Study of Evolution in Virtual Worlds

Hassene Ben Amar
Polytech Marseille,
Doshisha University
163 Avenue de Luminy
France, 13009, Marseille
hassenebenamar03@gmail.com

Masashi Okubo
Doshisha University
Tataratanioku, Keikikan
Building
Japan, 610-0321,
Kyotanabe
mokubo@mail.doshisha.ac.jp

ABSTRACT

This paper investigates the evolution of agents within a virtual world, focusing on DNA transfer between generations, identification of significant genes and the explorations of parameters influencing survival and gene significance. To address these questions, we create an artificial life simulation within Unreal Engine 5, which mirrors real-life characteristics and behaviors of animals. The methodology involves running a genetic algorithm with binary tournament, uniform crossover and n-point mutation and analyzing the collected data to determine the most significant genes in different cases. We demonstrate that parameters such as sensory capabilities, resource availability and mutation thresholds greatly influenced species' survival and their success in the virtual environment. The difference of size, health and sight capabilities is crucial for the survival of deers and their interaction with the environment. In conclusion, this study offers an insight on evolution and evolution dynamics, denoting the influence of resource availability, competition between agents, mutation thresholds and sensory capabilities, with a substantial potential if time and resources were allocated to the research.

0.1 Keywords

Virtual Worlds, Predator-Prey Problem, Genetic Algorithms, Computer Generated Simulations

1 INTRODUCTION

In a world where the environment is constantly changing due to various factors such as climate change, migration to the cities, industrialization, etc, we would like to predict or at least have an idea on how the different species that live on the globe might evolve in the future. We would also like to be able to understand how species used to be in the past and how they did evolve from their past self to their current descendant. The method we will use to try to bring an answer to this problem is to simulate the evolution system by using the concept of Artificial Life itself in a graphic engine (in our case, Unreal Engine 5). In this paper, replicating life and simulate the evolutionary system would follow a modified version of the genetic algorithm based on multiple conditions - the agent attribute or A.A which contains multiple statistics that we will develop more in the later of the paper. Depending on the agent species we will evolve, we want to be able to have multiple levels that simulate different types of environments, for example - a mountain range, a cave, a forest, plains, etc. We bring a new approach to the traditional genetic algorithm by implementing different species in the same level and creating a randomness of environment with the different resources available to the agent during his artificial life. The agents that exist in the level are not at all static and live depending on goals we implement in their A.I behavior. Finally, the performance evaluation index is based on the overall performance of the simulation and the gap between reality and our virtual world.

2 CREATING A REAL WORLD SIMULATION

The purpose of this paper is to create a realistic life simulation in Unreal Engine 5 (UE5) and study the evolution process of agents inside of it. We have to consider different factors to achieve such a result. These factors include giving the agents a physical form to interact with the environment, with other agents (predator and prey alike), being able to differentiate the agents with the help of attributes, implementing sight and hearing capabilities and automatically creating and managing resources. In the following sections, we will examine each of these factors in detail to develop a comprehensive realist life simulation.

2.1 The agent

The natural world is home to a vast array of ecosystems, blooming with a wide range of flora and fauna. To simulate these ecosystems and the organisms that inhabit them in a realistic manner, we must strive to accurately replicate their biological processes. Nevertheless, this

task is challenging due to the limitations of computational power. Our objective is to create virtual models of life that closely approximate reality while maintaining optimal performance. Therefore, we have to first recreate the fauna.

2.1.1 External representation

To interact with its environment and other agents in the system, each agent is required to possess a physical form (a 3D mesh) in the virtual world. Compared to [Ric01a], we did not model the assets ourselves, which could limit the computational performance of the system during collisions' calculations. Meshes used were complex and irregular in shapes, leading to a difficulty in defining collisions using primitives shapes. A solution was found using "Bounding Spheres" or "Sphere Capsule Collision" to wrap the model in an invisible sphere hence a simpler method to calculate collisions.

2.1.2 Internal representation

An agent has a physical form but it lacks the complexity that stems from life. Each species has abilities, faculties that separate them from others. In a more local scope, each of the members of these species is unique and has a different value on how much those abilities express themselves. DNA contains the information required to determine an animal's abilities. Due to a current limitation of computational power, modeling an animal's DNA the way it appears in the real world would prove challenging. Therefore, we found that the creation of a simplified DNA structure along with attributes that depend on the structure fixes our issue.

The agent's DNA

In animals, the length of the DNA structure can vary.

Animal Class	Average Length (base pairs - bp)	Minimum Length (base pairs)	Maximum Length (base pairs)
Mammals	3.5 billion	242 million	6.3 billion
Avian	1.385 billion	1.15 billion	1.62 billion
Serpentes	2.8 billion	1.3 billion	3.8 billion
Fishes	1.4 billion	103 million	133 billion

Table 1: Average, minimum, and maximum length of the DNA structure in the mammal, avian, serpentes, and fish classes.

In terms of memory, 1000000 base pairs (bp) = 1 Mb and 1000000000 bp = 1 Gb. With the number of agents required for the simulation and the technical constraints, even when considering modern computational power, we need to simplify the DNA data structure. We will discuss the implementation of the simplified DNA in a later part.

The agent's attributes

An animal is defined by its DNA and the abilities that the structure codes. An example of one of those abilities is the speed of the Cheetah, which has the title of "Land's fastest animal" [cheetah01] They are able to run extremely fast (at 70 mph or 112,654 km/h) because of a combination of physical traits and genetic adaptations such as their slender, muscular body, their enlarged nostrils and lungs (more oxygen = more energy), their long tail (maintains balance and control) and adapted claws (better traction and acceleration). We formalize those abilities, those adaptations in the form of attributes that possess numerical values and that depend on the simplified version of the DNA mentioned in the previous part. We created different variables that encompass all that is necessary to define an agent's abilities, for example: AgentID (unique identifier), AgentDamage (amount of damage the agent can inflict on hostile agents), SightRadius (radius of the agent's field of view). It should be noted that these attributes are an oversimplification of the true nature, primarily due to computational limitations and the inherent complexities involved in managing such extensive data. Simplifying the DNA data structure was necessary to navigate these complexities effectively. To initialize all attributes of an agent and automate the task, we created an algorithm which is explained hereafter.

Algorithm 1 Agent Initialization and Attribute Calculation

Initialize DNA structure of the Agent
Set sensory attributes: SightRange, Radius, Offset, HearingThreshold
for each sense **do**
    Calculate sense bonus
end for
Calculate statistics for each agent species with species-specific formulas
Initialize float attributes: AgentSize, Speed, Damage, Health, Survivability Score
Calculate:
AgentSize
Speed
Damage
Health
Survivability Score
Initialize Int Stats:
Set AgentID by random

2.2 Birth and Death simulation

In our world, birth and death are two processes that can make the number in a population of animals grow, wither or stay the same. Birth is the way new children come to life whereas death is the way life purges populations to keep numbers stable. This cycle of birth/death

is the most important part of life since it enables the ecosystems to preserve resources and maintain life as it is. The paper [Ric01a] defines the cycle in their simulation of Omosa with the creation of the agent as a baby (birth) that go through the multiple stages of life and death by old age or its health points dropping to 0 in case of a fight with a predator. Our simulation takes an entirely another approach to define the cycle of birth and death. Birth is defined by the live creation of an agent inside the level (one of the environments) or spawn for short. Death is defined by the live destruction of an agent inside the level or despawn.

2.2.1 Simulation of birth

To simulate said process in our virtual environment, we defined an invisible box called an AgentSpawner that possesses multiple characteristics - TotalToSpawn (the number of agents to spawn), MaxSpawnRange(the maximum distance at which the spawner can spawn agents), etc. If the location of the attempted spawn already contains an agent or another physical entity (actor in UE5) with collision enabled, the attempted agent spawn will fail and go to the next one.

Step 1: Check if the TotalToSpawn is higher than 0.
Step 2: Initialize i = 0
Step 3: Loop until i >= TotalToSpawn

- Calculate a random position with the Perlin variables on X and Y
- Spawn the actor

The first spawning process is completely different than for the rest of the simulation.

2.2.2 Simulation of death

In real life, death is essential to preserve the balance in ecosystems between the number of predators and prey, the availability and scarcity of resources. An animal can die of old age or from its wounds due to a fight with another animal, trying to defend its territory or while roaming freely in the environment. Our goal is not to simulate and solve the predator-prey but to see how DNA would travel between the generations. We don't create sub-populations of agents, which could lead to inner fighting over prey and resources and "unnecessary" deaths. Instead, all agents of the same specie are "friendly" with each other and don't have default interaction between them. The despawn of the agent would happen in two cases. The first case is that the AgentHealth attribute drops to 0. It is possible that there was a fight between predator(s) and prey(s) or if the AgentEnergy is under a threshold for a certain time. The second case is when the simulation passes to the next generation of agents. The figure 1 showcases a graph that sums up the cycle of birth/death for our simulation.

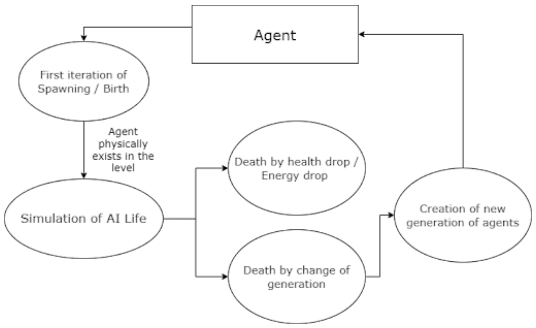

Figure 1: Graph of cycle of birth/death in our simulation

2.3 The Environment

The purpose of our research is to be able to create a realistic life simulation. In the real world, animals live in a variety of ecosystems, such as forests, mountains, caves, seas, etc. Each of these ecosystems contains an array of resources that are necessary for the survival and development of each species. For example, cheetahs live in grasslands, savannas and some mountainous regions. Those environments are all open landscapes with a lot of resources (food, shelter and water) where the cheetah can use its speed to its maximum potential to chase down prey. Based on this example, we can infer that each animal has an ideal environment where it can use to its fullest its capabilities. Therefore, to mimic reality as closely as possible, we need to try to recreate to its maximum possible the vast array of environments where all of the agent species prosper.

2.3.1 Initial level

Mimicking reality is a process that takes a lot of time and resources, to imagine and create the environment itself. To be able to test our agents' fundamental features while not having our environments yet ready, we created a simple level field consisting of basic shapes provided by Epic Games in UE5. This level does not accurately reproduce the environments in which the agent species typically live.

2.3.2 Environment design

Multiple ecosystems nurture life as we know it. In reality, here is a random factor that will determine where the fauna and flora will end up living. To fully mimic an actual environment, we need to recreate that randomness and make our worlds more realistic. Furthermore, we study the evolution of a multitude of agents. Some agent species won't share the same ecosystem because their real-life counterparts could never survive there. Hence, a need to create a multitude of ecosystems in order to cover a certain percentage of the environments the animals live in.

Simulating the resources

To infuse the simulated environments with the unpredictability of life, we take inspiration from solutions implemented by level developers over the course of video level history. A common solution in level design to create this chance factor or randomness is to use procedural foliage. It would be a good idea if we didn't want to add randomness to each restart of the simulation. Unfortunately, there is no free procedural foliage library on the Epic Games marketplace, hence a need to develop a solution for ourselves. We believe that changing the environment every 2 generations could influence DNA changes between generations of agents. The solution we developed is based on the AgentSpawner. We reused the concept of a transparent box with a physical location in the environment. Although the AgentSpawner architecture makes it only able to spawn one kind of agent, the EnvironmentSpawner is able to spawn a vast array of special actors called EnvironmentActor. An EnvironmentActor is an spawnable or placeable actor (depending on the needs) in the level that has multiple variables required to simulate how its real counterpart would behave (while being static). All the meshes that we use for the EnvironmentActor come from the megascans collection in Quixel Bridge (free with UE5).

2.4 Predator and prey system

As we try to reproduce their behavior in virtual worlds. We recreate the diet plan of each agent species. Our solution was inspired from the paper [Geo01a]. Both types of agents (predator and prey) in the paper possess the capability to get information from their environment and act based on their perceptions, though in different manners. The predator agent can only see in a straight line with a high range of sight while the prey can see in all directions with a lower sight range to not give it an unfair advantage.

2.4.1 Agent's perceptions

The paper [Geo01a]'s agents use sensors to provide the perception information to the robots. Without a perception system to detect hostile agents or resources, an artificial life simulation would not be complete. The simulation in the paper [Ric01a] collected "the total population, numbers of births and deaths for both prey and predators as well as the number of predator kills and prey deaths from old age" with different settings (predator/prey awareness, flocking, herding). When the predator class isn't aware of the prey, it fails to function properly and when the prey class isn't aware of the predator, it quickly becomes extinct. Unreal Engine 5 provides some systems to implement senses in the level but after careful consideration, we decided to implement our own sense of sight. We use "Multi Object Sphere Trace" x times per second, which consists of sweeping a view range and detecting objects in the agent's field of view. This would compensate for the short-comings of the UE5 powered systems and enable herbivore and omnivore agents to detect sources of food as well as enemies. All agents share the same set of attributes, including SightRange, SightRadius, SightOffset and HearingThreshold for the senses configuration. Nevertheless, we define how much an agent's senses are expressed based on the real-life counterpart's known ability and a bonus coming from DNA structure.

The DNA's structure is defined with 8 values - SightRange, SightRadius, SightOffset, HearingThreshold, AgentSize, AgentSpeed, AgentDamage, AgentHealth.

The architecture of the DNA bonus is a bonus that is calculated (randomly between 0 and x) and added to the value of the attribute only if the gene is equal to 1. This could either advantage or disadvantage an agent depending on how its perceptions would affect its actions.

2.4.2 Agent's actions and decision making

Our perception system reacts to the stimuli around it (in a limited range). The agent is an A.I powered actor in the level that has predefined actions based on what the agent's perception system would register from the environment. If the stimuli detected comes from a hostile agent (H.A), the agent could react via two distinct ways. If the H.A is one of its predators, the agent would run away from it until reaching a safe location where the perception system doesn't detect the H.A anymore. In the case of the H.A being a prey, the agent would chase it until reaching a radius called the CombatRadius where it could start attacking it until one of them dies. In addition to the predator-prey pursuit and combat system, an agent would also be able to register what EnvironmentActors exist in its field of view. It would calculate automatically the closest resource and move in its direction in order to consume its content. The next figure shows a graph summarizing how an agent can register perceptions and the decision making it can make.

It is safe to say that we managed to reproduce the different parts of an ecosystem, fauna and flora as well as the animals' behavior. The reproduction of animals in the level was made by giving agents a physical and internal representation to help them interact with the world, while simultaneously remaking the cycle of birth and death with the AgentSpawner and destruction functions. We made the environment with heightmaps and artistic techniques and recreated the unpredictability of resources location with perlin noise [perlin01]. Finally, we created a system to reproduce an agent's sight, hearing being too complex to add on our own for now.

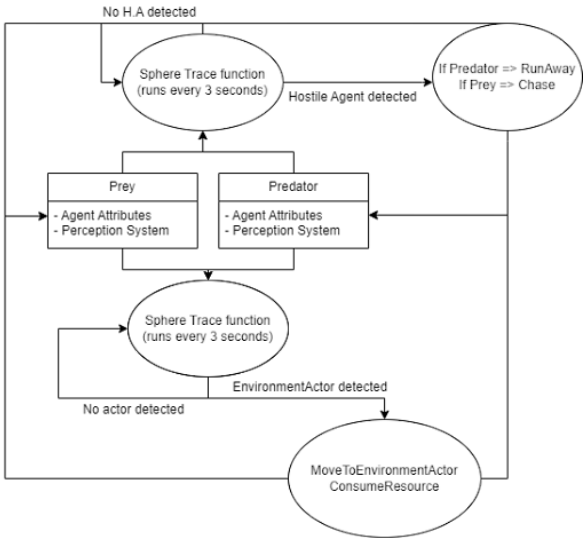

Figure 2: AI behavior, perceptions and decision making

Algorithm 2 Pseudocode of a Genetic Algorithm

```
Initialize the first generation
Evaluate the given generation based on a fitness function
if the first generation is the solution then
    return solution
end if
Selection of parent
Crossover of the parents' genetic representation and creation of children
Mutation of the offspring's genes
Check if we reached the termination condition
if not reached then
    Repeat from Step 2 with the offspring population
else
    Stop execution and print the found solution
end if
```

3 AGENTS' EVOLUTION

Evolution is the process over which heritable characteristics change in a certain population during the flow of time. The purpose of the paper is to study how our agents will evolve over the course of time and multiple generations, or tuning. Unfortunately, a problem such as this could not be solved using classical computing methods. Therefore, we follow the work of [Hol01a] with his well defined "Genetic Algorithm". In this part, we will view how this algorithm works, the different processes involved in it and how we remade them to fit our needs.

3.1 Overview of the genetic algorithm

3.1.1 Pseudo-code

A genetic algorithm or G.A is an algorithm that is used to tune certain traits in populations and arrive at a certain solution, best or accepted in a certain amount of time. It has the advantage of being able to look for solutions inside a wide solution space without needing a lot of information and being able to maintain a good performance.

3.1.2 Genetic representation

The reason why some problems require G.As is because the problem is way too complex to be solved using traditional methods. As stated in the introduction, these types of algorithms are commonly wielded to find solutions in a large search space and maintain a good execution time. However, G.As have a technical limitation. We have to represent all possible solutions of our problem with a linear representation. (for example, an array or a list) In consequence, we have to represent our DNA in a linear representation because it is the variable that we want to evolve. An agent's DNA is an array of size 8 that can only contain 0 and 1. We could fill it with numbers from R (real numbers) instead of just the {0, 1} set. Putting real numbers inside the DNA instead of 0 and 1 could possibly cause more harm than good because of two main reasons

- Truly random numbers are impossible to generate based on the deterministic nature of the algorithm used to create them in Unreal Engine 5[UE5Random]
- It would make the solution space go from 2^8 (256) to infinite.

The real-life counterpart of DNA can only take 4 possible values, which are adenine (A), cytosine (C), guanine (G), and thymine (T). These bases form specific pairs (A with T, and G with C), which makes the possibilities even lower. A possibility would be to change the set {0, 1} to {0, 1, 2, 3} but that would be something to discuss more in the next parts of the paper.

3.1.3 Fitness function

We implement a fitness function, which is simply a function that returns how close a solution is to the optimal solution of a problem. At the evaluation time in our simulation, we use the survivability time of alive agents as a fitness function. The survivability score is calculated based on the following formula:

$$\begin{aligned} \mathit{SurvivabilityScore} = (&1.5 \cdot \log_2(\text{Speed}) + \\ &3 \cdot (\log_{10}(\text{Size}) + \log_{10}(\text{Health})) \\ &+ 3.5 \cdot \text{Damage}) + \text{SenseBonus}; \end{aligned}$$

The different constants {1.5f, 3.5f, 3} were chosen and assigned a weight to underline the importance of each

value into the hypothetical chance of survival. We also added a SenseBonus to the equation of the SurvivabilityScore since it's theoretically accurate to hypothesize on the fact that better senses is equal to a survival chance for an agent. It is calculated using the following formula:

$$SenseBonus = \frac{\text{SightRange}}{\text{SightRadius}} \cdot \text{SightOffset} \cdot \left(\frac{1}{\text{HearingThreshold}}\right);$$

The ratio (SightRange / SightRadius) is used to perform the sphere trace function. A high ratio indicates a wide field of vision. However, a larger radius will increase the time needed to perform the trace, leading to a lower bonus. The higher the SightOffset is, the higher the bonus will be. This takes root in the fact that height is an advantage that can lead to agents being able to detect enemies as they approach as well as find resources easily. Finally, the (1.f / HearingThreshold) is such as when an agent's hearing ability is good, the higher the bonus will be.

3.1.4 *Termination criteria*

If a termination criteria is not implemented for a G.A, it will continue to run indefinitely because it wouldn't know what solution to look for in a given problem. In a problem such as ours without "conventional solutions", a termination criteria could probably be the number of generations, which could be changed between runs). The reason for such criteria is because of the intent behind our study. How genetic information would get transferred between generations and which combinations were the most interesting to keep by nature's law ? To avoid having the same solution over and over again, It could also be a good idea to implement a convergent protection. That means to stop running the algorithm if the N+1 generation's DNA set is very close to the N generation's DNA set.

3.2 Genetic operations

When new offspring are created in our world via reproduction, they undergo multiple stages and multiple processes until reaching birth. The real-life processes are vastly numbered and not the subject of this paper. However, the G.A is rendered useless if some of those processes are not reproduced in a virtual environment. They are called selection, crossover and mutation in a genetic algorithm.

3.2.1 *Binary Tournament*

After the evaluation is done, we have to select suitable parents for the crossover (C.R.O) step. In classical genetic algorithms, there are two methods to select parents. The "roulette wheel" method, which follows the "survival of the fittest" concept and consists of giving a probability of choosing a parent based on the ratio of its fitness to the sum of all fitnesses in a generation. This raises an issue of selecting only the best solutions in a set, which might lead to getting a suboptimal or not optimal solution for a given problem. The other method is called the binary tournament (B.T). It simply consists of taking two random solutions from the set and comparing the return value of their fitness function. The solution with the higher fitness will end up winning and get chosen. The paper [Eib01a] describes the selection process as a way to exploit current solutions and improve their fitness function. We changed its implementation to suit our need of having two parents/ two children and maintaining the population's numbers.

3.2.2 *Uniform crossover*

In real-life, crossover would happen when same type chromosomes meet during meiosis. Both can switch different parts when they are lined up. In our virtual world, we will avoid this method since our chromosomes do not have the same length as the real life DNA. C.R.O is crucial to G.As, it explores the set of solutions and looks for new ones that weren't available beforehand. In our paper, we shall use the uniform crossover. It separates the linear representation we evolve into n bits and randomly gives one of the parents' bits to the first child and the other parent's bit to the second child (based on a uniformly generated number). Therefore, such a method ensures we have explored more solutions. If we only had the selection process, it would mean staying in an area where the fitness might converge to a sub-optimal value.

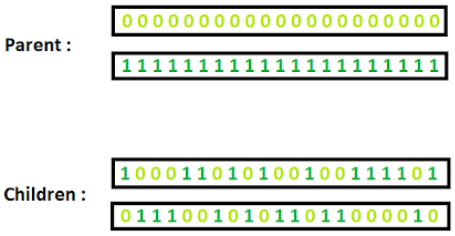

Figure 3: Graph of the uniform crossover with two parents and two offsprings

3.2.3 *N-point mutation*

Mutation in biology is described as the alteration of the DNA sequence of a bacteria, an organism, etc. It is said to be able to create changes in one's phenotype (observable characteristics). Nature of the changes are to be determined during the life of the individual whose DNA has changed. In traditional genetic algorithms, there are a few methods to reproduce the mutation (M.U.T) step (in a binary linear representation) such as the random mutation, which randomly selects a bit and changes its

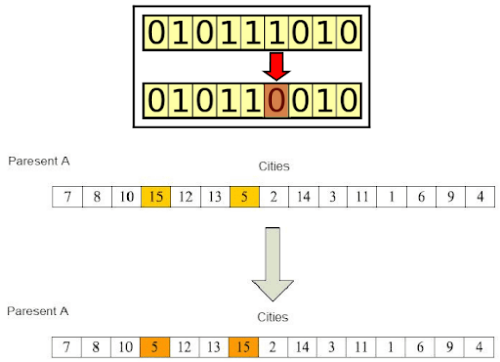

Figure 4: Comparison of the random mutation (binary-coded chromosomes) [Av01a] and the swap mutation (decimal-coded chromosomes) [Shih01a]

value, the swap mutation which selects two genes randomly and switches their value.

These mutation methods can bring new information to light but we consider them quite limited for tuning our agent's DNA. That being the case, we developed a novel yet simple algorithm called the n-point mutation which steps are developed in Algorithm 3.

Algorithm 3 N-point mutation algorithm

```
Get an agent's DNA
if length of the DNA not reached then
    Generate a uniformly distributed float between 0 and 1
    if generated float > 0.79 then
        change the value of the bit
    else
        Go to the next bit
    end if
end if
```

Keeping the threshold to a high value (0.79) helps us create a balance between creating new genetic representations for the offspring (exploration of the solution search space) and preserving the DNA structure from an unnatural disruption. We used a derivation of [Hol01a] to be able to fine tune and observe how a DNA structure would evolve through time and a limited amount of generations. Our agents' DNA are represented by a binary array of size 8, which could be changed. We evaluate them with a score that depends on their theoretical aptitude to survive in their environment, select them via B.T and reorganize them in an array for C.R.O, ensuring protection from premature convergence to a suboptimal solution. We apply the N-point crossover method as well as the N-point mutation with the hope of discovering more solutions during our run. To gap between reality and our world by changing the representation of the DNA to a linear array with {0, 1, 2, 3} as the possible genes instead of a binary array.

4 RESULTS AND FURTHER POSSIBILITIES

Creating the simulation was half of the process. The other half is to run some experiments with it. In order to obtain results with our G.A, we created a code which saves the agent species, survivability score, generation and DNA structure. In the following sections, we will analyze and interpret the data, discuss the results, and draw some observations. We would also like to define the next possible steps for this research, keep what is positive and understand where the simulation could be improved. (More data is available in the full thesis [Hba01])

4.1 Analysis of the results

These runs were made by changing the formulas and making the genes more significant for each attribute, and we did runs for four cases

- 75t and 150t with a M.U.T threshold of 0.79
- 75t and 150t with a M.U.T threshold of 0.83

We hope to see if making mutation a rarer occurrence would make it easier for the population to stay stable or if it would make it harder to survive. In this paper, we will only cover the most interesting cases out of the 4 : 75t - 0.79 M.U.T and 75t - 0.83 M.U.T.

Figure 5: Fox and deer populations: Gene comparison and individuals evolution (75t - 0.79 M.U.T)

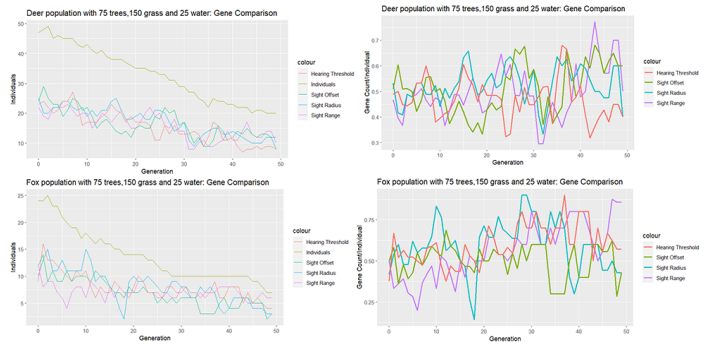

Figure 6: Fox and deer populations: Gene comparison and individuals evolution (75t - 0.83 M.U.T)

In the 75t - 0.79 M.U.T graph, the deer species seems to be thriving with a count of approximately 26 agents by the last generation. We also observe an augmentation of population as well as a higher count of deers individuals possessing the 3 genes for Sight just before generation 50. However, we also notice that, at the same time, the gene for Hearing gains a higher gene count/individual (g.c/l) too. A hypothesis could be that the M.U.T threshold is too low and makes gene mutation too common in this case. The fox species doesn't seem to be thriving compared to the deer species as their final numbers decreased by 72% (25 agents to 7 agents). We can interpret this as the result of competitiveness between the species and the foxes themselves (inter and intra species competitiveness). Furthermore, we note a decrease in population multiple times as the count of sight genes drops in the foxes' generation DNA. By generation 41, the number of individuals seems to stabilize, as the number of environment actor (E.A) always stays the same and the number of deers also decreases, leading to a better distribution of resources between all the agents in the 75t case. In the 75t - 0.83M.U.T case, the deer species doesn't thrive as well as their 75t - 0.79M.U.T counterpart. Their final population numbers decreased by approximately 25.93%. Gene count/individual pattern seems to be close to the 75t-0.79M.U.T case, except for the Hearing Threshold gene, whose g.c/l decreased drastically because of the lower mutation rate. The fox species seems to be thriving more in this case than their 0.79M.U.T counterpart. In fact, the hypothesis that sight genes count/individual and survival rate was linked tend to be validated by our data as we can see in Figure 7. In generation 10, we had an increase of $\approx 14.3\%$ for the fox population as well as an increase of 23.52% of Sight Range gc/l. Having mutation as a rarer occurrence could be beneficial for the system in its whole as it maintains balance between the two species used in the simulation. Making it a more common occurrence seems to tilt the favor to stronger species (deers) as the weaker species will not be able to focus on genes that should close the gap between them. It could also be conceivable that a higher number of resources isn't necessarily the best as their distribution in a system should be taken into consideration too. Foxes in the 75t - 0.79 M.U.T showcases that hypothesis. We will try to discuss more on the data, about other hypotheses and various possibilities for this research.

4.2 Discussion

The research is conducted to gain understanding about which genes are the most significant in our DNA structure, depending on the environment and other parameters. In the previous part, we analyzed the gene count/individual (g.c/l) for a few cases but we didn't focus on the significance of the genes.

< 0.001 : xxx	< 0.01 : xx	< 0.05 : x	> 0.05 : X

Table 2: Legend of the significance levels used in the following table

M.U.T	Fox 75	Fox 150	Deer 75	Deer 150
SRange	X	xx	xxx	xxx
SRadius	X	xxx	xxx	xxx
SO	X	x	x	xxx
HT	xx	xxx	xxx	xxx
ASize	X	X	x	X
ASpeed	X	X	X	X
AD	X	X	X	X
AH	X	x	X	X

Table 3: Fox and deer populations : Gene significance (second runs - 0.83M.U.T)

We note that AgentSize, Speed, Damage and Health are the most significant. It can be explained as all those define how an agent will survive in its environment, find resources and fare against its enemies. A real life example could be the gazelle and the lion. A lion might be stronger than a gazelle but a gazelle can outrun a lion on long distances. All sight genes significance for deers are inferior to 0.001 except for SightOffset in the deer75 (is inferior to 0.05). For the fox75, sight genes all possess a significance higher than 0.05. That could be explained by the scarcity of the resource and the competition the species has to go through with deers. Fox150, doesn't seem to possess that much significance for SightRadius (inferior to 0.001) and SightRange (inferior to 0.01) as the resources are abundant and fairly close to the foxes. The lower distance between foxes and resources could also explain why its AgentHealth significance went from X to x in table 3. Less health and less energy is needed to find resources. The Hearing Threshold gene always has a significance inferior to 0.001 (except for fox75 in 0.83 M.U.T). This gene, no matter this value, will never hold a significance in this version of the simulation as the hearing sense of agents wasn't replicated. In real life, foxes have a very sensitive hearing and use it to gather information about their environment. It enables them to hear sounds from behind, better than a deer, giving them a clear advantage during hunting. During all the runs, no combat happened between prey and predator as each agent teleports to its next location and never managed to get closer to its enemy. If agents moved in a traditional way, we could probably observe multiple foxes teaming up to chase one or multiple deers, cornering them and defeating them in battle. Agents shouldn't be dying as combat doesn't happen. Nevertheless, population numbers decrease as a result of an U.E5 bug where the calculation of a random position inside the navigable radius leads to the void and an automatic destruction of the falling agent by the engine.

4.2.1 *Outlook and Future Directions*

In this paper, we use a binary representation for the agents' DNA, to simplify operations. As stated in that part, life doesn't represent genes with a {0, 1} set but with a {C, G, A, T} set. The letters that creates the real DNA are nucleobases, which can only link in a certain way :

- Adenine (A) with thymine (T)
- Cytosine (C) with guanine (G)

Having 8 genes with {0, 1} as the values is simple but limited, as we only have 256 possible combinations for all agents. Using a numbered set like {0, 1, 2, 3} could be the solution as it would make the number of possibilities equal to

$$4\text{^}8 = 65536$$

However, DNA is a polymer in the form of a double helix where nucleobases only link a certain way as stated before. Replicating the double helix form would mean using a non linear representation for the AT, CG pairs. It is impossible to keep using a G.A for this kind of representation as a G.A is only usable for linear representation per the definition of its operators (selection, crossover, mutation). We could use a set equivalent to {AT, CG} in order to achieve that with a G.A but it would basically lead to the same result as using {0, 1} as our gene set (256 combinations) It is possible to completely change the DNA structure to get more combinations but that would mean changing how we calculate the A.A. It is unnecessary to modify the DNA as {0, 1} can be considered like {AT, CG} and the mathematical model seems to hold itself vis a vis the ecological dynamic patterns for our species survival. A direction the study should take is the correction of the agent teleporting, which would lead to more combat between species and hence more data concerning the predator/prey problem, as well as refining the agent attributes calculations. In the current state of the simulation, we overlook certain complexities real organisms face, simplifying interactions between agents. Moreover, we didn't develop parameters that have the potential to turn around the ecological dynamics in our simulation such as temperature variations, diurnal cycle, season changes, natural catastrophes, diseasesand the effect of human activities on the environment. Those should prove themselves extremely complicated to replicate in the U.E5 engine but could have a really interesting effect on our results and should be taken into consideration for the future of this research. It is also possible to expand the range of species in the simulation as only two species exist in the virtual world for now. We hope that bringing new species with unique genetic traits and behavior will help us observe more complex population dynamics and ecological relationships. Another key component for this research is the investigation of the role of mutation and its effect on survival and evolutionary process. Exploring different thresholds (example : 0.79 M.U.T, 0.83 M.U.T) and adding other types of mutation could enable us to gain understanding on the M.U.T operator's influence on survival, persistence of species and genetic variations over time.

5 CONCLUSION

The problematic behind this research was studying the evolution of agents inside a virtual world. We sought to understand the mechanisms of DNA transfer between generations, identify the genes of utmost significance and pinpoint parameters influencing survival and gene significance. Those were some of the questions we asked ourselves while working on this project. The methodology used to respond to these questions and the problematic were to create an artificial life simulation in Unreal Engine 5, which aims to mirror real life, reproduce animals' traits and behavior, run a genetic algorithm with binary tournament, uniform crossover and n-point crossover on each generation and analyze the data to find what genes hold the most significance per case. Analyzing the collected data during multiple runs enabled us to understand which parameters influence the most survival and a species' success in our virtual environment. Some species would thrive more than others in the virtual life environment, and thus due to recreating or not senses like sight and hearing. An example of this is how deers always managed to keep between 60% and 40% of their initial population whereas foxes always kept between 40% and 20% of theirs. It is explained by how recreating the hearing sense in Unreal Engine 5 showed itself to be a challenging task compared to the sight sense's recreation. In our simulation, deers were bigger than foxes and held a better sight sense than them, hence giving them an advantage when searching for resources or finding predators in case of danger. The analysis of the results denotes how important Sight genes were during all the runs. Resource availability and the absence of competition internally and externally of a species is a key factor for survival and a species' success. We noted that mutation, as a process, is important to create genetic diversity but a low threshold for mutation would impact agents' life and thus, survival in a negative way. Table 3 shows other genes held a lot of significance - AgentSize, Damage, Speed and Health. The genes are the ones defining how well an agent interacts with its environment (A.Size) and enemies (A.Speed and A.Damage) and how long it can search for food before needing to rest (A.Health). While the simulation, collected data and their analysis may bring to light some new information about ecological dynamics and artificial life simulation, we should acknowledge the limitations of our research. Agents do not move in our simulation, they teleport, which causes a problem in

terms of chasing/running away between predator and prey. Agents' behavior is quite simplistic and could be improved to create more cases to study. Our formulas for calculating agent attributes remain quite simple, refining or modifying them might be a route to pursue in this research's future. The paper [Ric01a] created an artificial life simulation where certain parameters like predator/prey awareness, flocking or herding can be enabled or disabled. It focuses on the cycle of life and death and the predator/prey problem whereas ours focuses on DNA transmission and evolution in a virtual world. Implementing these parameters in the future of our research, as we implement newer species, could lead to a better and deeper understanding of ecological dynamics and predator/prey problem. This research endeavored to explore the complex concepts of evolution and ecological dynamics using UE5 and artificial life simulation. Mimicking animals' behavior and the environment in a realistic manner proved itself a formidable challenge, but results were satisfactory. The paper contributes to bringing new knowledge as it underlines the significance of resources availability, mutation threshold and sensory capabilities. Given how this research overlaps an array of fields, it has the potential to benefit our knowledge of evolution if resources and time were allocated to it.

ACKNOWLEDGMENTS

I would like to express my sincerest gratitude to a few people that helped me achieve my thesis.

- My French supervisor, Dr. Alexandra Bac.
- My Japanese supervisor, Dr. Okubo Masashi.
- Ms. Rhiannon Follenfant

I also thank my family and friends for their continuous support as well everyone else that was involved and helped me during this paper, no matter how much they were involved.

6 REFERENCES

[Hol01a] Holland J. Adaptation in Natural and Artificial Systems, 1975.

[Ric01a] Richards D, Jacobson M. J. Evaluating the Models and Behaviour of 3D intelligent Virtual Animals in a Predator-Prey Relationship, 2012.

[Geo01a] Georgiev M, Tanev I, Shimohara K, and Ray T. Evolution, Robustness and Generality of a Team of Simple Agents with Asymmetric Morphology in Predator-Prey Pursuit Problem, 2019

[Eib01a] Eiben A. E, Schippers C. A. On Evolutionary Exploration and Exploitation, 1998

[Hba01] Hassene Hba Ben Amar, Masashi Okubo. Study of Evolution in Virtual Worlds. Neural and Evolutionary Computing [cs.NE]. Doshisha University; Polytech Marseille, 2023.

[Shih01a] Shih-Hsin C. and Mu-Chung C., Operators of the Two-Part Encoding Genetic Algorithm in Solving the Multiple Traveling Salesmen Problem, 2011

[Av01a] One-point mutation figure: `https://www.analyticsvidhya.com/blog/2021/06/genetic-algorithms-and-its-use-cases-in-machine-learning/`

[cheetah01] Fastest animals on earth: `https://www.britannica.com/list/the-fastest-animals-on-earth`

[UE5] Unreal Engine 5 Documentation : `https://docs.unrealengine.com/5.0/en-US/`

[UE5Random] Random function documentation in UE5: `https://dev.epicgames.com/documentation/en-us/unreal-engine/BlueprintAPI/Math/Random?application_version=5.0`

[perlin01] Perlin noise `https://en.wikipedia.org/wiki/Perlin_noise`

7 ANNEX

- The source code of this project can be accessed via the link : `https://github.com/hassenebenamar/Research_Project`