation

3.1.3 Determining the fitness of an individual

The fitness of an individual i was given a numerical value according to equation (3.1) below. The fitness of each individual was adjusted before the selection phase by subtracting the fitness with that of the least fit individual in the generation. Thus rendering the fitness of the least fit individual to be 0. The fitness function is quadratic, and was designed in such a way as to more clearly separate well performing individuals from worse performing ones.

$$f(i) = (TimeOfDeath(i) - TimeAtRoundStart)^2 \tag{3.1}$$

Instead of measuring fitness via the total time spent alive, fitness was based on the time elapsed since the round started, thus measuring the fitness of a gene carried down through generations. The function was defined this way in the belief that this would encourage breeding among animals, as breeding would present a way for an individuals genes to live on beyond the individuals death, which is fundamental for living species. This approach was taken rather than giving a concrete reward for breeding, which would have given the animals an incentive to breed even if it did not contribute to the survival of the species. By making the function this general, the animals will only evolve to reproduce if it is beneficial for the species.

3.1.4 Selection, crossover and mutation of individuals between generations

Between each generation, alterations in the form of selection, crossover and mutation of individuals occurred with the aim of improving the overall fitness of the population. Naturally, some alterations occurred throughout the round as the population reproduces through natural selection, however the majority of the alternations were between the rounds.

Each generation had a population size of 100 individuals, with the initial population consisting of rabbits which had FCMs with randomised weights between -1 and 1, as to maximise the diversity in the population. The population size of 100 was picked since smaller population sizes often led to the solutions converging to local maximums quickly, and larger ones led to the simulations taking too long to run. A larger population size would increase the simulation time because each generation was split into different rounds with 20 rabbits each. For 100 individuals this would equal five rounds. This was done for two main reasons, the first being that a low round population would reduce the lag of the simulation and the second one being that the food supply was not large enough to sustain a high round population.

Roulette Wheel Selection was used as the main selection method with the slight modification that the best performing individual would always be picked as a parent in order to fully capitalise on good genetic material. Uniform crossover was picked as the crossover operator and Gaussian mutation was selected as the mutation operator throughout simulations. These two were selected instead of BLX-α since they would

let mutations have a significant impact even if all parents had the same genes. The Gaussian mutation had a standard deviation of 1 and this value was chosen as a result of trial and error. The mutation chance was set to 0.05 and this value was experimented with and our simulations showed that a lower chance of around 0.01 would usually result in the solutions getting stuck in local maximums and where a higher chance would mutate too often, thus destroying good genes.

Elitism was also used to directly copy the two best individuals of each generation over to the next one, in order to guarantee conservation of good genetic material.

Natural selection was implemented through breeding and used the same parameters as the Genetic Algorithm except the selection operator since the parents are the mother and the father that reproduced.

Ideally the values of the variables used in the evolution such as standard deviation and mutation chance should be based in the values that R Gras et al. chose for their simulations, but unfortunately they did not disclose their values used. Instead, the variables were picked through a process of trial and error, and the chosen values seemed to yield good results in comparison to other values.

3.2 The terrain

The environment was implemented as a continuous grid where entities could move around freely. This approach was taken rather than a discrete model such as a CA since many video games today allow for free movement rather than discrete states. Furthermore, continuous worlds require more complicated calculations in regards to navigation which means it was important to have a similar setup in this project in order to get accurate performance metrics.

Perlin noise was used to create the terrain as it was simple to implement and translates into an earth-like surface. Furthermore Unity had a function that generated Perlin noise which allowed for the generation of unique terrain via different seeds. The characteristics of the map could easily be altered by changing various parameters of the noise which resulted in changes such as making mountains and ponds more frequent as well as wider and taller/deeper. The simulated world consisted of three main parts: the ground, water and plants. The ground consisted of a square mesh generated with Perlin noise. The value of the generated noise map decided the elevations of each point on the mesh which created mountains and valleys. To add water a certain height was specified corresponding to sea level and water was generated at this elevation. The water consisted of a mesh that was generated with the help of Delaunay triangulation. Plants were then placed in random locations that were not below sea level nor high up enough to be on an inaccessible mountain. The different parts of the world were colored using several gradients to represent the height map. For instance, a high elevation would be coloured white to represent snow and a medium elevation would be coloured green to represent grass.

3.3 The animals

Two different species of animals were implemented, namely rabbits and foxes. An animal belonged to a species and was recognized as such by other animals. The foxes were predators whose diet consisted solely of the rabbits, their prey. The rabbits ate the plants that were placed throughout the world. Multiple different evolvable traits were originally implemented in early simulations, such as speed, size and fertility. However, having all these evolve simultaneously made it hard to study the evolution of the FCMs specifically. Therefore all traits were made constant so that the weights in the FCMs were to only genes subject to evolution.

3.3.1 Senses and Memory

The animals used sight and smell to perceive the environment and classified entities in their surroundings as either food, mates, water or foes. Sight was implemented by ray-casting in a cone shape facing the same direction as the animal. Sight could be blocked by terrain, animals or plants, making the environment a factor in how well animals could sense their surroundings. Smell was a sphere around the animal and could not be blocked by any obstacle. The senses recorded a `SensedEvent` that encapsulated two aspects: the number of each type of entity nearby and their proximity. The number of occurrences and proximities were used in the animal's FCM as explained in section 3.4.

Each sensed entity was given an urgency which was determined based on what relation the animal had to the sensed entity. The urgency of a foe was determined by a combination of the distance to the foe and its speed. The urgency of plants and water resources were determined based on proximity and size. Shown below in equation (3.2) is a function of how the rabbit evaluated the urgency of foxes by estimating time of an eventual chase. The quotient t_{chase} is calculated by the dividend d, the distance between the rabbit and the foe, and the divisor which is the difference between v_r and v_f, denoting the speeds of the rabbit and fox respectively. The lowest value of $t_{chase} \leq 0$ was the most urgent foe. In cases when the foe was faster than the rabbit, the closest foe was deemed to be the most urgent.

$$t_{chase} = \frac{d}{v_f - v_r} \tag{3.2}$$

3.3.2 Mating

Mating was implemented as a means of natural selection. When two animals mated, the children would instantly be created at the spot where the mating animals stood. The resources (food and water levels) of the two animals that reproduced would be combined and distributed equally amongst themselves and their children. This ensured that the total amount of resources did not increase which would otherwise mean that the populations could thrive in environments without water or food simply by mating.

3.4 The Fuzzy Cognitive Map

Fuzzy cognitive maps were used to model behaviour as opposed to using decision trees. Decision trees would require data of how the animals should behave in order to model the behaviour itself, meaning the behaviour would not be able to emerge automatically from scratch simply by running simulations. FCMs on the other hand had already been show to work as a behaviour model in an ecosystem.

The implementation and use of the FCM was highly inspired by the work of R Gras et al. and their paper *An individual-based evolving predator-prey ecosystem simulation using Fuzzy Cognitive Map as behavior model* [13]. Their simulation showed according to them, great promise and R Gras et al. writes "*Agents make pertinent decisions of action considering several independent sensitive pieces of information, the evolutionary system generates and selects behavioral models with new innovative notions, the correlation between the number of species and the number of individuals is in accord with such existing correlation*". Their success led us to use similar concepts and equations in the hope of reaching a similar results. It needs to be mentioned that R Gras et al. used a purely mathematical and non-visual simulation, whilst we were attempting to run the simulation in Unity. However, our FCM receives inputs only in the form of fuzzy variables, allowing it to be decoupled from Unity and used by game developers using any environment.

There were three different types of concepts in the FCM.

- Causal Concepts - also known as inputs, such as the presence of foes, the proximity to food and their thirst.
- Causal and Affected Concepts - also known as perceived states, such as the animals perceived sense of hunger, thirst and fear.
- Affected Concepts - also known as actions, such as the act of searching for food, mating or drinking.

A majority of the concepts used were based on R Gras et al.:s concepts.

3.4.1 Inputs - Causal concepts

In order for the inputs to be able to have both positive and negative effect on affected concepts, each had a complement that achieved that effect. For example, `Hungry` were paired with `NotHungry`, as they do not necessarily have proportionally inverse correlation to affected concepts. There were two different kinds of inputs, the internal states of the animal and its surroundings, both needed to be prepared before use within the FCM.

The internal states of the animal were all represented as values between 0 and 1 and were therefor not subject to fuzzification. For an internal state with a value a its input A was simply $A = a$ and its complement $A_c = 1 - a$.

The animals surroundings were prepared via fuzzification. An animal sensed its surroundings using sight and smell and thereafter transformed the sensed surroundings into the fuzzy input variables. The FCM evaluated two aspects of the sensed surroundings, the presence and urgency of each type of entity. The presence was the number of occurrences in the surroundings and the urgency the value of the most urgent entity. Urgency was translated into inputs such as `FoodClose`, `FoodFar`, `WaterClose`, `WaterFar`, `MateClose`, `MateFar`, `FoeClose` and `FoeFar`. Presence is translated into inputs such as `FoodPresenceHigh`, `FoodPresenceLow`, `WaterPresenceHigh`, `WaterPresenceLow`, `MatePresenceHigh`, `MatePresenceLow`, `FoePresenceHigh` and `FoePresenceLow`. Where each aspect having `Close` was paired with a complement `Far` and `High` paired with `Low`.

The fuzzification of senses is discussed more in depth in section 3.5.

3.4.2 Perceived states

The percieved states were both causal and affected concepts, which are to be regarded as the processing center of the FCM. The concepts were `pHunger`, `pFear`, `pThirst` and `pHeat` which all affect each other, for example, fear could inhibit hunger whilst hunger could strengthen the sense of thirst. These values were contextualised concepts as they were updated based on their previous values, the aforementioned inputs and each other. Meaning, that a mildly varying surrounding would not necessarily affect the animals, since their intent was clear, whilst simultaneously opening up for rapid changes of action if the surroundings were to change substantially. Examples of sudden changes could be that food was eaten by someone else, or that foes were approaching. It is important to differentiate the perceived hunger, `pHunger`, from the input `Hunger`. The perceived hunger was to what extent the animal could prioritize the actual hunger with regards to other perceived statuses. For example: A frightened animal would sometimes not act on its hunger, since the perceived fear was higher than the perceived hunger.

3.4.3 Actions

The actions were purely affected concepts and were only affected by the perceived senses. The actions were `GoingToFood`, `GoingToWater`, `SearchingForMate` and `Escaping`. The FCM was isolated in the sense that it needed to be inferred in order for the animal to know what action to perform. Inference of the FCM meant returning the action concept with the highest value when the FCM was queried for an action. If multiple actions attained the same value and thus creating a tie, the action receiving the highest value have they not been bound by [-1, 1], was chosen. Thus all decision-making was centralized and isolated in the FCM.

3.5 Fuzzification of senses

This section presents how senses were fuzzified and prepared for the FCM. As the FCM was purely mathematical there needed to be a way of determining the strength

and importance of surroundings, which was done via fuzzification.

3.5.1 Fuzzyfying proximity of an entity

The proximity of an entity was determined by fuzzifying the distance using a sigmoid function shown in equation (3.3) which is is plotted below in Figure 3.2.

$$\frac{1}{1 + e^{(\frac{|distance|}{4} - 4)}} \tag{3.3}$$

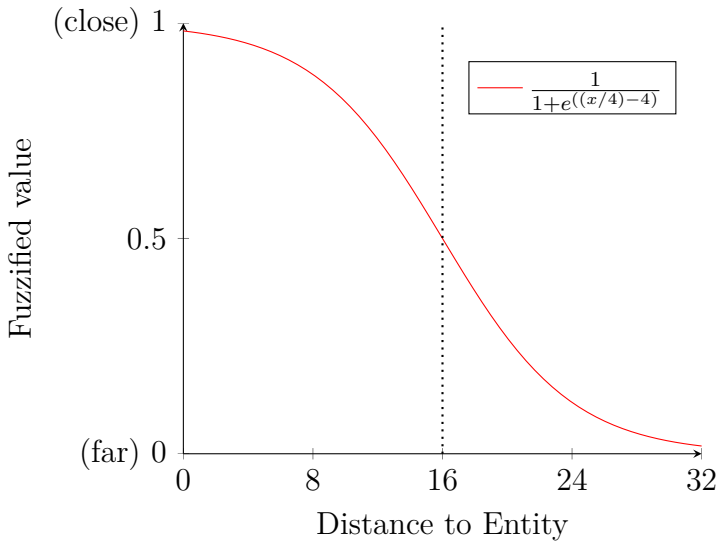


Figure 3.2: Showing our sigmoid membership function for fuzzyfying distances as close or far.

A sigmoid function were used as a membership function for determining the fuzzification of distance as close or far, as it is common in ecological modelling [32]. Large distances converge towards 0 and small distance converge towards 1. The rate of change was highest when the distance was around 16 units, since it was half of the animals sensing radius and because it was when the distinction between close and far was made.

3.5.2 Fuzzyfying presence of entities

The `Presence` category determines fuzzified values for the number of different entities that are being sensed by the animal at a given time. The inputs are: `FoodPresenceHigh`, `FoodPresenceLow`, `WaterPresenceHigh`, `WaterPresenceLow`, `MatePresenceHigh`, `MatePresenceLow`, `FoePresenceHigh` and `FoePresenceLow`. They were fuzzified using the logarithmic function (3.4) shown in Figure 3.3.

$$Log_7(\#entity + 1) \tag{3.4}$$

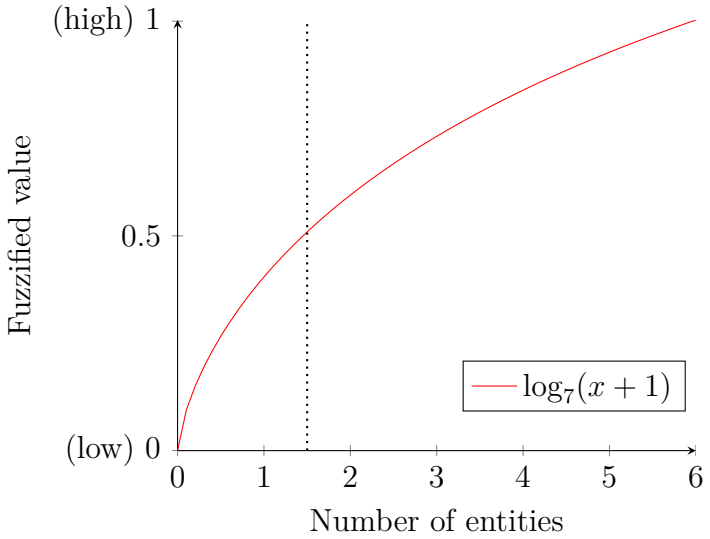


Figure 3.3: Showing our selected logarithmic function for fuzzyfying presence. One or less entities is a low presence, more than one is a high presence.

4

Results

Observations and numeric data gathered from the simulations are presented in this section. The results are separated into three main sections. These are the rabbits' survivability which encapsulated their causes of death, the amount of time the species managed to survive and the population sizes. The other sections show changes to the rabbits' behaviours and the simulations' performance in terms of frame rate.

4.1 Evolution of Rabbits

In this section, results are presented relating to the evolution of the rabbits. The theories and interpretations of the results are then discussed further in the discussion section. Note that the last data point in the round duration graphs has been removed in the peaceful simulations. This was done to improve the readability of the data points for the preceding rounds, since the last round would span several hours.

4.1.1 Changes to the population

Overall survivability of the rabbits clearly showed a positive trend between generations. In the peaceful simulations, the rabbits quickly managed to evolve and eventually died of old age rather than by hunger or thirst, a cause frequent within the first two generations. Figure 4.1 shows the fraction of rabbits that died of age each round in `PP40`. Viewing the graph, occurrences of animals dying of old age was found after round 15. After 175 rounds the rabbits reached a point of self-sustainability and approximately 35% of the rabbits died of old age.

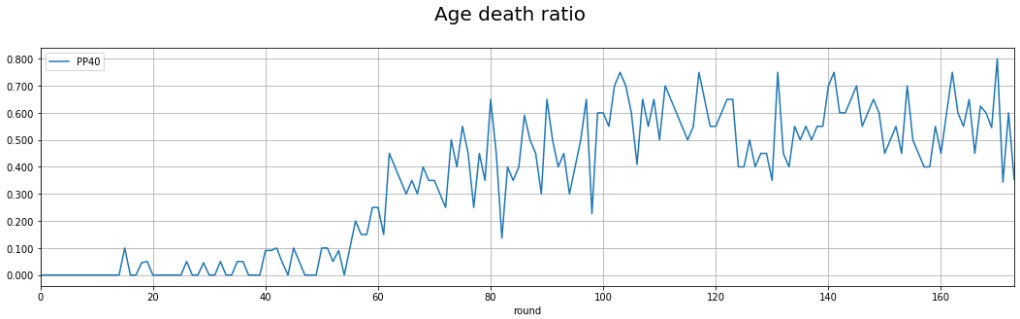


Figure 4.1: Age / death ratio for every round in simulation `PP40`

In simulation `MP20` the ratio of deaths due to age increased, peaking at round 47, only to later decrease at round 50, as seen in Figure 4.2. In the final round, rabbits were able to thrive for over 20 hours. Figure 4.3 shows that the round time increased over time.

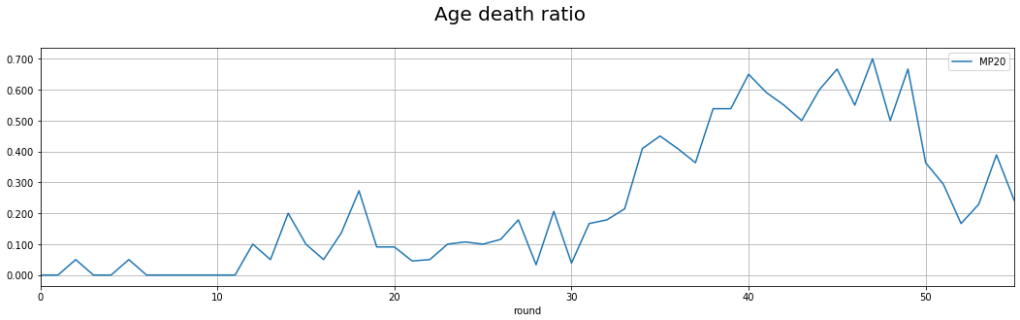


Figure 4.2: Age / death ratio for every round in simulation `MP20`

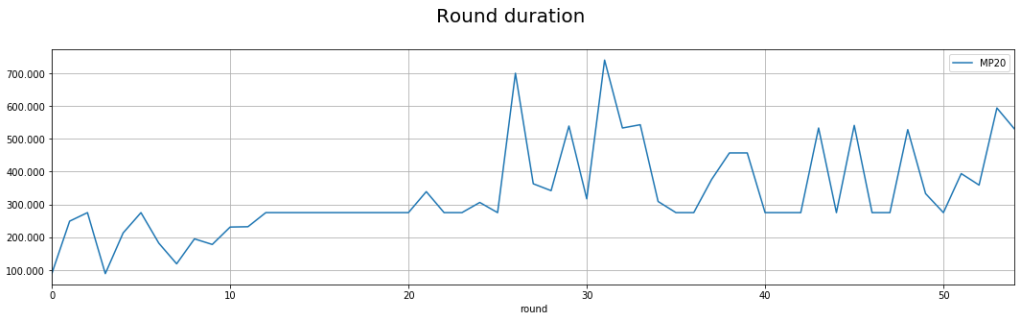


Figure 4.3: Round duration of `MP20`

The introduction of foxes in hostile simulations decreased the ratios of deaths due to age. Figures 4.4, 4.6 and 4.8 show the ratios of deaths due to age for the simulations `PH40-1`, `PH40-2` and `PH20`. Figures 4.5, 4.7 and 4.9 show the duration of each round. Note that the Y-axes contain different values for the simulations.

The rabbits managed to live until old age at round 550, round 380 and round 52 in simulation `PH40-1`, `PH40-2` and `PH20` respectively. In PH40-1, no increase to

the amount of rabbits dying of old age can be seen until round 550. After round 550, rabbits continually improve and frequently reach a round duration of over 400 seconds, and on two occasions manage to survive for around 1000 seconds. The average time decrease towards the end for PH40-1 but the frequency of long lasting rounds increased.

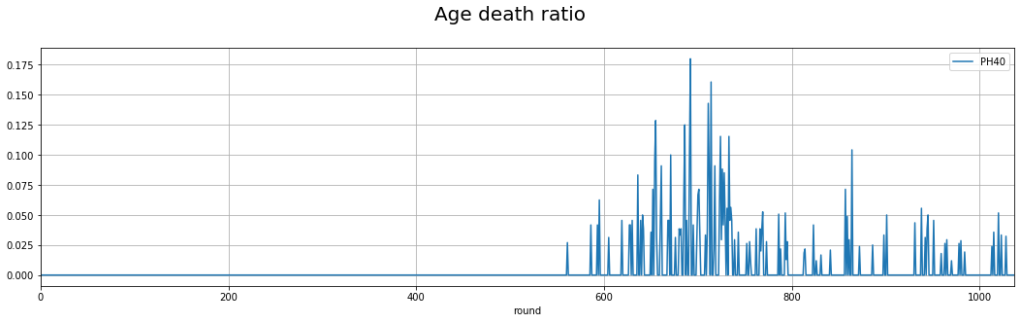


Figure 4.4: Age / death ratio for every round in simulation `PH40-1`

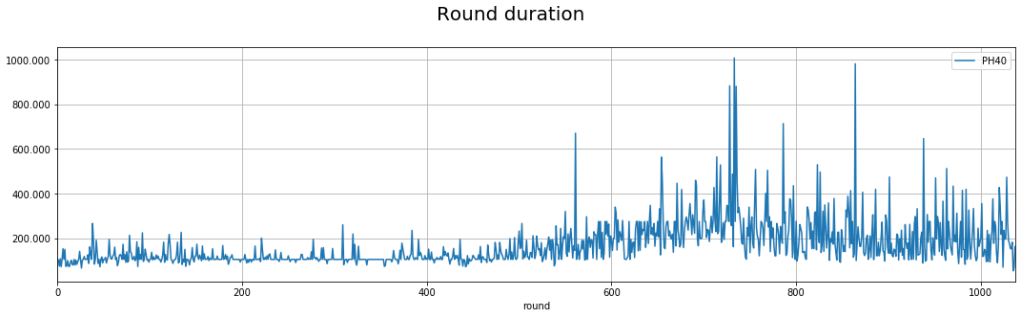


Figure 4.5: Round durations of `PH40-1`

PH40-2 saw a similar improvement curve and towards the end of the simulation the rabbits almost always managed to stay alive until old age. But the rabbits of PH40-2 never managed to get past 275 second round time (age limit), although data tells us that around three children were born each round in the end of the simulation. Figure 4.7 shows that the initial round durations are around 105 seconds long, which corresponds to the amount of time rabbits could survive without drinking water.

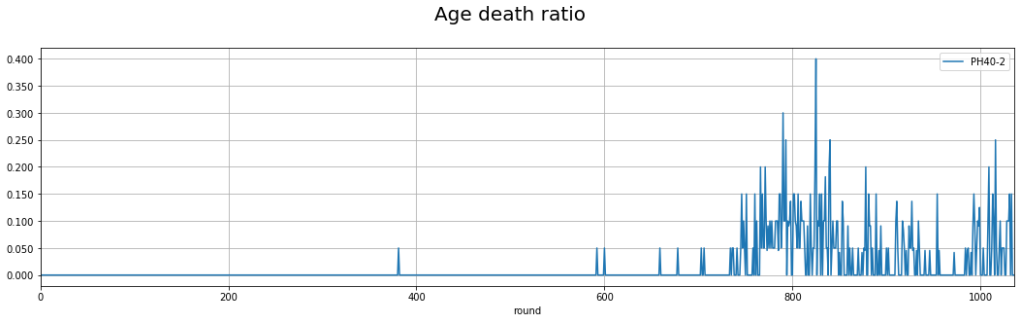


Figure 4.6: Age / death ratio for every round in simulation `PH40-2`

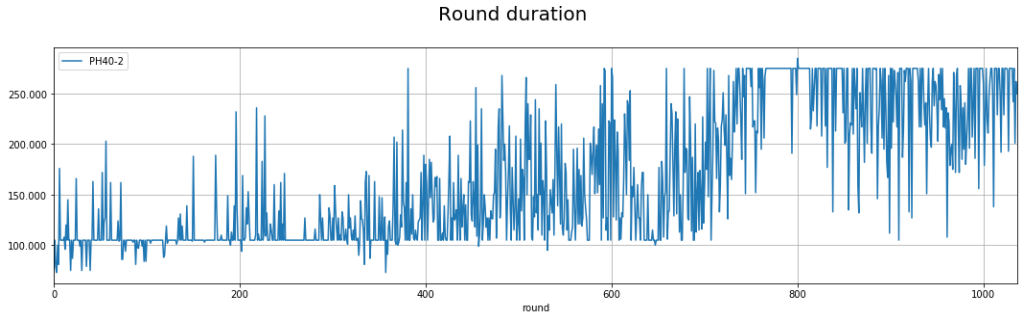


Figure 4.7: Round durations of `PH40-2`

Shown in Figure 4.8, the rabbits in `PH20` initially reached a high ratio of deaths due to age which declined down to 10% over time, similarly to `PH40-2`. But rabbits were able to thrive significantly longer in this simulation compared to the other hostile simulations, lasting over 10 hours in the last rounds, seen in Figure 4.9. Note however that the rabbits went extinct eventually, even after thriving for hours, indicated by the fact that there are multiple spikes at the end of the graph.

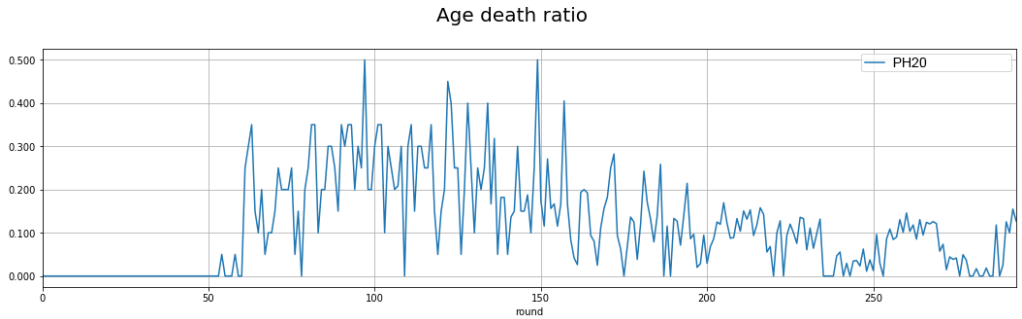


Figure 4.8: Age / death ratio for every round in simulation `PH20`

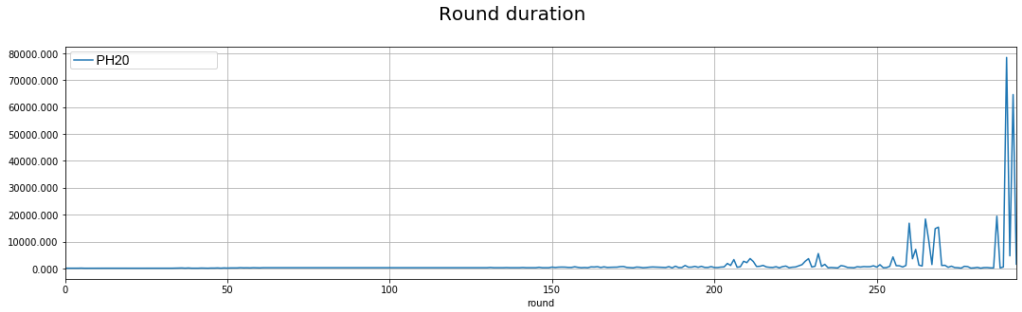


Figure 4.9: Round durations of `PH20`

Food importance

This section shows the impact which the amount of available food had on the rabbits. The stable populations of PP40 and PP20 are shown in 4.10 and 4.11 and they prove that the amount of food has an effect on population size. The population size of PP40

varied between 80-130 rabbits once they had reached the point of self-sustainability and the population change for PP20 varied between 50 and 70.

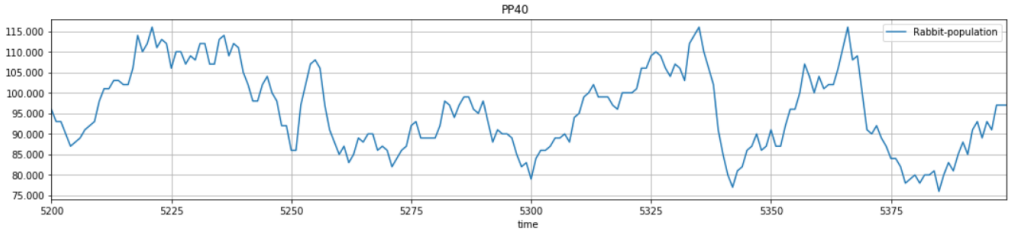


Figure 4.10: Population change for `PP40`

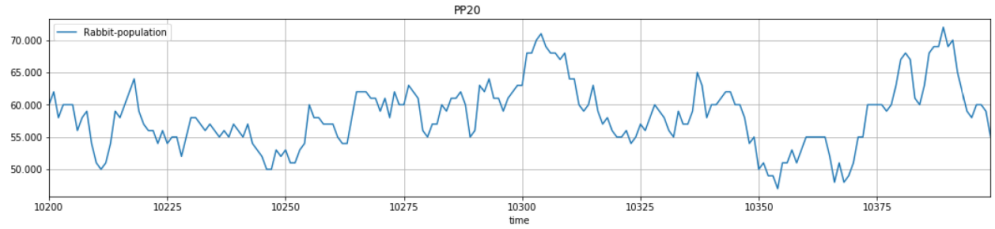


Figure 4.11: Population change for `PP20`

4.1.2 Evolution of the rabbits' behaviours

The figures in this section show graphs of the distributions of the actions performed by the rabbits. By comparing graphs, behavioural patterns and differences between simulations can be identified. The last data point of each graph represents the last round, which for the peaceful simulations corresponds to a long lasting round in which the rabbits had reached the point of self-sustainability.

The distribution of actions performed by the rabbits in `PP40` are illustrated in Figure 4.12. By looking at the graph, it can be seen that at the point of self-sustainability, the rabbits performed the `GoingToFood` and `GoingToWater` action 60% and 30% of the time respectively. The action `SearchingForMate` increases at the last data point which is expected since this action is essential for the rabbits to self-sustain.

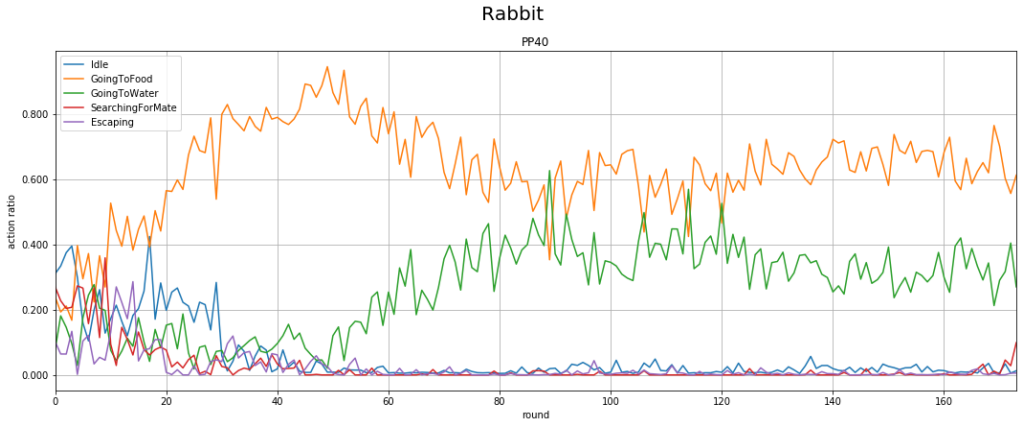


Figure 4.12: The average usage of every action in each round during simulation PP40

The action trends for simulation `MP20` is shown in Figure 4.13. Note that the very last data point is the most significant as the last round was the only one where the rabbits reached the point of self-sustainability. The distributions of the actions performed were: `GoingToFood` 70%, `GoingToWater` 20%, `SearchingForMate` 10% and `Escaping` 2%. This distribution shows that rabbits in `MP20` spent more time looking for food, less time looking for water and about as much time looking for mates compared to `PP40`.

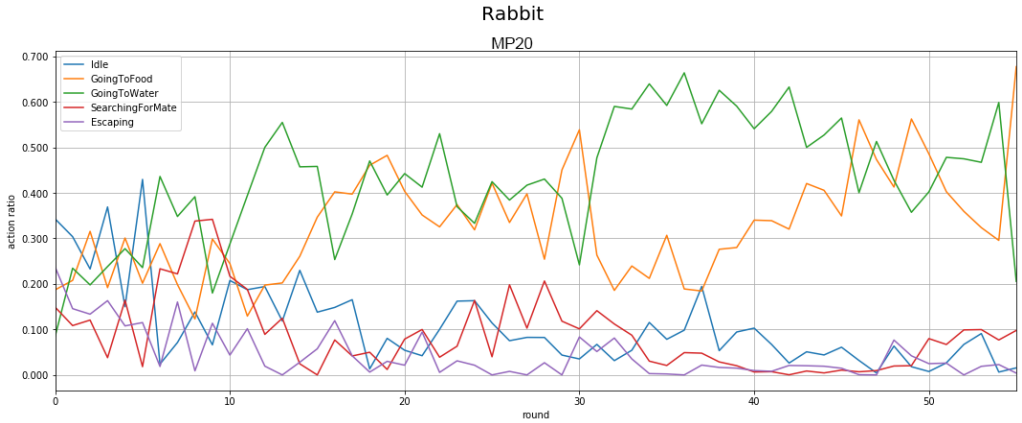


Figure 4.13: The average usage of every action in each round during simulation `MP20`

The distribution of the actions performed by rabbits in the hostile simulation `PH40-1` (shown in Figure 4.14) differ from the peaceful `PP40` (shown in Figure 4.12), as rabbits escaped more frequently throughout the evolution. Furthermore, by comparing the distributions of actions in the two `PH40` simulations with `PP40`, less fluctuating behaviour is exemplified.

By viewing Figure 4.14, it can be seen that the `Escaping` action is frequent until round 500, where the act of `GoingToFood` is performed more often. However,

the rabbits in the final rounds of `PH40-1` did not perform the `Escape` action. The frequency of `GoingToWater` is low until round 600, as preceding rounds had dehydration as a common cause of death. As `GoingToWater` became more prevalent so did `GoingToFood` and they got a similar distribution as the one found in `PP40`. Another point of interest is the `SearchingForMate` action, as the frequency was volatile during the simulation but reaches 10% at the end.

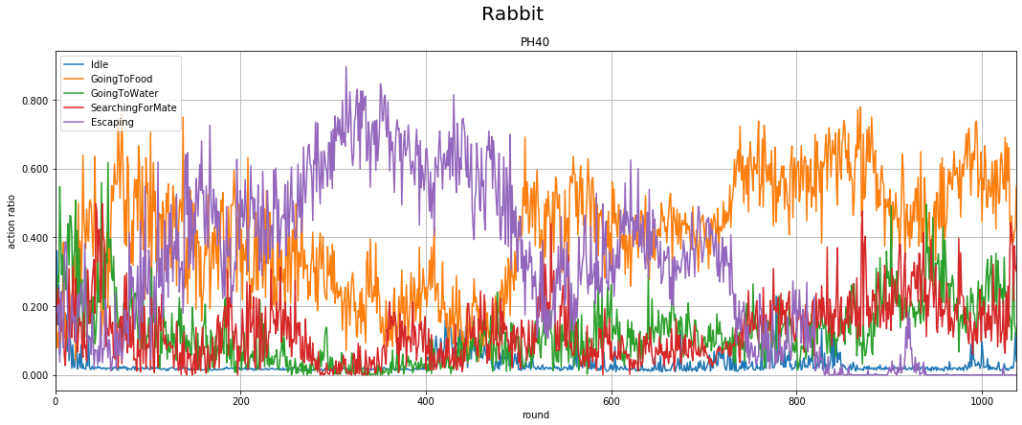


Figure 4.14: The average usage of every action in each round during simulation `PH40-1`

The evolution in PH40-2 saw a slightly different progression and the distribution of its actions can be seen in Figure 4.15. In the hostile simulation, the `Escape` action is initially frequently used. However, it then sees a continuous decrease throughout the simulation. The `GoToFood` action oscillates back a forth but increases towards the middle and end of the simulation. The action `GoToWater` becomes more frequent first at round 250 and evolves to be used approximately 33% of the time ultimately. `GoToWater` and `GoToFood` have similar distributions as `PP40` and `PH40-1`. The `SearchingForMate` action does not manifest itself in this simulation, although it spikes near round 850.

.

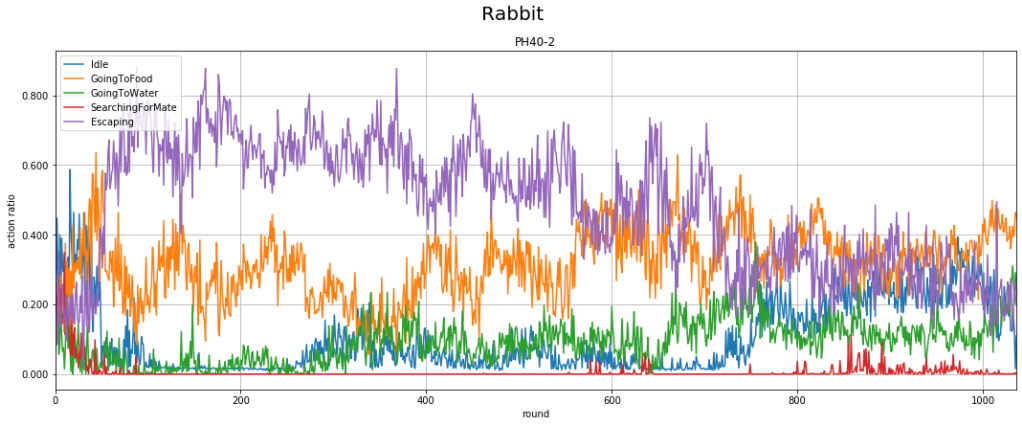


Figure 4.15: The average usage of every action in each round during `PH40-2`

Figure 4.16 shows the behaviour development of simulation PH20. The usage of the `Escape` action declined over time but was not completely phased out, similarly to PH40-2. Note however that the action was phased out in some specific rounds, as seen in Figure 4.17 after *time* = 15000. After a while the graph bumps back up to around 20%, meaning all rabbits died and a new round was started. In the end the usage of `Escape` was around 20%, `GoToFood` was 45%, `GoToWater` was 20% and `SearchingForMate` was 15%.

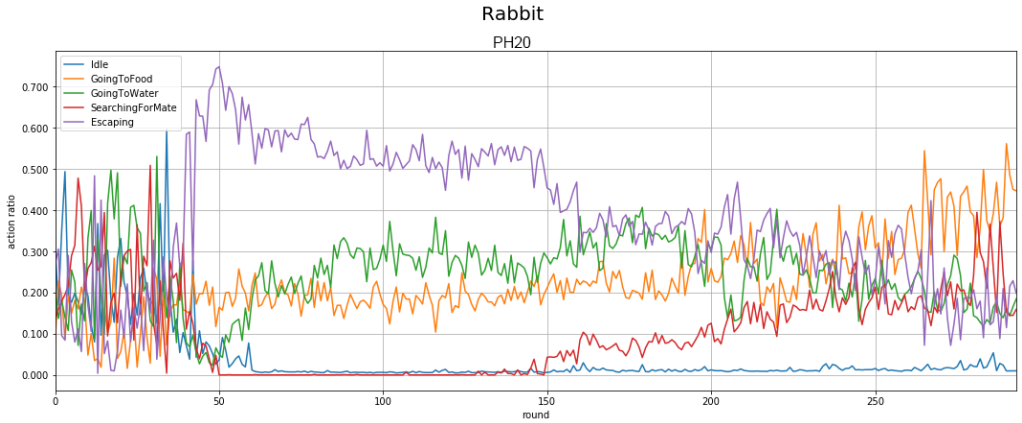


Figure 4.16: The average usage of every action in each round during `PH20`

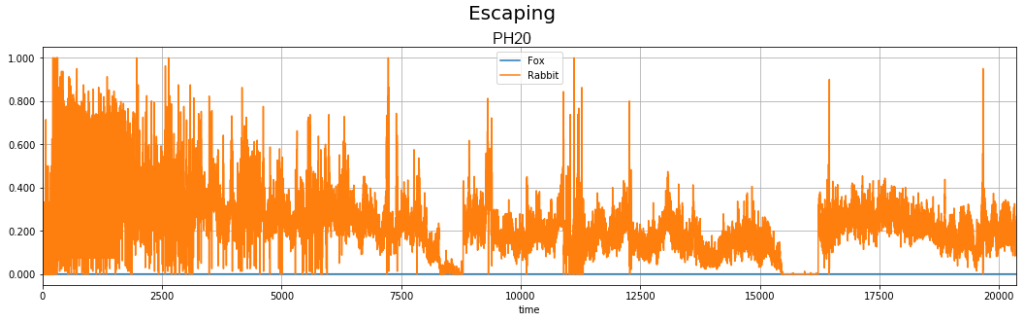


Figure 4.17: The average use of the `Escaping` action across 20 second intervals in simulation `PH20`

4.2 Computer processing load

The following table 4.1 shows the performance of different simulations running at normal game speed.

Simulation	Entities	FPS (Average)
PP	20	200
PP	60	80
PP	100	10

Table 4.1: FPS achieved with different entities in simulation

When running simulations of 100 or more entities, the load on the CPU would regularly spike which would slow down the program considerably and the observed FPS would often be less than five during these spikes. Spikes were observed to be caused when the animals were closely packed together around each other. Screenshots of the Unity Profiler during simulations are included in Appendix A.

5

Discussion

In this section, results are analysed and interpreted, difficulties experienced throughout the methodology and development of the project are evaluated, ethical and social aspects of this project are analysed, and future work along with potential improvements are discussed. Additionally, potential alternative approaches and solutions are compared in order to provide a valuable discussion.

5.1 Results Evaluation

Takeaways and problems regarding the results from section 4 are discussed in the following section, divided into five parts. The first part discusses to which extent different environments gave rise to distinct behaviours. The second part discusses problems with the design of the FCMs and the third part discusses how well the ecosystems were able to thrive. The fourth part explains the ability to add new actions to the FCM as a behavioural model and the final part discusses the practicality of the simulations in terms of performance.

5.1.1 Environment dependence of behaviours

The results obtained from the simulations clearly show that the different environments made a difference to the evolution of the rabbits' FCMs and behaviours. These behaviours were not trivial either, and it was shown that the improvements were made step wise. For instance, for the peaceful simulations the `SearchingForMate` action always saw a decline in popularity during the first generations, to then see an upshot further along the line ultimately leading to the population becoming self-sustainable. This is most likely the case because the rabbits that managed to balance drinking and eating would outlive the children that did not know how to do this properly, even though the child was born later than the other rabbits. It was not until the rabbits managed to live until old age that mating truly became beneficial, since the child could properly sustain itself to outlive its parents.

The hostile simulations saw an even more interesting progression in behaviour. The `Escape` action would quickly become prioritised among the rabbits. As time went by, the `Escape` action decreased in popularity and the rabbits often figured out how to balance eating and escaping quite well. This would however only allow

the rabbits to live until they died of thirst. The GoToWater action needed more time to manifest itself and the theory behind this is that the rabbits would always be stationary when they drank water, and thus be vulnerable to predators. The GoToFood action would on the other hand allow them to move around, as would the SearchingForMate action, which might suggest why the rabbits of PH40-1 and PH20 chose to perform this action a lot. The rabbits performed the SearchingForMate action a lot more than the rabbits in PP40, which probably means that they used this action for moving around, and not only for breeding. The animals in PH40-2 did not evolve to perform this action and it can be seen that the escape action remains quite high, which also suggests that the escape action is needed if the SearchingForMate action is not performed. In any case, when comparing PH20, which was the most sustainable hostile simulation, to the peaceful simulations it becomes apparent that the behaviours were environment dependent. The most developed rabbits in PH20 were using the Escape around 20% of the time whereas the most developed rabbits in the peaceful environments used it close to 1% of the time.

Another benefit that the rabbits of PH40-1 and PH20 got from the SearchingForMate action was that a lot of children were born. This makes it so that the foxes have a lot of targets to hunt which should in theory mean that every rabbit has a higher chance of survival since there is a lower probability for each rabbit to be hunted.

Two other forms of environment dependence that emerged from the simulations were the population sizes and the life span of rabbits. Simulations with a lot of plants greatly increased the number of rabbits that could thrive and hostile environments resulted in fewer rabbits becoming old. This means that not only do the different environments impact how each animal behaves, they also impact how entire populations look. This can be seen as another form of variation which may be desired by people who play video games.

5.1.2 The evolving perceived concepts in the FCM

One unforeseen side effect of the FCM implementation was the naming of perceived concepts becoming detached of their semantic meaning. For example, some FCMs evolved to have the input node hunger affect the perceived node pHunger negatively, which is not realistic. This happened because there were no rules implemented regarding which perceived nodes each input node could form weights to, so these evolved randomly. This made it difficult to understand the structure of any given FCM but they still worked as intended. For instance, an FCM with a weight of 1 from hunger to pHunger followed by a weight of 1 from pHunger to goingToFood would have the exact same function even if the perceived node had been pHeat.

The nondeterministic nature of the perceived states also led to the structures of the FCMs being inconsistent between different simulations. Having the structures being consistent would have made it easier to determine what behaviour each FCM had and compare them. This was not a huge problem though since the action graphs were sufficient for comparing behaviours.

5.1.3 Sustainability of simulated ecosystems

The simulations required many generations to improve, which meant that the animals had an unaccustomed behaviour prior to that. For example, many rounds of the simulations ended at 275 seconds, meaning that the animals would only eat and drink without mating. This is an example of a local maximum which the animals had difficulty escaping from. In order to escape these maximums the GA had to explore solutions which would sometimes be hard to find. For instance, if the population of rabbits were locked into a behaviour of solely eating and drinking, then the GA would require a lot of exploration to reach a behaviour where another action such as `SearchingForMate` would overrule the other ingrained actions. As viewed in simulations `PH40-1`, `PH40-2` and `PH20`, escaping rabbits proved to be deemed the most fit initially, before the act of `GoingToFood` was explored, improving the rabbits even further.

Another problem with regard to sustainability was the somewhat arbitrary round time threshold of one hour that was defined in order to classify ecosystems as self sustaining. For example, the rabbits in simulation `PH20` went extinct even after surviving for multiple hours, which might make the threshold seem like a bad metric. This is not a problem though. There is no guarantee that any ecosystem will thrive indefinitely but it was still necessary to have some rule to go by in order to determine when an ecosystem was considered sustainable enough. Furthermore, if a species went extinct it could simply be spawned back into the simulation.

5.1.4 Extensibility of actions

FCMs paired with evolution is practical in terms of extensibility since new actions and inputs can be integrated without having to balance and map them to each other explicitly. Given that a simulation gives rise to a thriving ecosystem, all necessary inputs and actions of the FCMs will have been mapped to each other. Note however that if all weights connected to an action in an FCM evolve to zero would mean that the corresponding action would never be used. This can also happen even if the weights are not zero if there is always some other action with higher priority. These problems can occur in two situations.

The first situation is when all actions are not needed to reach a local maximum of the fitness function for a simulation. This means some actions might never be used if the simulation does not manage to introduce enough diversity in the gene pool to explore beyond the local maximum. One such case was seen in simulation `PH20` where rabbits stopped escaping from foxes in one round. Rabbits might not have had to escape to survive as a species since they greatly outnumbered the foxes although escaping could have made each rabbit live longer. Avoiding local maximums could be circumvented by making severe mutations more common so more behaviours are simulated.

The second problem with regard to extensibility is situations where an action always has a negative impact on fitness but is still desired. An example of this could be a sleep action in which animals become stationary and unable to react to their environment without any positive effects. If the evolutionary mechanisms are able to optimize the fitness function well enough, this sleep action would ultimately never be used since it is worse than simply standing still. To avoid these situations it would help to introduce some benefit for each action to make them viable and have a potential to increase fitness. This benefit could either be tangible such as increasing the animal's energy or simply be a direct fitness increment. For example, sleeping for one second could increase fitness by ten.

5.1.5 Performance

The CPU load of the simulations shown in section 4.2 is definitely acceptable for some games considering that simulations with 60 entities managed to stay above 30 fps which is the standard frame rate for many games. However, the frame rate of the simulations decreased as the number of entities increased. The current implementation could therefore be impractical in games such as Minecraft where occasionally more than 100 entities are in the same place at the same time. On the other hand there are games such as Terraria that have fewer entities on screen generally, meaning evolving FCMs could be used as a behavioural model in these cases.

The ecosystem modeled using FCMs by Gras et al. had a much slower execution time of 40 minutes per time step which would correspond to 0.0004 fps [13]. This slow execution was the result of simulating 400 000 entities simultaneously, meaning their performance could most likely be comparable or better than our if they used the same number of entities as us. However, the ecosystem simulated by Gras et al. took place in a non-graphical, two dimensional grid consisting of one million discrete cells, which requires less performance than the types of environments that appear in video games. This highlights the importance of the performance results of our implementation, that the performance of a behavioural model based on FCMs is acceptable in a video game environment specifically.

5.2 Difficulties

Throughout any successful project there will be a range of challenges that arise leading to difficulties that need to be overcome. This project has been no exception.

One of the main difficulties when simulating the evolution was the amount of time that each simulation took. A lot of resources were put into solving this problem with the hope of being able to run the simulation at high speeds such as 100x the standard game speed in Unity. Actions were taken to solve this including disabling animations, simplifying senses such as sight as well as trying to bypass Unity's own game speed functionality to try and implement our own. Although these actions improved the performance of the simulation, the speed could only be set to around 5- to 10x the normal speed without putting too much strain on the CPU. Being

able to run the simulation at a higher speed would have allowed us to fine-tune certain variables in the simulation such as the standard deviations for the mutation operator, mutation probability, crossover probability, fitness function etc.

As mentioned in section 3.1.2, the foxes could not die and only existed in order to kill rabbits. This meant that only the rabbit was subject to evolution, which was not the original idea behind our research. We originally wanted to see how rabbits and foxes evolved together in an ecosystem to see if interesting behaviours emerged. This idea had to be abandoned due to the difficulty of finding a good balance between rabbits and foxes. For instance, the foxes would need to be significantly fewer than the rabbits, since they would otherwise need to wipe out the entire rabbit population in order to survive. Having fewer foxes would mean that more rounds were needed in order to reach a sufficient population size of around 100 foxes, which would slow down the evolution of the animals. Due to this time issue, the main investigation was made on rabbits with pre-defined foxes.

One challenge associated with procedural content generation is the freedom associated with using various techniques. There is not really a clear path towards a refined product and each developer needs to freely interpret and combine various techniques to produce a satisfying result. Perlin noise will not produce a coherent terrain with multiple elements by itself, but can be a valuable tool in the hands of a developer with a sufficient amount of creativity. Verifying that there will not be inconsistencies in certain generated environments also becomes complicated. If a system is able to produce over a thousand different variants, then there is a large probability that certain defects can stay unnoticed.

5.3 Application

There are a couple of ways which our research and solution discussed can be applied or used in future work. One interesting way would be to turn the project into one which is more interactive. An option would therefore be to control one of the animals and attempt to survive in the environment by going up against a multitude of different animals and species. This was discussed throughout the entirety of the project but never materialised as it was outside of the initial scope. It would entail slowing down the speed so that an animal could be controlled in real time, where one human controlled entity could coincide with the NPCs. The results could be fairly interesting to see if a player's instincts in acting on the information received from an animal could be better or worse than that of the other animals who have developed an ideal fitness to suit the environment. The results from this could perhaps show us how the way we as humans believe is the optimal way to act on our senses is comparable to how the evolved species act on their instincts. Essentially it could be seen as both a fun interactive way to live in the ecosystem, or even a challenge to outlive all of the other animals and avoid dying from malnourishment or becoming prey. As aforementioned, although this is outside of the initial scope, game developers could apply this research and apply it in order to create a fun and interesting game.

5.4 Social and Ethical Aspects

The use of open source code has long been a topic of discussion given that some people believe there is an ethical responsibility that comes with using another person's creation [50]. Researchers from select universities around the United States have widely discussed the topic of ethics within open source software, as have multiple online publication mediums [51]. For example, Ben Werdmuller states in his article that "OpenSSL, an open source service which every major internet company depends on, was until recently receiving just $2,000 a year in donations, with the principal author in financial difficulty." This is an ethical issue which puts a question mark over the sustainability of projects that drive large parts of our society as major companies are making money of another persons work without them receiving recognition for it. This is being highlighted as it is important to discuss ethical aspects and attempt to mitigate them as much as possible. Hence, this project has used a couple of open source assets from the Unity Asset Store. However, they are not intended to be used commercially or released to the open market without first analysing the licence agreement. Should the simulation be made available and the licence agreement states they are not clear for commercial use, these assets, which are currently only there for visual effect, would either be removed or a permission request would need to be accepted by their creators. Additionally, should a third party attempt to use this simulation for monetary benefits, it would be a breach of intellectual property regulations.

5.5 Alternative Solutions

As previously mentioned in the introduction, there was extensive discussions at the beginning of the project to determine whether cellular automaton should be used. In section 3.4 within the method portion, it was then clarified why an FCM was chosen as oppose to the ever reliable cellular automaton [52]. However, this may not have been the case, assuming that cellular automaton was used throughout the methodology and development instead, there may have significant differences in the final product and the results. Looking at the expected results, they would be expected to produce entirely different values for the two variants of the simulations discussed in section 3.1. This is due to the fact that cellular automaton is less dynamic and does not continuously update throughout the simulation as it has predefined final values. For example, as previously described in the introduction, a predefined rule that 2 cells with foxes should eliminate a single cell with a rabbit given that they are all adjacent. This would lead to a much simpler solution which would be highly reliable and predictable leading to results being as expected. Additionally, using cellular automaton could have avoided a host of difficulties described in the previous section as there would not be a huge uncertainty as to how the traits and actions would develop throughout the simulation.

There was a similar simulation found which was done by Sebastian Lague [53]. The interesting part of this simulation was that it was also a simulation of predator-prey with both rabbits and foxes. From the video posted, Sebastian's simulation seems to

mainly focus on the changes to the species population over time given certain factors such as desirability and reproduction rate [54]. A similarity to Lague's simulation is that the predator and prey populations behave in very similar ways. In both simulations, the rabbit population grows until foxes are introduced, who then seem to slowly but surely exhaust all of the rabbits. Therefore, the solution discussed throughout this paper seems realistic in terms of predator-prey and also population evolution.

An alternative solution, similar to Lagues simulation, could therefore be to create completely trait based behaviours. Consider a rabbit with two traits, *diet* and *speed*, and that is coded to always eat food. It might appear that a fast rabbit which eats other animals has a different behaviour compared to one which is slow and eats plants, even though they are implementing the same behaviour. Lague had 4 main traits which evolved; speed, gestation period for females, reproductive urge, and sense distance. These traits were then monitored throughout the simulation and then their development could be seen over a longer period of time. The simulation discussed throughout this report however, differs because no trait evolution is measured. Instead results are based on the evolution of the FCM, which in turn describes which actions an animal values highest. An alternative solution could therefore have been to look at the evolution of traits within the species, given different terrain and environments.

As mentioned in section 3.4 a certain portion of this project was inspired by the work of Robin Gras et al. and their research paper *An individual-based evolving predator-prey ecosystem simulation using Fuzzy Cognitive Map as behavior model.* [13] Although there are a large number of similarities between this project and Robin Gras' work the two have slightly diverged as development has proceeded and new ideas have been implemented. An uncanny similarity is the use of the FCM and how it alters depending on breeding and mutation. Both projects allow interbreeding between the same species and have the possibility of mutating the genes which in turn evolves the species giving them the best chance of survival. A slight difference is that Gras et al. allow new species to emerge during reproduction with mutations. They compare the mutated FCM to a "general" FCM for a specific species and given that they are distinct enough they are deemed to be different species. In Gras' words, "species can emerge or disappear at any time step".[13] This means that they look at speciation, which is something that our project does not. Although quite complex, some interesting results could have emerged by monitoring speciation throughout the simulation. A solution could be, for example, saving the values of the FCMs for for all the rabbits and calculate an average for those values denoting the species' general FCM. After the simulation had run its course the individual FCMs could be compared to the general one and given large distinctions a certain "rabbit" may now be classified as a different species. As this would be an interesting simulation perhaps ending in a large variety of species, it would have been a viable alternative solution and is something that could even be considered for future work as time is a limiting factor in this project.

5.6 Future Improvements

While the progress that has been made throughout this project is by no means disappointing, there are most definitely aspects that can be improved and further developed given the time and effort. The most significant areas of improvement are explained in this section, namely implementing more terrain, evolving animals faster, making actions more dynamic and improving performance.

5.6.1 Integrating more variation in the terrain

Certain features of the procedurally generated terrain is expressed with a single height-map. This means the surface terrain perhaps might be a touch too uniform to model how different landscapes look in reality. For example, large sections of grasslands or mountainous regions are much more likely to occur in a physical ecosystem than in the simulated one in this project. A potential improvement is therefore spending additional time making the terrain and ecosystem as a whole much more realistic where agents react organically to that specific world. This would include flat savannas and grasslands, mountainous regions where there are clusters of peaks and valleys all contributing to a massive variety in the terrain where agents could end up having a preferred biome or perhaps altitude. One method which is commonly combined with Perlin noise to break this monotony is Voronoi Diagrams. This technique could be integrated into our procedure in order to get features such as mountain ranges. A significant amount of inspiration for future improvements within PCG could be taken from Noor Shaker's book "Procedural Content Generation in Games" [1]. Although this would be very interesting to spend time on improving in the future, it is more of a luxury rather than a necessity for the purpose of this simulation.

5.6.2 Faster evolution

One of the most difficult tasks was designing the evolution mechanisms to allow animals to adapt in a fast manner while still maintaining their fitness between generations. The current evolution system contains a myriad of different parameters such as standard deviation for the mutation operator, mutation probability, crossover probability, fitness function, the number of parents to be selected and the number of animals copied with elitism per round. There was not enough time to thoroughly explore the best configuration for each parameter and there is certainly room for improvement here. One can also investigate other genetic operators or even explore other evolutionary algorithms further such as evolutionary strategies or evolutionary programming.

5.6.3 Unbiased actions

In this project, the animals evolved to perform actions depending on their current state and environment. The actions themselves were however completely predictable and worked the same for every animal of the same species. For instance, if an animal

was given the directive `GoToFood`, the food object that it would move to would be calculated by a predefined mathematical formula based on variables such as size and proximity of the food object.

One improvement would be to make certain aspects of the actions subject to mutation as well. This would potentially allow for even more interesting behaviours. One common problem was for instance that all rabbits were heading for the same plant when they were given the directive to eat, which caused them to battle over it even though there could be plenty of free plants around them. The `GoToFood` action could have been made more advanced so that each animal could prioritize food sources differently.

5.6.4 Evolving traits and species

Having traits evolve was an early idea that did not make it into the final build as it would have been too much work to balance both the FCM and traits of the rabbits. More traits would allow for more distinct and interesting rabbits that would see the world differently from one another. This could be taken even further by letting different groups of rabbits diverge into species, similarly to what was done by Robin Gras et al. [13]. Different species could have unique interactions with each other, for example rabbits viewing foxes as friendly. Rabbits could also be cautious with hares when they first meet, but as time elapsed they could become used to each others presence and learn to coexist despite remaining distinct species.

5.6.5 Performance improvements

The program was quite performance heavy when a lot of entities were simulated, as seen in section 4, and it would be good if performance could be improved as it would make the program more accessible in video games. One of the heavy processes seems to have been the animals' senses. The CPU-load was observed to increase drastically whenever a large amount of animals were grouped around each other. This is most likely because every rabbit sensed all the rabbits (and other objects) around it, meaning that the total time spent making sense calculations would increase exponentially whenever the rabbits were grouped. The whole process of sensing could therefore be improved. One suggestion would be to run performance heavy tasks with bigger gaps in between. The problem that arises here is that entities will react much slower to changes in the environment if the time gaps are too large. Therefore it would be useful to investigate how to balance responsiveness and performance. Another solution could be to simply improve the underlying algorithms as a whole.

6

Conclusion

This report has investigated the possibility of creating dynamic ecosystems, by using a fuzzy cognitive map as behaviour model and procedurally generating the environment. NPC decision making is based on the variables that the individual perceives. These variables are then fuzzified and used as an input for the fuzzy cognitive map. Furthermore, the FCM is also evolved using genetic algorithms selecting NPCs that survive the longest. The landscape is mainly generated by translating results of a Perlin noise function and the water is created by using Delaunay triangulation.

The program shows that FCMs can be used to handle decision making which consists of determining when to perform actions such as eating, drinking, escaping and mating. Furthermore, it is also shown that the FCMs do not have to be constructed by hand, but can be created through an evolutionary process by using genetic algorithms. Animals participating in an evolutionary process are shown to survive for an increasing period of time as a result of becoming adapted to their environment. Animals living in different environments also evolved to exhibit different behaviours. For instance, results show that rabbits who exist in a world with predators evolve to frequently perform the escape action, whilst rabbits existing in an peaceful environment free of predators evolve without requiring the escape action. The environments also impacted how many animals could sustain and how long each animal lived.

There are some disadvantages associated with the methods used throughout the project. The most significant being the high CPU load when there are more than 60 entities present in the simulation. This is a limiting factor as the simulation cannot be run with large numbers of animals. Furthermore, running the simulation at higher speeds also increases the CPU load. A desired outcome would be to run the simulation at very high speeds in order to yield results faster but due to CPU limitations this was not possible.

The use of a FCM shows promise as a cognitive model that can be evolved in an ecosystem. Although, currently the high CPU usage limits the use of a FCM in real time applications where a high number of entities need to be simulated, the results obtained and discussed can hopefully help other developers attempting to develop similar solutions in the future. The implicit aim is that this solution will facilitate understanding of how FCMs can be used to create more intelligent NPCs by giving them the ability to evolve according to factors determined by game developers.

On a final note, although the purpose of this project has been fulfilled, showing how behaviours of NPCs can evolve in an ecosystem using an FCM as behavioural model, it proved to be more difficult than expected to simulate the changes over a long period of time due to the lack of computational power. Therefore, there is still a need for some further research and improvement in order to best run a large scale simulation which consistently evolves an FCM over multiple generations to produce fit NPCs depending on the environment around it. An interesting topic for further research is optimizing the parameters for the genetic algorithm used in order to make simulations produce fit NPCs faster and more reliably.

Bibliography

[1] N. Shaker, J. Togelius, and M. J. Nelson, *Procedural Content Generation in Games*, 1st ed. Springer Publishing Company, Incorporated, 2016.

[2] G. N. Yannakakis, "Game ai revisited," in *Proceedings of the 9th Conference on Computing Frontiers*, ser. CF '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 285–292. [Online]. Available: https://doi.org/10.1145/2212908.2212954

[3] M. Prinke. (2017) Why is the video game experience emphasizing open world environments more and more these days? [Online]. Available: https://www.quora.com/Why-is-the-video-game-experience-emphasizing-open-world-environments-more-and-more-these-days/answer/Mike-Prinke

[4] Nintendo, "The legend of zelda: Breath of the wild," *Game [Switch]. Nintendo, Kyoto, Japan*, 2017.

[5] J. Togelius, *Playing Smart: On Games, Intelligence, and Artificial Intelligence*. The MIT Press, 01 2019. [Online]. Available: https://doi.org/10.7551/mitpress/11723.001.0001

[6] builtin. [Online]. Available: https://builtin.com/artificial-intelligence

[7] I. Millington, *AI for games*. CRC Press, 2019.

[8] J. P. Salmon, S. M. Dolan, R. S. Drake, G. C. Wilson, R. M. Klein, and G. A. Eskes, "A survey of video game preferences in adults: Building better games for older adults," *Entertainment Computing*, vol. 21, pp. 45 – 64, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1875952117300381

[9] J. Cox, "Variety in the video game industry: An empirical study of the wundt curve," *Managerial and Decision Economics*, 10 2017.

[10] R. Valdes, "The artificial intelligence of halo 2," Nov 2004. [Online]. Available: https://electronics.howstuffworks.com/halo2-ai.htm

[11] F. Jopp, H. Reuter, B. Breckling, *Modelling Complex Ecological Dynamics: An Introduction into Ecological Modelling for Students, Teachers & Scientists*. Springer, Berlin, Heidelberg, 2011. [Online]. Available: https://link.springer.com/content/pdf/10.1007%2F978-3-642-05029-9.pdf

[12] F. Brauer and C. Castillo-Chavez, *Introduction and Mathematical Preliminaries*. New York, NY: Springer New York, 2012, pp. 123–164. [Online]. Available: https://doi.org/10.1007/978-1-4614-1686-9_4

[13] R. Gras, D. Devaurs, A. Wozniak, and A. Aspinall. (2009-10) An individual-based evolving predator-prey ecosystem simulation using a fuzzy cognitive map as the behavior model. [Online]. Available: https://www.researchgate.net/publication/26234726_An_Individual-Based_Evolving_Predator-Prey_Ecosystem_Simulation_Using_a_Fuzzy_Cognitive_Map_as_the_Behavior_Model

[14] Gisling, *Lotka Volterra equation Maple plot*, Dec 2013. [Online]. Available: https://commons.wikimedia.org/wiki/File:Lotka_Volterra_equation_Maple_plot.png

[15] B. Breckling, G. Pe'er, Y. G. Matsinos, *Cellular Automata in Ecological Modelling*. Springer, Berlin, Heidelberg, 2011. [Online]. Available: https://link.springer.com/content/pdf/10.1007%2F978-3-642-05029-9.pdf

[16] P. Bak, K. Chen, M. Creutz. (1989-12-14) Self-organized critically in the 'game of life'. [Online]. Available: https://www.nature.com/articles/342780a0.pdf?origin=ppub

[17] P. Rendell, "A universal turing machine in conway's game of life," in *2011 International Conference on High Performance Computing Simulation*, 2011, pp. 764–772.

[18] P. Hogeweg. (1988-07) Cellular automata as a paradigm for ecological modeling. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0096300388901002

[19] L.O. Moraesa, C.E. Pedreira, S. Barrena, A. Lopez, A. Orfao. (2019) A decision-tree approach for the differential diagnosis of chronic lymphoid leukemias and peripheral b-cell lymphomas. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016926071831736X

[20] R.B Mofrad, N.S.M. Schoonenboom, B. M.Tijms, P. Scheltens, P.J Visser, W.M. van der Flier, C. E.Teunissen. (2018) Decision tree supports the interpretation of csf biomarkers in alzheimer's disease. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352872918300721

[21] S. D. Marko Debeljak, *Decision Trees in Ecological Modelling*. Springer, Berlin, Heidelberg, 2011. [Online]. Available: https://link.springer.com/content/pdf/10.1007%2F978-3-642-05029-9.pdf

[22] M. Debeljak and S. Džeroski, *Decision Trees in Ecological Modelling*, 01 2011, pp. 197–209.

[23] J. Togelius, E. Kastbjerg, D. Schedl, and G. Yannakakis, "What is procedural content generation? mario on the borderline," in *2nd International Workshop on Procedural Content Generation in Games, PCGames 2011 - Co-located with the 6th International Conference on the Foundations of Digital Games*, 2011.

[24] (2016) Rogue. [Online]. Available: http://pcg.wikidot.com/pcg-games:rogue

[25] J. Fingas. (2015) Here's how 'minecraft' creates its gigantic worlds. [Online]. Available: https://www.engadget.com/2015/03/04/how-minecraft-worlds-are-made

[26] B. Kosko. (1986) Fuzzy cognitive map. [Online]. Available: http://sipi.usc.edu/~kosko/FCM.pdf

[27] L.A. Zadeh. (1965) Fuzzy sets. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S001999586590241X?via%3Dihub

[28] R. Axelrod, *Structure of decision: The cognitive maps of political elites*. Princeton university press, 2015.

[29] E. Kayacan and M. A. Khanesar, "Chapter 2 - fundamentals of type-1 fuzzy logic theory," in *Fuzzy Neural Networks for Real Time Control Applications*, E. Kayacan and M. A. Khanesar, Eds. Butterworth-Heinemann, 2016, pp. 13 – 24. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780128026878000025

[30] S. L. Ö. Uygar Özesmi. (2003) Ecological models based on people's knowledge: a multi-step fuzzy cognitive mapping approach. [Online]. Available: http://levis.sggw.pl/~rew/scenes/pdf/Ozesmi.pdf

[31] E. Papageorgiou and J. Salmeron, "A review of fuzzy cognitive map research at the last decade," *IEEE Transactions on Fuzzy Systems*, 01 2013. [Online]. Available: https://www.researchgate.net/publication/232707632_A_Review_of_Fuzzy_Cognitive_Map_research_at_the_last_decade

[32] M. Glykas, *Fuzzy cognitive maps: Advances in theory, methodologies, tools and applications*. Springer, 2010, vol. 247. [Online]. Available: https://link.springer.com/content/pdf/10.1007%2F978-3-642-03220-2.pdf

[33] D. Carli, F. Bevilacqua, C. Pozzer, and M. d'Ornellas, "A survey of procedural content generation techniques suitable to game development," 11 2011, pp. 26–35.

[34] K. Perlin, "An image synthesizer," in *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '85. New York, NY, USA: Association for Computing Machinery, 1985, p. 287–296. [Online]. Available: https://doi-org.proxy.lib.chalmers.se/10.1145/325334.325247

[35] W. Pokojski and P. Pokojska, "Voronoi diagrams – inventor, method, applications," *Polish Cartographical Review*, vol. 50, pp. 141–150, 09 2018.

[36] H. Edelsbrunner, "Triangulations and meshes in computational geometry," *Acta Numerica 2000*, vol. 9, pp. 133 – 213, 01 2000.

[37] D. Austin. (2006, Aug) American mathematical society. [Online]. Available: http://www.ams.org/publicoutreach/feature-column/fcarc-voronoi

[38] Hferee. (2011) Delaunay voronoi. [Online]. Available: https://commons.wikimedia.org/wiki/File:Delaunay_Voronoi.svg

[39] J. H. Holland, "Outline for a logical theory of adaptive systems," *J. ACM*, vol. 9, no. 3, p. 297–314, Jul. 1962. [Online]. Available: https://doi.org/10.1145/321127.321128

[40] K. De Jong, D. Fogel, and H.-P. Schwefel, *A history of evolutionary computation*, 01 1997, pp. A2.3:1–12.

[41] National Geographic. Natural selection. [Online]. Available: https://www.nationalgeographic.org/encyclopedia/natural-selection/

[42] J. H. Holland, *Genetic Algorithms and Adaptation*. Boston, MA: Springer US, 1984, pp. 317–333. [Online]. Available: https://doi.org/10.1007/978-1-4684-8941-5_21

[43] D. Rani, S.K. Jain, D.K. Srivastava, M. Perumal. (2013) 3 - genetic algorithms and their applications to water resources systems. Oxford. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780123982964000039

[44] K. Jebari, "Selection methods for genetic algorithms," *International Journal of Emerging Sciences*, vol. 3, pp. 333–344, 12 2013.

[45] Cronholm144, *Extrema*, Jun 2000. [Online]. Available: https://commons.wikimedia.org/wiki/File:Extrema.svg

[46] P. Kora and P. Yadlapalli, "Crossover operators in genetic algorithms: A review," *International Journal of Computer Applications*, vol. 162, pp. 34–36, 03 2017.

[47] K. De Jong, "Learning with genetic algorithms: An overview," *Machine Learning*, vol. 3, pp. 121–138, 10 1988.

[48] G. Ochoa, I. Harvey, and H. Buxton, "On recombination and optimal mutation rates," in *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - Volume 1*, ser. GECCO'99. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, p. 488–495.

[49] R. Kala, *Optimization-Based Planning*. Butterworth-Heinemann, 2016.

[50] F. Grodzinsky, K. Miller, and M. Wolf, "Ethical issues in open source software," *Journal of Information, Communication and Ethics in Society*, vol. 1, pp. 193–205, 11 2003.

[51] B. Werdmuller, "Why open source software isn't as ethical as you think it is," Sep 2017. [Online]. Available: https://words.werd.io/why-open-source-software-isnt-as-ethical-as-you-think-it-is-2e34d85c3b16

[52] P. Gacs, "Journal of statistical physics," *Reliable Cellular Automata with Self-Organization*, vol. 103, p. 45–267, Apr 2001. [Online]. Available: doi.org/10.1023/A:1004823720305

[53] SebLague, "Seblague/ecosystem-2," Aug 2019. [Online]. Available: https://github.com/SebLague/Ecosystem-2

[54] S. Lague, "Coding adventure: Simulating an ecosystem - youtube," Jun 2019. [Online]. Available: https://www.youtube.com/watch?v=r_It_X7v-1E

A

CPU Load

Figure A.1 through A.4 shows the load of the CPU during simulations. This appendix shows screenshots taken from the Unity profiler during simulations. The two most prevalent colors, yellow and blue represent “Others” and “Scripts”. According to the Unity Manual “Others” is overhead-management. The scripts-section include written code, for example: animals sensing their environment or the calculation of an FCM.

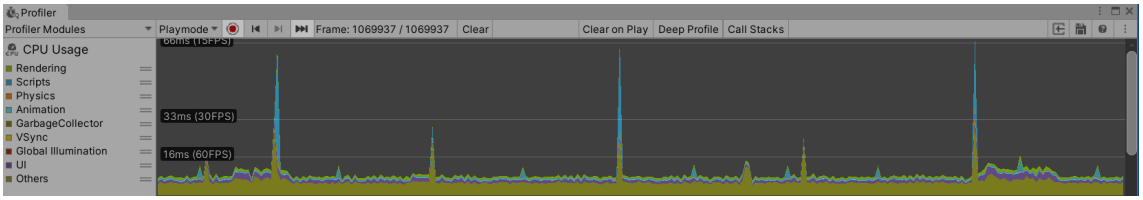


Figure A.1: The Unity profiler during a simulation with 20 entities

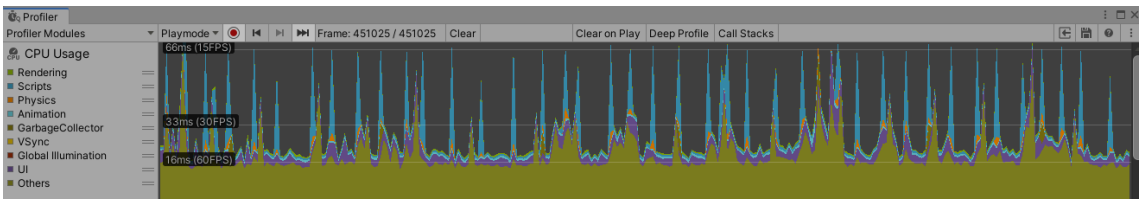


Figure A.2: The Unity profiler during a simulation with 60 entities

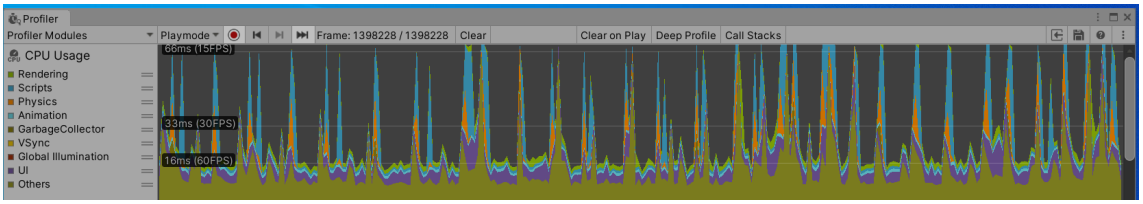


Figure A.3: The Unity profiler during a simulation with 100 entities

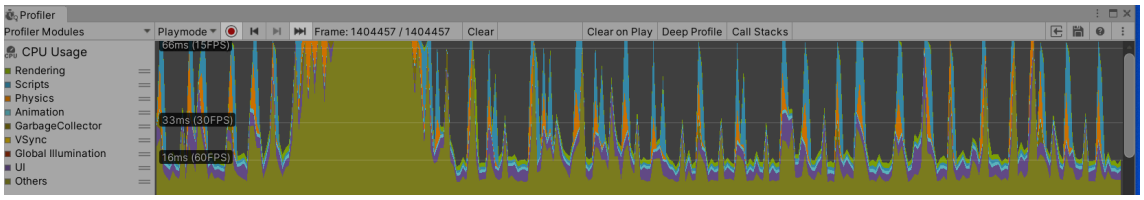


Figure A.4: The Unity profiler during a simulation with 100+ entities where the CPU-load spikes for a couple of seconds