NTNU
Innovation and Creativity

Massively Online Games with Food Chains

Thibault Collet

Master of Science in Computer Science
Submission date: July 2007
Supervisor: Anne Cathrine Elster, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

Two of the main complaints of the Massively Multiplayer Online Games (MMOG) community is the lack of interaction with the virtual universe, and the repetitative patterns of action of the game play. Introducing a virtual food chain for the targets, would add two new features to the game play. The genetic evolution of the opponents would make each fight unique. Furthermore, the players will have to coordinate their actions in order to keep a balanced alimentary chain.

As these kind of games are designed for many thousands of players in real-time, a lot of compute power will be needed to simulate the genetic evolution. Parallel computing can greatly improve the performance, assuming the software is properly designed with parallelizations in mind.

This project will focus on developing a model and prototype of a alimentary chain for MMOGs. Parallelization strategies, including parallelizations with respect to species and parallelization with respect to the map surface (environmental domain) will be discussed. Benchmarking and performances analysis will be done if time permits, hopefully, guiding further development in this area.

Assignment given: 23. February 2007
Supervisor: Anne Cathrine Elster, IDI

Massively Multiplayer Online Games with Food Chains

Thibault Collet

Master of Science in Computer Science
Submission date: 21st of July 2006
Supervisor: *Dr. Anne C. Elster*

Norvegian University of Science and Technology
Departement of Computer and Information Science

Abstract

MMOGs (Massively Multiplayer Online Games) are today a multi-bilion dollar industry where typically thousands of players interact in a virtual world. Two of the main complaints of the MMOG community are the lack of interaction with the virtual universe, and the repetitive gameplay patterns. Introducing a virtual alimentary chain for the targets would allow for a much more exciting experience since the genetic evolution of the opponents would make each fight more unique, and collaboration among player community would be necessary to keep a sustainable balance in the virtual world.

As these kind of games are designed for many thousands of players in real-time, a lot of compute power will be needed to simulate the genetic evolution. Parallel computing can greatly improve the performance, assuming the software is properly designed with parallelizations in mind. A couple of different schemes will be considered.

Trying to fetch the ecology studies, a prototype architecture is presented here. A non-linear genotype-phenotype transformation mechanism (also called *morphogenesis*) has been designed with the purpose of obtaining Lotka-Volterra equations result. These research will lead to discussions and conclusion, hopefully guiding further development in this area.

Acknowledgment

I would like to thank my advisor, Dr. Anne C. Elster, for not only believing in my ideas and for also giving me the opportunity to realize them, but also for her invaluable support and advise. She was always ready to discuss new ideas, she always gave me insightful and very competent comments. She has been an important guide (not exclusively technically speaking) and I'm really grateful to her.

I would also like to thank the whole High-Performance Computing Research group. The regular meetings held each Friday were very instructive. More than that we had a lot of enjoyable collaborations and interactions.

It is not easy to spend one year of study abroad, and I sincerely want to tell them all that this semester would not have been the same without them.

Table of Contents

List of Figures

List of Tables

Massively Multiplayer Online Role Playing Game Definition :

"*Massive(ly) multiplayer online role-playing game (MMORPG) is a genre of online role-playing video games in which a large number of players interact with one another in a virtual world.*

As in all RPG's, players assume the role of a fictional character (most commonly in a fantasy settings) and take control over many of that character's actions. MMORPGs are distinguished from single-player or small multi-player RPGs by the number of players, and by the game's persistent world, usually hosted by the game's publisher, which continues to exist and evolve while the player is away from the game."

Wikipedia.org

Chapter 1

Introduction

Massively Multiplayer Online Gaming (MMOG) industry has been growing up very fast these last years (see figure 1). Originally indented for a narrow category of players, current large public games like *World of Warcraft* (Blizzard Entertainment,2004) or *Second Life* (Linden Lab, 2003) made of these new games the future of computer video games industry. Nowadays almost each Video game developer company has produced one MMO or is doing so.

After a period where the same game schema has been reproduced over and over, players are looking for new gaming experiences. Developers hence have to invent new way of interacting with virtual environment. The new generation consoles are facing the same problem. *X-box* (Microsoft) and *Playstation 3* (Sony) which settle for improving performances (better graphics, artificial intelligence) are overtaken by the *Wii* (Nintendo) which is quite limited in performances but offers a new way of interaction (a controller able to capture the player's movements).

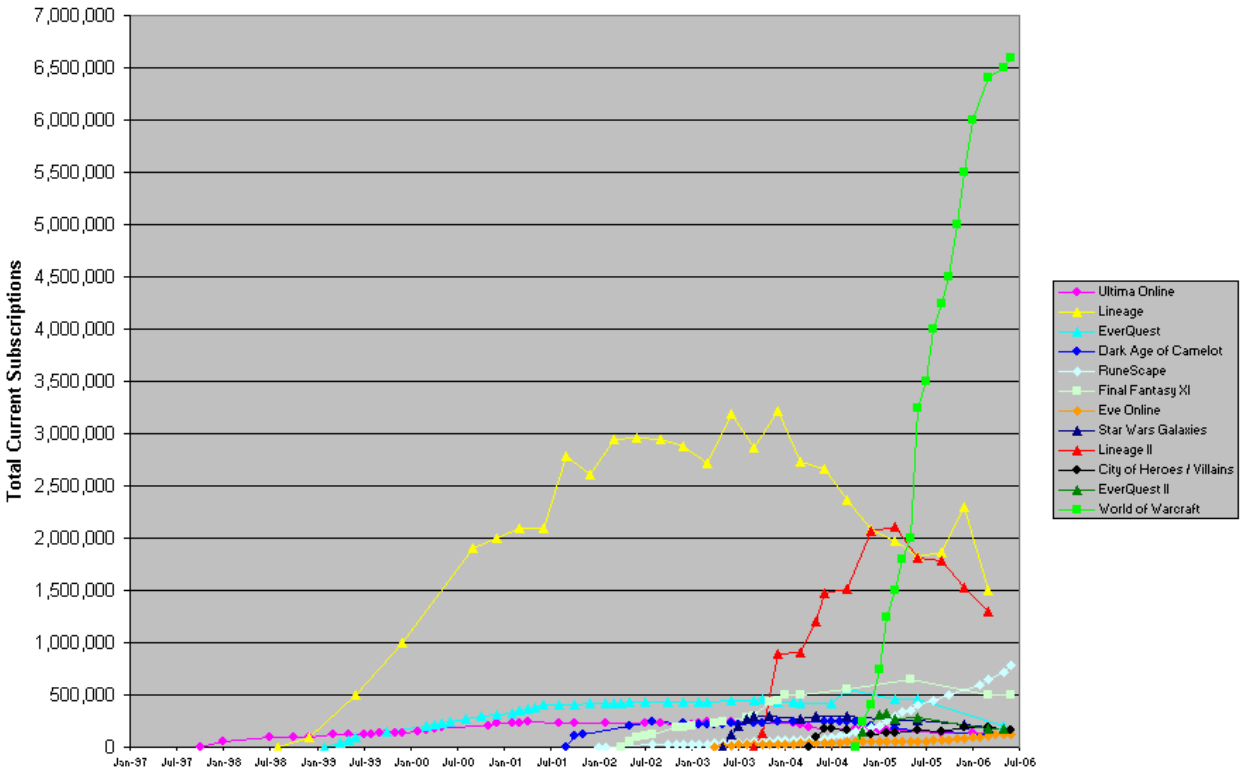

Figure 1 : MMOG Active Subscriptions (from www.mmogchart.com [1])

In both *World of Warcraft* and *Lineage 1 & 2* (NCSoft, 1998 & 2003), the two major actors of the market today, the gameplay is based on the player's evolution. The player's avatar (in-game character) has to defeat monsters to gain experience, thus allowing him to use new abilities and equipment.

Originally coming from role-playing game, these roots where slowly lost at the expense of production based on character evolution. While some players seek more power, others are more willing to build something. *Star Wars Galaxies* (Sony Online Entertainement, 2003) was a success thanks to the possibility of creating (and decorating) houses, towns, and even taking part of the galactic war. In one word : “having some influence on the universe”.

This paper try to introduce the concept of bringing food chains in these virtual worlds.

1.1 Thesis outline

Chapter 2 discusses the roots of MMOGS, their evolution, and discusses players motivations as well as what kind of both gameplay innovations and limitation should we can expect from adding food chains. We also get into the famous question“How to model life ?” and a general discussion about food chains.

Based on former studies,we develop our model and prototype game incorporating food chains in Chapter 3. This chapter includes the basic rules, the behavior state evolution on the non-linear phenotype/genotype mechanism, as well as the architecture of the prototype of our model and some simulation results.

Synonymous of speed-up performances, massively architectures, parallelization was kept in mind since the very beginning of prototype's development. Therefore, Chapter 4 focuses on parallelization discussions for such an implementation. Conclusions and future work are given in Chapter 5

1.2 Hardware and software environment

When “performances”, “supercomputer” or “cluster” are part of the project's vocabulary, Unix based operating system is hanging around. Due also to my interest for open source philosophy I decided install Linux on my working station (Distribution : Ubuntu Edgy 6.10 [2]). As it contains an Intel Core Duo T2300, the shared memory test could be done with OpenMP [3]. For some trials relative to cluster computer, I had access to “Clustis2” from NTNU (composed of 22 Dell PowerEdge 750 nodes).

Due to the special structure of genetic algorithms, object oriented languages widely used. Even if my previous experience in GA was based on Java, I decided here to focus on C++ for the following reasons :

- Seek for performances
- Precise memory management
- Parallelization in mind (OpenMP, MPI, ...)
- Video games area project

Chapter 2

Background's picture

2.1 Massively Multiplayer Online Role Playing Games

2.1.1 Roots of MMORPG's

Originally role playing game are located around a table, using only pen, paper, and imagination (dice could also introduce a random feature). In 1978, with the Internet revolution, Multi users Dungeons offered the opportunity to play with players from anywhere in earth, hidden from them behind the screen .Typically running on a bulletin board system pr Internet server, the game is usually text driven, where players read descriptions of rooms, objects, events, other characters, and computer-controlled creatures or non-player characters (NPCs) in a virtual world. Players usually interact with each other and the environment by typing commands. The evolution of computer games fastly improved the immersion in the virtual universe due to mainly a nice and precise graphic Avatar, the size of the servers (the *Massively* of *M*MORPG), the content (number of quests, ...), the interaction with the environment (introduction of players crafted items, players town, event realization...).

2.1.2 Evolution of the genre

The first 3D interfaced MMORPG success was *Neverwinter Nights* (Bioware 2002). Based on the multi users dungeons schema, one player (the Game Master) create a dungeon with quests for few players. Each team is isolated in a universe completely independent from another. The rules were based on the most famous Pen and Paper game : *Dungeon and Dragons* (Wizard of the coast, 80's). *Ultima Online* (Origin Systems, 1997) introduced the persistent universe notion, the craft and the housing. Despite his old graphics, this game is considered by many as the first MMORPG as we know them nowadays. Since the introduction of these gameplay innovations few evolution has been made, each production settles for improving its mechanisms.

When we look closely at *World Of Warcraft* which is considered as the best MMORPG mainly due to his success (see figure 2), we find severals gameplay qualities, but coming from previous generations. The success is more attributed to its graphic style, ease of use, very good balanced among the classes, or even the market policy of blizzard than to innovations. The inherent purpose of this work try also to explore gameplay's innovations, with the hope of guiding further research in this area.

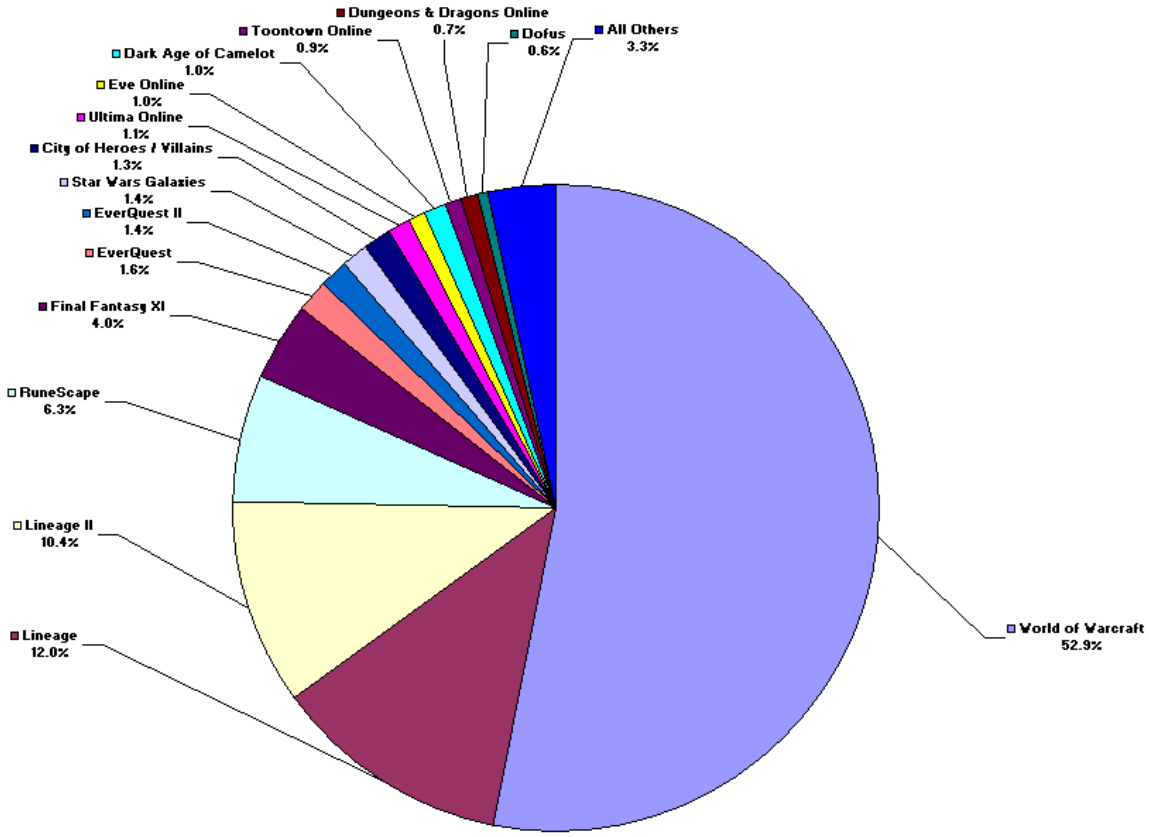

Figure 2 : MMOG Subscriptions from www.mmogchart.com [1]

2.1.3 Players motivations

After all, if we don't understand why players are in these on-line worlds to begin with, then we can never truly appreciate the more complex phenomena that emerge from these environments.

Richard Bartle's vast experience with game economies is now available in his textbook [4]. His theoretical model divide players motivations in 4 categories :

- **Achievers** : These players regard points-gathering and rising in levels as their main goal. They are more orientated through "Player versus Environment" part of the game
- **Explorers** : Players try to find out as much as they can about the virtual world. They delight in having the game expose its internal machinations to them.
- **Socialisers** : Players use the game's communicative facilities, and apply the role-playing. Socialisers are interested in people, and what they have to say. The game is merely a backdrop, a common ground where things happen to players. It's the categorie of players the most attracted by Role-playing
- **Killers** : Killers get their kicks from imposing themselves on others. They seek for powerful reputation, respect from other players. They are more orientated into "Player versus Player" game

According to Nick Yee who realized a deep factor analysis and questionnaire-based research [5] among on-line players, the graphics of a game make it attractive, but the gameplay underlying is the part of the game that keeps players in one universe.

"M*any of us enjoy the "wow" factor afforded by the visual impact of the newest and most graphically advanced game – this facet clearly sells games. Visual realism is only a part of what makes a game world and the characters in it believable, if any aspect of the game shatters our immersive gameplay experience and we are less able to suspend disbelief within the game world. In other words we may have a beautifully created wall using the latest vertex and pixel shader programs to enhance the illusion of the game world existence, but the illusion is shattered when our supposedly intelligent character continually bangs his head off the wall in an attempt to get round it! This is only a simple example that we can all relate to, especially if we play games of the RPG (role play game) genre, but there are many other examples of AI behaviours in games that have a negative impact in our immersive gameplay experience"*

Nick Yee [5]

2.1.4 Gameplay influences of an Artificial Food Chain

"The point is innovation of AI can lead new game design formats and gameplay scenarios."

Darryl Charles [6]

Innovative AI approaches may bring, more believable AI, flexible character behavior, or enhanced graphics, but in general the prospective game player will not care about these improvements except they are an integral part of what makes the game enjoyable to play [7]

Most part of actual productions focused on Achievers and Killers players. The AI is reduced to the least possible, all the informations about the environment is available on the Internet reducing greatly the part of exploration. It is mainly due to the fact that Achievers and Killers are easier to keep on one game. Adding new contempt for these players means introducing new top level, adding new weapons or special abilities.

The socialisers and explorers are therefore strongly criticizing actual games, waiting for a game where the "pex" or the "grind" (fact of repetitively kill monster using always the same pattern) does not take place. According to specialized forums (www.mmorpg.com, www.mondespersistants.fr, www.jol.fr) there is a huge part of the community waiting for the new generation of online games where interactions with the environment reach a higher level.

I think food chains can be a solution. The consequence of an improved game AI design and implementation for the player is that the game provides a more rewarding and interesting challenge than it would have done otherwise. As animals evolve genetically, each fight differs from the previous one, therefore the player has to adapt himself to the special characteristics of his opponent. Furthermore, in order to keep a balance environment, players had to coordinate their actions, even if they are part of opposite factions.

It also raise some issues. First of all, if Artificial Intelligence is so poor in this kind of games, it is due to the tough job of creating an already simple world fully functional. Indeed, when coming out, most part the games are quite not finished and need a lot of improvements, bug fixing. Furthermore, a "game master" team has to work on the stability of the ecosystem.

2.2 Artificial Life

The main motivation of Artificial Life is to model and understand the formal rules of life. ALife is life created by man, not by nature. Thanks to our technological improvements, we are now able to create "living" artifacts. It means these special life form will gather all the main characteristics for something to be "living" at our eyes scale which is of course not exempt of issues [8].

> *"The field of Artificial Life is devoted to studying the scientific, technological, artistic, philosophical, and social implications of such an artifact. Alife complements the analytic approach of traditional biology with a synthetic approach: rather than studying biological phenomena by taking living organisms apart to see how they work, we attempt to put together systems that behave like living organisms."*
>
> C.G Langton

2.2.1 Linear and Non-linear Systems : the phenotypes / genotype issue

As mentioned by Christopher G. Langton [9], the distinction between linear and nonlinear systems is fundamental. More than that, it explains why the principles underlying the dynamics of life should be so hard to discover. In brief, a linear systems is one where the behavior of the whole is just the sum of the behavior of its parts. They obey to the principle of superposition. If we analyze these parts independently we can get a full understanding of the whole system. On the opposite side, nature is generally non-linear. It means, form of life do not obey to the principle of superposition. Even if we divide this type of system into simpler components, a full study of them will never give us a better understanding of the whole. The point is the primary behavior of a component take in account the interaction between the parts, not only the properties of the part itself. If we break up this system, the interaction (a major factor) disappear, letting us with an incomplete element to analyze. Nature is of course non-linear. This was the main motivation for the elaboration of Cellular Automate (one of the most basic but still very complex Artificial Life form).

" It consists of a regular grid of cells, each in one of a finite number of state. The grid can be in any finite number of dimensions. Time is also discrete, and the state of a cell at time t is a function of the states of a finite number of cells (called its neighborhood) at time t − 1. These neighbors are a selection of cells relative to the specified cell, and do not change (though the cell itself may be in its neighborhood, it is not usually considered a neighbor). Every cell has the same rule for updating, based on the values in this neighbourhood. Each time the rules are applied to the whole grid a new generation is created."

Definition of Cellular Automata from Wikipedia.org

In the 1970's a two-state, two-dimensional cellular automaton named *Game of Life* became very widely known, particularly among the early computing community. Invented by John Conway, and popularized by Martin Gardner in a Scientific American article [10], its rules are as follows: If a black cell has 2 or 3 black neighbors, it stays black. If a white cell has 3 black neighbors, it becomes black. In all other cases, the cell stays or becomes white.
Despite the very simple laws applied to each cell, the complexity of different pattern has been studies since years.

It is obvious to state that Game of Life is a perfect example of a non-linear system where one cell's state at a 't' time, is based on the neighbors' cell at 't-1' .

An other illustration of the non-linearity of Life is the phenotype/genotype distinction. The genotype is the complete set of genetic instructions encoded in the sequence of nucleotide that makes an organism's DNA. The phenotype is the expression of the genotype. It represent the physical organism itself. The process by which the phenotype develops through time under the direction of the genotype is called *morphogenesis*. The interaction between both is fully non-linear[11]. The main example is the different influence a mutation can occur in the phenotype (topic studied later in this paper).

We can talk about the phenotype trait associated with a particular gene. We can talk about phenotype for a particular cell, but also for an entire multi-cellular organism. Phenotype is complex and multi-level. If we want to simulate life, we need to reproduce this hierarchical structure. In general, prototypic traits at the level of the whole organism will be the result of many nonlinear interactions between genes, and there will be no single gene to which one can assign responsibility for the vast majority of prototypic traits.

2.2.2 Unpredictability of phenotype from genotype

Nonlinear interactions between the genes provide gives us an extremely rich variety of possible phenotypes. The other side of the coin, however, is that in general, we cannot predict the phenotype that will emerge from specific genotype, due to the general unpredictability of nonlinear systems. Therefore, a trade-off exists between behaviorally rich but essentially unpredictable nonlinear systems on the one hand, and behaviorally restricted but predictable (and therefore "programmable") linear systems on the other.

Furthermore, in the real life it is not possible to determine what specific alterations must occur in the genotype to effect a desired change in the phenotype level. The problem is that any specific phenotype trait is, in general, a result of a certain number of nonlinear interactions between the genes. An example from biology would be: What changes would have to be made to the human genome in order to produce six fingers on each hand rather than five ?

Based on this, we have a better understanding of how the prototype need to be realized. In order to be closer to real life, there is some rules the prototype has to follow (genotype/phenotype, non-linear system, multi-level phenotype). But on the other side, the final purpose of the prototype (video game integration) give us some heavy constraint we need to face of (as we cannot predict the phenotype from a given-genotype).

2.2.3 Artificial Intelligence

Artificial life and artificial intelligence are strongly linked together. But for some reasons, the field of intelligence is older. Recently both field where mixed with sub-symbolic programing : Genetic Algorithms, Neural Networks, Ant Algorithms... But still both field keep their own purpose.

Artificial Intelligence is concerned with generating intelligent behavior. It focuses on the problem of creating behavior generators. However, although it initially looked to natural intelligence to identify its underlying mechanisms, these mechanisms were not known (nor are they today). Therefore, the technology of serial computer programming was influencing (and influenced by) the algorithms. As a consequence, from the very beginning artificial intelligence created behavior that bore no demonstrable relationship to the method by which intelligence is generated in natural systems.

In fact, AI has focused primarily on the production of intelligent solutions rather than on the production of intelligent behavior.

By contrast, most of the mechanisms of natural living systems are known. There are still many holes in our knowledge, but the general picture is in place [9]. Therefore, Artificial Life can start by recapturing natural life. Furthermore, Artificial Life is not primarily concerned with building systems that reach some sort of solution.

Some AI technics called sub-symbolic are on the edge of these fields. Inspired by natural fact, mechanisms, selections, they are built with the ongoing purpose of finding a solution to a specific problem. But they still remain based on natural life and their algorithms are widely used in artificial life.

2.2.4 Genetic Algorithms

GAs were introduced as a computational analogy of adaptive systems. They are modeled on the principles of the evolution via natural selection, employing a population of individuals which evolve genetically due to operators such as mutation and recombination. A fitness function is used to evaluate individuals, and reproductive success varies with fitness.

The Algorithms [12]

1. Randomly generate an initial population M(0)
2. Compute and save the fitness u(m) for each individual m in the current population M(t)
3. Define selection probabilities p(m) for each individual m in M(t) so that p(m) is proportional to u(m)
4. Generate M(t+1) by probabilistically selecting individuals from M(t) to produce offspring via genetic operators
5. Repeat step 2 until satisfying solution is obtained.

The paradigm of GAs described above is usually the one applied to solving most of the problems presented to GAs. Though it might not find the best solution. more often than not, it would come up with a partially optimal solution.

As the reader may have noticed, we are talking here about “best suitable solution”. This is not the purpose of the genetic algorithm study here. An artificial ecosystem does not seek for a “solution”. What is relevant in GA's is the life modelisation part. Understanding the mechanism can be helpful for the prototype.

Three key part for this algorithm need to be focused here.

- The first one is the function responsible of the creation of a phenotype based on a genotype. As we saw above, the non-linearity of life impose us to keep a non-linear function. This is the first main difference with GA where linearity is not such a big matter.
- Secondly, the crossover and mutation mechanisms has the same restrictions. The mutation has to be randomly executed, and with equal probability of modifications. The phenotype needs to be fully linked with the genotype. During a crossover, a trait should be given to the child.
- Finally the selection is working following a completely different way. In GA, each individual of the population is evaluated through a fitness function. Based on this fitness value (and most part of the time, the average value of the population) each individual receive a probability to be elected for the next turn. In Artificial Life the selection is fully natural, and the programmer cannot act through a direct way on it.

Apart from these three modifications, the basics of GA are kept for virtual environment creation as such a model of life has already proved his efficiency[12][13].

2.3 Food Chains

A food chain shows how each living thing gets its food. Some animals eat plants and some animals eat other animals. For example, a simple food chain links the trees & shrubs, the giraffes (that eat trees & shrubs), and the lions (that eat the giraffes). Each link in this chain is food for the next link. A food chain always starts with plant life and ends with an animal.

1. Plants are called producers because they are able to use light energy from the Sun to produce food (sugar) from carbon dioxide and water.
2. Animals cannot make their own food so they must eat plants and/or other animals. They are called consumers. There are three groups of consumers.
    a. Animals that eat only plants are called herbivores (or primary consumers).
    b. Animals that eat other animals are called carnivores.
        - carnivores that eat herbivores are called secondary consumers
        - carnivores that eat other carnivores are called tertiary consumers e.g., killer whales in an ocean food web … phytoplankton → small fishes → seals → killer whales
3. Animals and people who eat both animals and plants are called omnivores.
4. Then there are decomposers (bacteria and fungi) which feed on decaying matter.

    These decomposers speed up the decaying process that releases mineral salts back into the food chain for absorption by plants as nutrients.

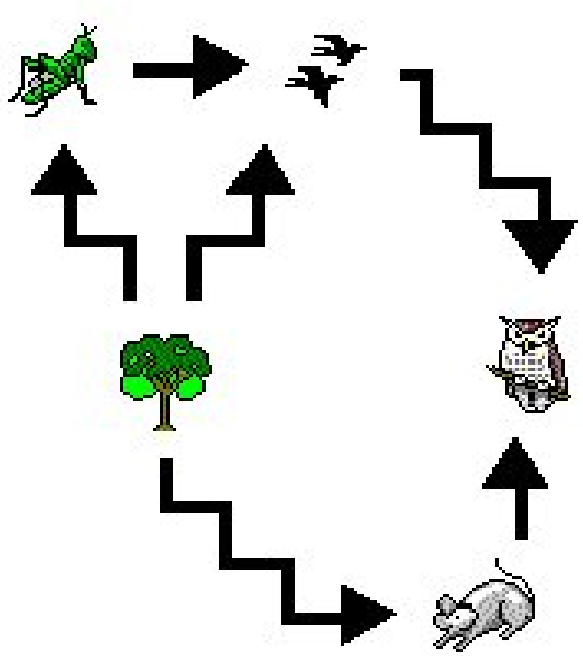

Figure 3 : Food Chain

This interdependence of the populations within a food chain helps to maintain the balance of plant and animal populations within a community. For example, when there are too many giraffes; there will be insufficient trees and shrubs for all of them to eat. Many giraffes will starve and die. Fewer giraffes means more time for the trees and shrubs to grow to maturity and multiply. Fewer giraffes also means less food is available for the lions to eat and some lions will starve to death. When there are fewer lions, the giraffe population will increase.

Above everything stands the human. He can act in two ways on a food web : directly and indirectly. First of all the human is one of the rarest species able to hunt and digest each life form. He can directly act as a population growth limiter or reducer (in the good as in the bad way). Indirectly the human can be a predator for species in the top of the food chain. For example, hawks have few, if any, natural predators, but they have enemies such as pollution, particularly from pesticides, and habitat destruction from developments. In short, humans are their main predators even if they don't hunt them directly.

In a food chain, energy is passed from one link to another. When a herbivore eats, only a fraction of the energy (that it gets from the plant food) becomes new body mass; the rest of the energy is lost as waste or used up by the herbivore to carry out its life processes (e.g., movement, digestion, reproduction). Therefore, when the herbivore is eaten by a carnivore, it passes only a small amount of total energy (that it has received) to the carnivore. Of the energy transferred from the herbivore to the carnivore, some energy will be "wasted" or "used up" by the carnivore. The carnivore then has to eat many herbivores to get enough energy to grow.

Odum analyzed the flow of energy through a river ecosystem in Silver Springs, Florida [14]. His findings are shown here. The figures are given in Kilocalories per square meter per year (kcal/m^2/yr).

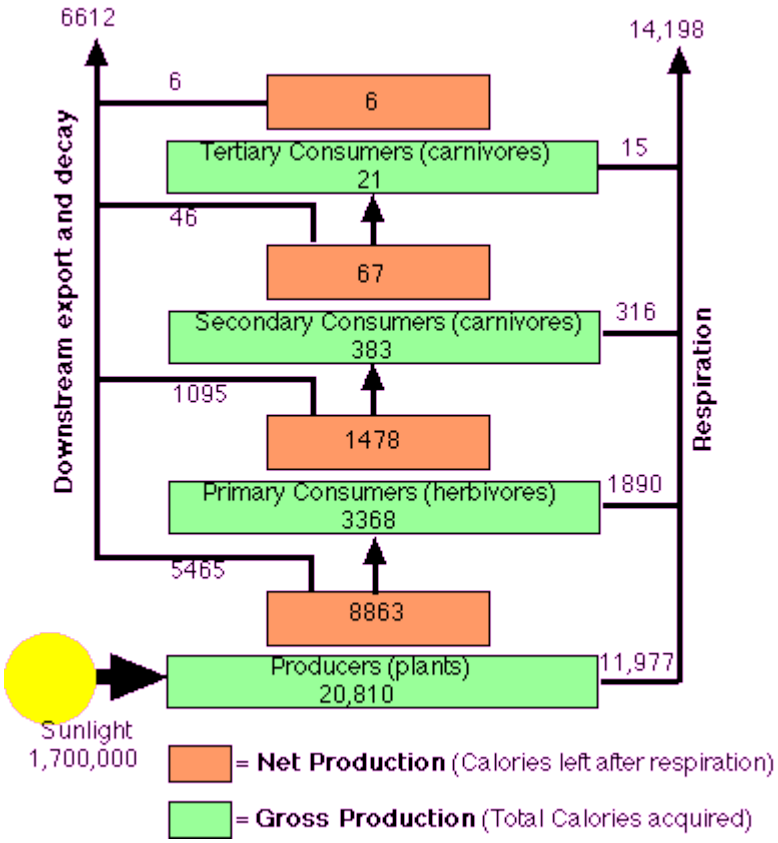

Figure 4 : Energy flow in food chain

At each trophic level,

- Net production is only a fraction of gross production because the organisms must expend energy to stay alive. Note that the difference between gross and net production is greater for animals than for the producers - reflecting their greater activity.
- Much of the energy stored in net production was lost to the system by
    - decay
    - being carried downstream
- Note the substantial losses in net production as energy passes from one trophic level to the next.
- The ratio of net production at one level to net production at the next higher level is called the conversion efficiency. Here it varied from
    - 17% from producers to primary consumers (1478/8833) to
    - 4.5% from primary to secondary consumers (67/1478).
- From similar studies in other ecosystems, we can take 10% as the average conversion efficiency from producers to primary consumers.

In our model, the energy will be represented by a characteristic names “stomach”. For each action realized (deplacement, hunting, reproduction) the individual will lose a bit of it. Each time he success hunting, the eating process will give him part of the amount of energy of the prey (symbolized by his Health Points). The population pyramid will have to be taken in account. The higher we are looking in the chain, the smaller the population will have to be.

The number of organisms in a population changes over time because of the following: births, deaths,immigration, and emigration. Of course, births and immigration increase the size of the population; whereas, deaths and emigration decrease the size. The increase in the number of organisms in a population is referred to as population growth. There are factors that can help populations grow and others than can slow down and even prevent populations from growing.

In our model, immigration and emigration will not be designed. The map will represent a small closed ecosystem. One lair will be assigned to each animal species (where reproduction will occur)

The human acting as a god on a food chain reveals the interest of implementing this in persistent universes. As a change in the size of one population in a food chain will affect other populations, the player will have a direct influence on the environment for each action doing in relationship with the ecosystem (hunting, collecting animal or vegetable resources and even pollution).

Chapter 3

Model

To get a better understanding of the underlying mechanisms of a food chain applied to games, I started the development of a prototype. Some issues we need to address clearly states when trying to realize such a prototype. The objective was to realize a simple food chain (including one herbivore species and one carnivore). Life is very complex, and before trying to simulate life mechanisms, it is important to determinate, according with the craved purpose, which part of the system we would like to reproduce, which pattern we would like to obtain. Based on this, certain part of the life (either in macro or micro scale) will be selected while others will be represented thanks to just one variable.

3.1 Basic Rules

This part shall present the basics rules of our ecosystem. We can divide the behavior of our subjects with two main objectives. First of all, animals need to eat and drink to survive. Then, when the minimal age for reproduction is reached (and the subject is not starving nor thirsty), it will start to look for a mate in a reproduction purpose. Keeping in mind the final purpose of such a prototype, we decided not to take so much care about the plant reproduction. We consider they spread as fast as they are eaten in order to focus better on animals population evolutions.

3.1.1 Hunting process

First of all the animal need to locate a target. If the later correspond with the type of food our subject is looking for, the hunting process start. The hunter will start to run toward his target. The moment the prey will noticed to be chased and starts to run away is based on the hunter's discretion and the prey's detection attributes. If the hunter reach the prey, he will try to hit it. If he succeed, the hunt is over : the hunter eat the prey, and start looking for another one. If his stomach is full enough to afford a copulation energy cost, he will reach home, looking for a mate.

3.1.2 Reproduction

To simplify reproduction state, each species has a associated lair. When an individual desire to reproduce, he will reach this lair, and looking for a individual of the opposite sex in the same situation. When a couple is created, the genes are mixed up (in a way for the child to inherit of the best characteristics the parents used to survive) to create the genotype of the child(ren). The number of child is related with the hierarchical position of the specie into the food chain. The higher it is, the less children a copulation will produce. When the reproduction act is done, the male goes back hunting meanwhile the female starts the maternity period (whose last for a period based on the species)

3.1.3 Mathematical model restrictions

Due to the toughness of writing down with mathematical and computerize structure life process, some simplifications must be done. We also have to keep in mind that our final purpose is not to reproduce perfectly life mechanisms, but to implement the relevant ones in a multiplayer game, thus enhancing gameplay. A tricky part was for animals to find a suitable target. The *lookForTarget* function took me some times to be designed. Computer resources limitation and parallelization purpose makes it even trickier. We address this issue later in this paper, but what we need to understand here was the need of centralize each species. The “lair” concept was introduced, giving to each species a safe place for reproduction. Thus, how to force species to meet themselves in order to force hunting ? If a random function is used for the moves during the looking-for-target part, few meetings occurs. It would have been possible that each animal knew exactly where their target lair is. But then another problem occurs. Some path were automatically created, having for result a bad repartitioning among the map (which could lead to a tough job to synchronize processors during parallelization).

To remedy to this issue, the “river” concept was introduced. One or more water points where introduced as the “thirsty” gauge for each animal. Animals were then forced to meet each others. These two concepts gave us some good result even if an active learning feature would be much more interesting (discussion around it in the “futur work” part of this paper).

3.2 Behavior state evolution

> "*A behavior is a regularity in the interaction dynamics between an agent and its environment, for example, maintaining a bounded distance from the wall, or having a continuous location change in a particular direction. One or more behavior contribute to the realization of a particular functionality.*"
>
> *Luc Steels* [15]

As we saw during the prestudy, many implementation of a virtual food chain has already been realized. But none with the purpose of a persistent universe integration has been found. Therefore, most part of them does not take in account the relative position of each individual toward the others. How does a hunter find his prey ? How and when the prey react ? To sum up, the first problem encounter by the author was : how designing the hunting process ?

Based on role playing games attributes/fighting systems, on video games limitations the following prototype was designed. It has also being inspired by a mail game simulation of evolution [16] where a player is responsible of one species in a jungle interacting with others. Even if this simulation simplify the hunting process (e.g does not have the concept of the Map so important in video games) and focus on genetic evolution, the later was a good source of idea for this linear/non-linear paradigm mentioned above.

The prototype realized was order following the typical turn-based combat system. During each subdivision of time, the program will go through each individual of each species. Based on the status (which symbolize his primary objective for the turn) the individual will act. This is the list of the possible status the individual can take with a brief explanation of each one :

- 1 : hunting. The individual is looking for a target.
- 2 : escaping. The animal is chased by a predator and try to escape from him.
- 3 : chasing. It has a target in sight and try to caught it.
- 4 : heading home for reproduction. The stomac of the individual is full enough to spend some energy for reproduction
- 5 : looking for mates. The individual is at home, looking for a mate of the opposite sex ready for reproduction
- 6 : looking for water. Thirsty, the animal has to drink.

According to his actual status, some actions will be chosen by the element, maybe leading to a change of status. The following diagram expose all the possible relationship between the different status.

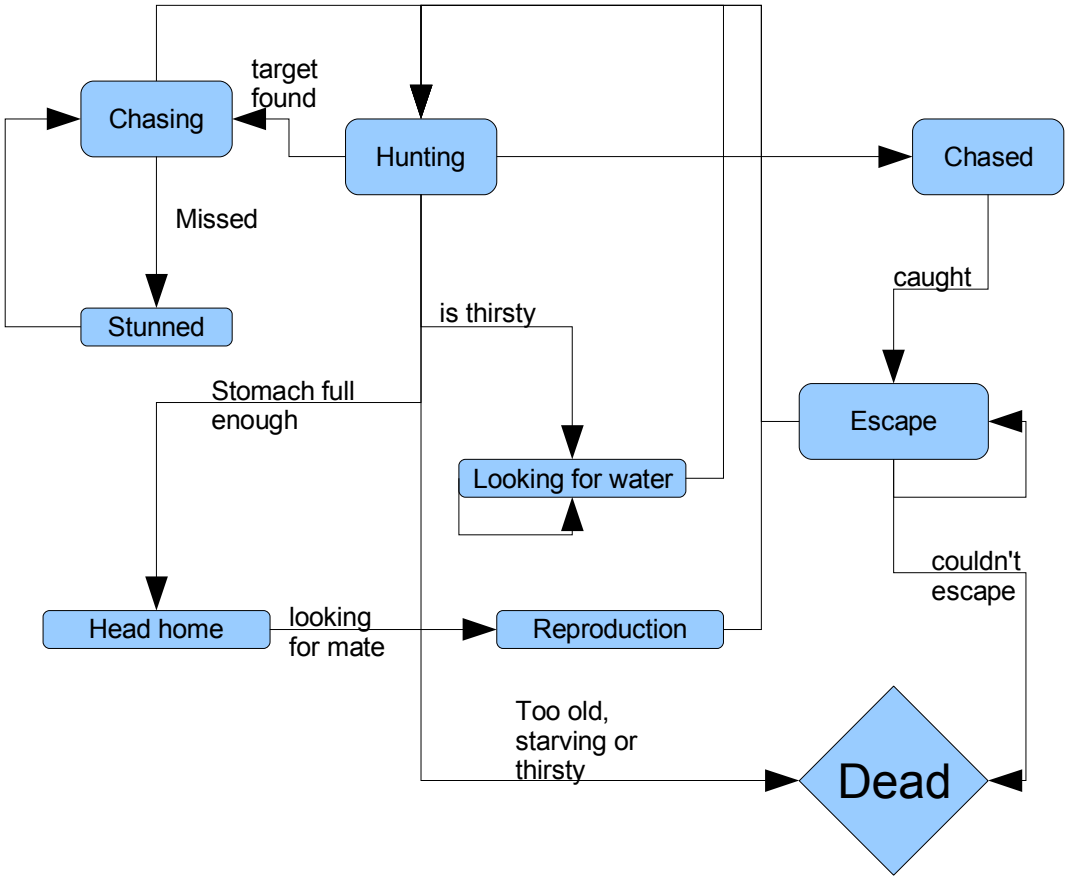

Figure 5 : Behavior state evolution

3.3 Non-Linear Phenotype/Genotype mechanism

The phenotype results from the expression of the genotype. The genotype is represented through a pair of chromosomes (two int-arrays), one from the father, one from the mother. This model (including sex differentiation) is closer to the reality than genetic algorithms studied above (generally using a unique gene). For each gene, a random function will chose during the reproduction process if the one of the father or the mother will be expressed in the phenotype. In addition, there will be mutations allowing the species to evolve. The mutation will act randomly on one characteristic.

In real life, a particular skin color or size of the paws can have many influences (some positives, some negatives). As I wished to keep in mind this relationship, this particular model was built. Attributes are calculate based on the value of the genes. There is two levels of attributes. The first one is directly based on the genes values whereas the second one is based on the first one. Let us first look at the table 1 which represent the chromosome structure (represented as a table of Integer in the code).

Nose
Ears
Eyes
Mouth
Paws
Horns
Tail
Legs
Arms
Brain
Sex
Size

Table 1 : representation of a chromosome

Then, primary attributes needs to be deducted from this gene. In order to keep the non linearity of genotype/phenotype relationship, one gene cannot be responsible of a whole attribute and need to influence the others. Another interesting point to notice is the table 2 can easily be accessed in the code for further modifications.

	Perception	*Agility*	*Power*
nose	++	.	.
ears	++	+	-
eyes	++	-	+
mouth	-	+	++
paws	+	-	++
horns	.	.	++
tail	+	++	-
legs	-	++	+
arms	.	++	.

Table 2 : Attribute map example

In the chromosome, one gene (which is maybe the most important value) is the *Size*. Two others genes will give information about the size of the target. If the target found has a size included between *minSize* and *maxSize*, the hunter will start chasing the target, otherwise he will find it too small or too dangerous. This system was the best one found in order to create a food chain. Originally the animal was hard coded with the typical species he had to hunt. But this architecture restrict severely the environment self-evolution. If the mouse after a particular mutation becomes twice faster, all the cats are going to die. In real life they would start to hunt more birds… This "size" parameter solve this problem.

But more than that, the size specify how much points will be given to each physical characteristic in the above table.

For example, an individual of size 16 has the following genotype :

nose : 2
ears : 2
eyes : 1
mouth : 3
paws : 2
horns : 0
tail : 3
legs : 2
arms : 1

The total is of course 16 (the size). The '++' symbol means a factor 2, '+' means a factor 1 whereas '-' means a negative factor of 1.

Then, an ear of size 2 will have the following impact :

+4 perception
+2 agility
-2 power

When the total is computed with all the physical traits, each primary attribute must be positive (or equal to 0). If it's not the case, the life form is declared inapt to the environment and died right after being born.

The primary attributes are only used directly in one circumstance : perception will be opposed to Agility (balanced with size) in order to determinate if the prey detect the hunter approaching. Apart from that, there is no other situations where the primary attributes will be used, but they are necessary to compute the secondary attribute values.

- Hit & Escape : When the hunter succeed to reach the prey, the Escape value of the target will be opposed to the Hit value of the hunter.
- Endurance : associated with the size, it will determinate the speed of the animal.

The figure 6 presents the relationships between first and second level attributes :

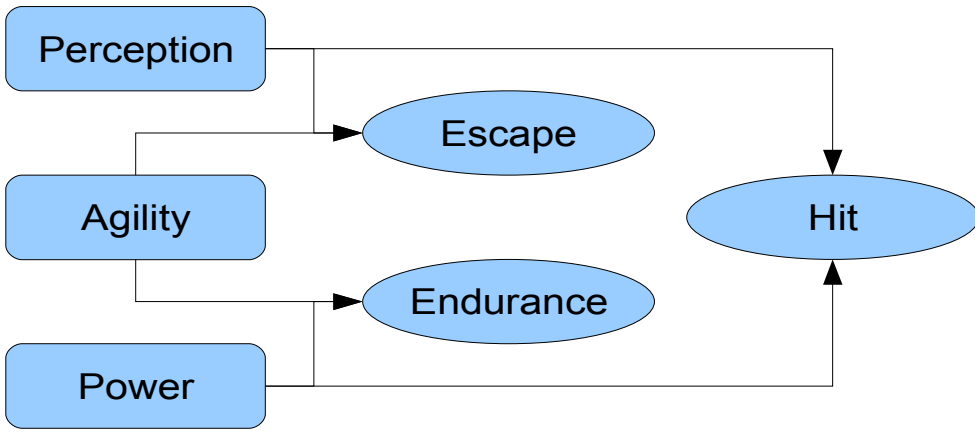

Figure 6 : First and second level attributes relationships

3.4 Prototype's architecture

3.4.1 Genotype Class

The lowest level class. An instance of this class will mainly contain the chromosome. Two different constructors exists. One which create randomly a gene given the size (for the initialization of the pool when starting the simulation), the second one will create a genotype given the chromosomes of the parents. The rest of the methods is for accessing the gene.

3.4.2 Phenotype Class

The phenotype contains mainly both attributes levels, but also a reference to the genotype. It also contains all the functions to compute the attributes given a genotype. A great amount of get methods was required to access each attribute.

3.4.3 Individual Class

The heart of the program. All the individual's characteristic such as the age, the stomach or the thirst are stored here. A reference toward the phenotype (coupled with the required methods) is used for accessing the attributes. The destination and status values are used to save the purpose each individual is following. Here is a short list of the main methods of this class :

lookForTarget *:* when an animal is looking for a target, and randomly destination is chosen for him. He will then run toward this point until he reach it (in that case another destination will be set up), or until he find a target. At the beginning of each iteration, before moving in the direction of his destination the individual will look around him to see if there is a target which with a size fitting the requirements. To do that a special method called **check** will receive a cell's reference and return a value indicating if the cell contains a suitable target. First of all, the **check** method is called with the cell containing our individual. If no target is found, the surrounding cells whose border are shorter than the range of the animal will be checked. If a target has been found in the primary cell, it will be saved in the target value, and only the cells whose borders are closer than the target will be checked. Then, the main function will use the appropriate method to change the status and

the destination of the individual.

lookForMate : This method is called when the animal is ready to reproduce (at home, stomach full enough and not thirsty). The individual will then look for the others with the same status than him but an opposite sex. If one is found, the reproduction method of the species class will be called.

chase, go, goHome, goToRiver : Considered as deplacement methods, they are used for moving the animal toward the destination they need to reach.

fight : When the hunter reach his prey, the fight start. The Hit value of the hunter will be opposed to the Escape one of the opponent. If the hunter wins, the prey will be eaten (filling the hunter's stomach). If the hunter miss his prey, he will be stunned during few iterations allowing the prey to escape.

escape : this function is designed to compute the opposite direction of the hunter and to make the prey running in this direction.

isThirsty, isHungry, drink, fillStomach, digest *:* The two formers are testing methods returning a boolean value indicating if the animal needs to drink or eat. drink will be called when the animal reach the river. **fillStomach** will be called after a successful hunt. The value the animal will eat is the size of the prey minus a substantial value as explained in the prestudy part.

init and stop methods : Some methods can be found with a name as **escapeInit** or **escapeStop**. They are used to properly set the destination, hunter, target and status values.

compare : When attributes of two different animals are faced of, this particular method is called. It works as follow :

coef = preyValue/(preyValue + hunterValue)

Then a random number between 0 and 1 is picked up. If it is higher than *coef*, the prey won. This is used when fighting, to determinate if the prey detected the hunter chasing it, ...

3.4.4 Species Class

The species contains mainly a list of reference to all the animals of the so-called species. Some characteristics common for all individual of the species are saved here (the lair location, the maternity length, the average number of birth for each reproduction act, and the attribute map (used for calculating attributes values based on genotype).

It is also responsible of keeping up-to-date some statistics as the size of the population, the average individual's size, ... Some output functions were also introduced for the creation of a new species or a new individual. Two different constructors exists, one for animals the other one for plants. Two create methods are also part of this class.

The two others main methods of this class are the one responsible of death and the one responsible of reproduction. death receive in argument the references of an individual. It is responsible of removing it from the list, and liberate the memory space. The reproduction methods receive the male and females references, and create as many new baby as the average number of birth of the species (ponderate with a random function). The maternity length is applied to the babies and the mother while the father remains free of his actions.

3.4.5 Cell Class

This class is not really linked with the others. If we see the map as a grid, a cell is the smallest part of the grid (its size can be determinate at the start of the program). It basically contain a list of Individual's references which are in the cell and the methods used for filling or modifying it.

3.4.6 Main function pseudo-code

```
initialization

for each time step

      for each species
            for each individual
                  based on location, fill the right cell

      print the map

      for each species
```

```
{
      for each individual
      {
            based on the status
            {
                  hunting :
                        check if the individual is thirsty or hungry
                        otherwise look for target
                        if target found -> set chasing mode

                  escaping :
                        if escaping is a success : resetStatus

                  chasing :
                        if reach the target -> start fight
                              if fight won -> eat, and notify the prey as
                              dead
                              if fight lost -> stunned

                  goHome :
                        if home reached start looking for mate
                        if not go home

                  lookingForMate :
                        if male :
                              if female found
                                    reproduction
                                    male -> resetStatus
                                    female -> startMaternity

                        else : waiting for male

                  lookingForRiver :
                        goToRiver
                        if River reached
                              drink
                              resetStatus
            }
      }
}

for each species
for each individual

      if individual is marked as dead -> put the reference in a
      buffer
      if individual is hungry
            if survival test failed -> isDead
      if individual is thirsty
            if survival test failed -> isDead
      if individual is old
            if survival test failed -> isDead

delete buffer

next time step...
```

3.5 Results

The main goal was to obtain the Lotka-Volterra equations (also known as the predator-prey equations) which are used to describe the dynamics of biological systems in which two species interact. They are :

$$\frac{dx}{dt} = x(\alpha - \beta y)$$

$$\frac{dy}{dt} = -y(\gamma - \delta x)$$

Where y is the number of predator, x the number of prey and t the time. α,ß,γ and δ are parameters representing the interaction of the two species. The prey are considered to have unlimited food supply, and to reproduce exponentially (α) unless subject to predation (ßy). On the other side, the growth of the predator population is represented with δxy (so it depends of the prey population) while -yγ represent the natural death of the predators.

The equations have periodic solutions. An approximate linearised solution yields a simple harmonic motion with the population of predators following that of prey by 90°

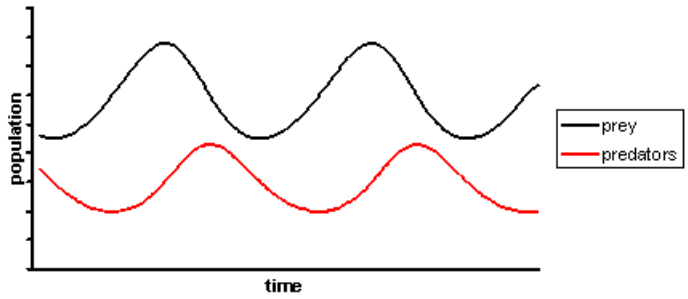

Figure 7 : Lotka-Volterra equations graphical solution

In the model system, the predator population rise when there are plentiful prey but, ultimately, outstrip their food supply and decline. As the predator population is low the prey population will increase again. These dynamics continue in a cycle of growth and decline.

The purpose was to obtain this result without computing the equations. Removing the genetic evolution feature (fixed genotype with average attributes, no mutations, no crossover while reproducing), after few hundreds iteration, we could indeed observe this phenomena. But the genetic evolution is naturally part of a virtual ecosystem.

It was much more tricky to obtain the same result with genetic evolution. Theoretically it is normal. Indeed, in a real ecosystem not only two species are interacting, which means if the prey evolve and succeed to escape the predators, then the predator will disapear and a different species which is able to hunt the evolved prey will take his place in the ecological niche. In our model interaction was not programed during the run of the prototype. Therefore it was not possible to introduce a new species. Thus, it always ended with the extinction of one (or both) species. In theorie, players and game master are supposed to both work on this issue introducing new species, or hunting individual which could threaten the stability of the system.

When launching the prototype with more than two species, the results were randomly acceptable. The ecosystem remains stable during a couple of hundred of iterations before mutation began to show its effect and unbalance the ecosystem.

The main conclusion I draw about these experiment is related to the attribute system presented above. It shows some good results as, according to the principle of real life, if one individual start to gain a specialization in one particular attribute he will never survive. If the *hit* value rise at the expense of perception, he will be not able to find any prey and will not survive. But some issues raised showing the limit of the representation. Herbivores has a strong tendency to develop *perception* and *agility* as *power* is not necessary to eat plants even if it has some influence on the speed. This could be a normal situation if it was not also the case of the predators. Indeed, perception revealed to be much more important as without it, the prey cannot find any target. Therefore a balance has to be done in order to equilibrate more precisely the hunting system.

Chapter 4

Discussion around Parallelization in MMOG

The key insight into the natural method of behavior generation is gained by noting that nature is fundamentally parallel. This is reflected in the "architecture" of natural living organisms, which consist of many millions of parts, each one of which has its own behavioral repertoire. Living systems are highly distributed, and quite massively parallel. If our models must be close to life, they must also be highly distributed and quite massively parallel. Indeed, it is unlikely that any other approach will prove viable[9].

In order to better introduce a parallelized food chain in massively multiplayer games, a study of the actual architecture of such kind of games is required.

4.1 Massively multiplayer games servers

Through the literature[17][18][19], two main models were developed : Client/Server and Peer-to-Peer. In the later each peer maintains a copy of the game state (there is no central server). Therefore, each entity is in charge of performing I/O with its own player and of maintaining its own game state copy. Often, in games such kinds of Peer-to-Peer solutions are based on fully connected overlay networks. Thus, to ensure the consistency of the game state across different players, each event produced by a player is directly sent to all others without the intervention of a central server. An example of a famous Peer-to-Peer game is *MiMaze*[20].

The Client/Server architecture is the classic solution used in game commercial products. In this case, a single copy of the game state is maintained at the server-side. Each event generated by an client (player) is sent to the single GSS that processes the event; then, the GSS forwards the newly computed game state to all other I/O clients. During the game, the GSS controls the generated events' validity. In this scenario, clients have only to take inputs from the players and render the output state. Consistency is easy to maintain as well as cheating avoidance, because only a single GSS maintains the game state, and thus illegal manipulations of the game state are extremely difficult.

Let us compare the characteristics of the peer-to-peer and the client-server architectures below [18].

- **Scalability**: Since the main server has to collect all participants' data and server all participants of a game, the server may become saturated when the number of participants goes up. Some tricks using mainly parallelization, are widely used, but a P2P architecture is definitely more scalable.
- **Delay**: In a centralized architecture, data reach their destination through the server, which can increase the network delay up to two times in a distributed architecture, where data only crosses the network once to reach their destinations.
- **Robustness**: It is desirable that the failure of any participant has no effect on other participants in the centralized system. This means all participants are equivalent and independent, and has all necessary information to compute the state of game at any time. The server-client architecture is obviously less robust since the system may fail due to a single point failure (i.e. the server).
- **Consistency**: Since all participant received information from the server only in a centralized architecture, global consistency is guaranteed. Moreover, the server introduces a natural synchronization among players.
- **Cheat-proof**: The presence of a centralized server makes cheating difficult. In a distributed architecture, each entity makes its own decisions, and there may be no authority to identify potential cheaters. Consequently, the deployment of distributed architectures will require a specific distributed mechanism to deal with cheating of participants.

So far, client / server architecture has be most widely used. Only research projects are interested in fully distributed games (with *MiMaze* as the only one actually working). Based mainly on the cheat-proofing advantage of such a solution. It also offers the editor company a full control of the server. In these circumstances the company can ask for a monthly fee to access the server. More than that, distributed architectures cannot be applied to every kind of online games. How could we keep an environment persistent when it hinge upon connected state of the players ? *MiMaze* is basically a Pac-Man where the players have to kill each other in a labyrinth. Thus only the position and the actions of the player need to be sent. Synchronization of player actions was a tough job as stated by L.Gautier & C. Diot in their research paper[21]. If one player has to be responsible of one species or one part of the map, the information to send (and receive) overcomes the actual performances of our networks.

Role playing games have always been centralized (on a game master originally). In video games it become much easier for the developers to add or modify regularly contempt to the game (to balance players abilities, manage the virtual economy, introduce a new animal, ...). Therefore, the persistent notion of the universe requires a client / server architecture.

4.2 Shared memory system

Such a work ask for a great amount of computational speed [22]. We will investigate here how our solution can be parallelized, what kind of issues we may encounter with an integration to actual server's architectures, and propose some designed to bypass those obstacles.

NPC behaviors and genetic evolution impose a high level of resources, especially when we are talking about massive numbers. Not to give so many influence to just one player on the full ecosystem, animals should be much more numerous than players. To add these features to an existing game server running we can obviously think of a supercomputer linked to the server taking care of the reproduction and hunt between animals. At the end of each iteration, the actual environment system should be accessible by the main server.

Shared memory systems are usually specially designed and manufactured, but can also be very cost-effective, especially small shared memory multiprocessors. In such kind of system, any memory location is accessible to any of the processors. A single address space exists : each memory address is unique.

Based on the number of processors, different architectures are used to link the processors and the memory in order to obtain a uniform memory access time. This gives us the opportunity to modify the charge of each processor dynamically [23].

The bottleneck of shared memory systems is located in the “shared”. Sharing data is a tricky process as reading variable by different processes does not cause conflicts, but writing new values may do so.

One of the fundamental issue which should be addressed as soon as possible is the domain repartition. In other terms : “How to share properly the individual on the processors ?”.

The first obvious solution is among the species. From a first look, according to the prototype's architecture, it appears to be quite simple. The “species” class elements are spread upon the processors.

The second option would be sharing the map among the processors, giving each one the responsibility of an area (a block of *Cell's* instances)

4.2.1 Sharing memory

If one animal our processor is taking care of meets another one, our processor will have access to the whole memory, so will be able to gather the required informations (as the size or the perception parameters). But if the animal start to be chased, the processor will need to modify the status. If at the same time, the processor responsible of the prey has just detected the element being thirsty (so needs to reach the river as soon as possible), he will try to modify the status value. Depending on which one will obtain the access first, the status can be corrupted leading to a non-reasonable behavior. Therefore, a memory access control algorithm is required to avoid such blocking. We need firstly to locate the critical sections. Some mechanisms exists for ensuring that only one process accesses a particular resource at one precise time. The process prevents all other processes from entering in their critical section. It is called mutual exclusion.

Whatever the domain repartition chosen, all the functions which modify any individual characteristic the thread is not responsible of are critical. Therefore the only operations which could be dangerous will be the list of the Individual class' methods which modify the class' private attributes. According to the species domain repartition principle, each species is concentrated on one processor. The whole class Species does not need any modifications as none of the other threads requires any write-access to is. On the case of “map repartitioning”, two individuals from the same species can be managed by different threads spread among the processors. Thus, all the public methods of the Species class that have an impact on the private members need to be classified as critical too. High performance programs should avoid as much as possible critical section, because their use can serialize the code[23]. The repartition among species obtain an advantage on this precise point.

4.2.2 Sharing work

It is obvious that if all the processors finish their attributed work for an iteration at the same time, there is less power lost that if one processor is late and the others need to wait for him. Even if the amount of work can vary greatly from one individual to another (one drinking while one reproducing for example), we can assume on the average if there is the same number of individual per processor, the execution time should remain almost the same. It is why, whatever domain repartition is used, we need to share approximately the same number of individual per processor.

For the spacial domain repartition, we need to divide the zone into as many area as there are processors. The best suitable way found so far is called Orthogonal recursive bisection method[24]. First a vertical line it found that divide the zone into two areas containing the same number of individuals. For each area, an horizontal line is found following the same rule. We stop when these is as many areas as processors. The following figure gives us an example of such a repartition (but with the goal of having only one individual per area) :

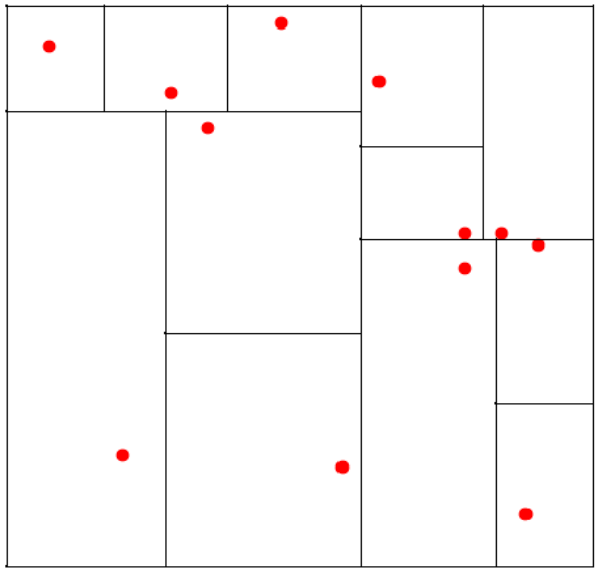

Figure 8 : Orthogonal Recursive Bisection Method [23]

If we wish divide the individual based on their species, we have to keep in mind all the individual of the same species must remain on the same processor. In addition, whereas with the Orthogonal Recursive Bisection method we stop the process when we reach the number of processor the system has, we have here no desire to base the species' number of our system on the number of processor available ! Few different species can be managed by the same processor. But as seen above each processor has to carry the same

number of individual.

It is an NP-Complete optimization problem. On of the simplest restatements of the problem is as a linear bin packing problem, where each processor is a bin, and species are represented as object to pack, whose length is the number of individual. Thus, the approximation algorithms (best fit decreasing) used with bin packing can be adapted here.

To get as much speed-up as possible, the major part of blockage should be avoided. They are mainly created due to waiting for an access to a critical section, or synchronization methods. As the only synchronization required between the processors is a barrier situated at the end of each time step, the only way to avoid latency here is to fairly spread the work among the processors.

The sharing data issue is more complex. Even if a spacial repartition will create more critical sections, in practice animals from the same species located at opposite position on the map are not requiring to be on the same processor. Thus, spacial repartition should produce less access to animals managed by others processors, reducing the busy waiting probabilities. But only empirical studies based on experimentation could bring us additional information.

Finally, industrial gaming production prefer to keep secret the structure of their server. Therefore, it becomes tough to extrapolate the place our shared memory system has to take, and how it has to be linked with the rest of the components' server. Even in the optic such a machine can be added to an existing persistent universe, the very expensive scalability it offers would gives us more an prototype with experimental purpose than a stable game.

4.3 Distributed Memory System

As discussed previously, cluster is the widest type of parallelization architecture used in MMO games. *Guild Wars* (Ncsoft, 2005) for example use it massively. The game is based on a whole instances system. When some players form a group for a particular mission, a special instance of the zone is created for them. They cannot interact with players situated in others area (apart from chatting). Each server is in fact a cluster where dynamical domain repartition is massively used to move players team from one server to another, creating and destroying area especially for them. From a computational point of view, it one of the best solution, but it has an impact on the gameplay. First of all each change of zone is responsible of a loading period, which is always criticize by players[5]. More than that, we loose the persistent component of the universe.

One particular space-opera game tried on the opposite to maintain one unique persistent universe where all the players can interact. *EvE online* (CCP games, 2003) is not very famous outside gaming communities, but its architecture remains one of the most advanced. Main and only *EVE Online* cluster (named *Tranquility)*[25] is composed from 2 CISCO Alteon routers, 4 IBM Blade proxy servers, 55 IBM x335 Sol servers and 2 clustered database IBM Brick x445 servers with Windows 2000 and MS SQL Server. The raw power of the cluster is 400 GHz with 200 GB RAM. Machines are interconnected with 1 Gbit/s or 2 Gbit/s lines. Even if *World of Warcraft* rules the market with more than 50% of the players[1], there is never more than few thousands players connected to the same server. Eve online regularly beat his own concurrency record with today around 40.000 players connected at the same time. This record was achieved mainly because of good scalability of the cluster (due to the extensive use of *Stackless python*[26]).

In a cluster architecture, the memory is not shared. Each processor has its own memory. Due to the time required, data transfer between nodes has to be avoided as much as possible. When the persistent universe is designed to be massive, cluster are widely used. Figure * (from *Minion of Mirth*[27], Prairie games, 2006) present a typical architecture of a cluster server for gaming purposes. The world deamon is the single point connexion which exist between one world and the master (and character) server. The world daemon introduce the player from the character server on the zone cluster responsible of the region where the player log out last time. Each region is divided upon a number of zone each one managed by a node of the cluster. The domain repartition chose by most part of the actual production is therefore a geographical repartitioning. Thus it becomes obvious this is the type of repartition we need to focus to implement food chain in it. Furthermore, as dynamical repartition cannot be done (due to the huge communication cost of clusters), sharing the ecosystems among species would become sooner or later unbalanced as species population evolve.

Exactly as in G*ame of Life*, the contempt of each cell at t+1 is based on the surrounding cells at t. If an animal near the border find a target on the neighbor-cell, he should start to run in this direction for hunting. Following the same principle than *Game of Life*, at the beginning of each time step, each cell sends information about its border to the neighbors. Therefore the computation can be done locally for each area, and then the new can be sent to the original cell. But this can be Obviously a synchronization algorithm will be required to coordinate the different actions realized.

In order need to reduce significantly the communication flow, we can minimize the number of objects located in the borders. A trick already used by *World of Warcraft* for example is to centralize as much as possible the important spots in the cells. In this game, all the towns (where the players spend a huge amount of time) are in the middle of each region. The only element that can cross borders are players. Our prototype cannot work this way, as animals are fully autonomous and travel from region to another one. But sill, centralizing rivers, lairs, and plants in the middle of each cell will greatly reduce communicational flow.

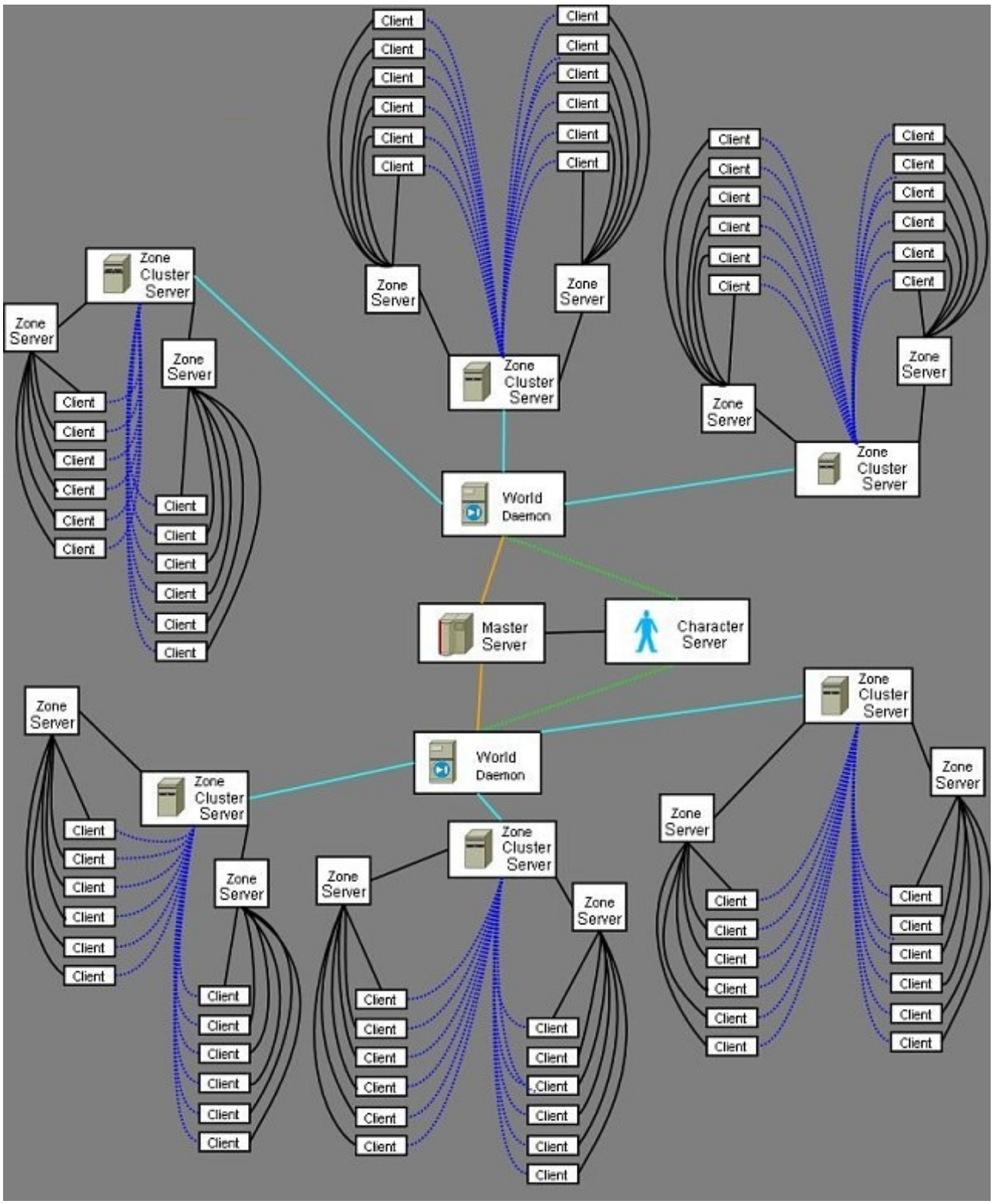

Figure 9 : Structure of *Minion of Mirth* Server[27]

Chapter 5

Conclusions and Future Work

In this thesis we investigated the idea of introducing artificial food chains into Massively Multiplayer On-line video Games (MMOGs). After emphasizing the lack of interaction with the environment in current games, and the desire for the players to reach a new dimension of gameplay, we studied how food chains could fill these gaps.

The issues raised by such an addition to the MMOG standards were discussed. We then presented a prototype architecture of a food chain for video games integration purposes. The fighting system based on genetically evolving attributes provided encouraging results, but also showed how system balancing issues that need to be addressed in future work.

The desire to use parallel programing structure associated with the high performance computation required by such a program, led the development to always keep parallelization in mind. This brought us about a discussion on the topic, offering theoretical organization structure for both shared memory systems and clusters.

Only practical experiments can now further the idea, the best being a team joining together experts from the different fields (ecology, high performance computing, video games design, artificial life & intelligence, network...)

5.1 Future Work

This work can be extended in many directions. The following subsections list some of the most obvious ones that we hope to look into in the near future.

5.1.1 Interactions

The main feature of video games is “interaction”. So far our prototype remains an algorithm with no possible interaction during the execution. With the ability to create new species, to kill existing animals in order to reduce their population or to introduce new ones with different genotype in order to extend the genetic pool of one species would strongly increase the interest of the result and give us a better picture of the idea's viability.

5.1.2 Active Learning

In order to get an ecosystem closer to reality, active learning need to be introduce. The behavior of the animals would look much more real. They would be able to learn about the place where food might be found, about the best way to hunt or about the predators they need to avoid. If a predator is drinking it could be a good option to wait he have finish before getting closer to the river. Therefore, a neural network can be attributed to each individual. As neural networks need to be trained, the baby can be programmed to follow his mother during the first iterations in order to learn.

5.1.3 Parallelization

I had the desire to work on parallelization since the beginning since I knew it would be required sooner or later for such an ecosystem. Nevertheless, I did not reach the point soon enough where I could share the work among many processors. But the development of the prototype was done all along with parallelization in mind. Therefore, after resolving the interaction issue, the next step will definitely be the parallelization of the algorithm on a cluster.

References

[1] MMOChart Webpage (Analysis of MMOG Subscription Growth) url:http://www.mmogchart.com

[2] Ubuntu distribution webpage url:http://www.ubuntu.com

[3] OpenMP webpage url:http://www.openMP.org

[4] Bartle, 2004, *Designing Virtual Worlds*, Indianapolis: New Riders.

[5] Yee, N. 2005. *Motivations of play in MMORPGs.* Paper presented at the Digital Games Research Association Conference (DIGRA), Vancouver, June 16-20

[6] Charles, D. 2003. *Enhancing Gameplay: Challenges for Artificial Intelligence in Digital Games*. in Level Up: Digital Games Research Conference. Utrecht

[7] Pottinger, D C, July 2003. *Computer-Player A.I.*, Game Developer Magazine, pp 24-29.

[8] Bedau, M. A., McCaskill, J. S., Packard, N. H., Rasmussen, S., Adami, C., Green, D. G., Ikegami, T., Kaneko, K., & Ray, T. S. (2000). *Open problems in artificial life*. Artificial Life, 6, 363–376.

[9] Langton, C. G. (Ed.), 1989. *Artificial Life*, ACM

[10] Martin Gardner, 1970, MATHEMATICAL GAMES The fantastic combinations of John Conway's new solitaire game "life" by Scientific American 223 (October 1970)

[11] Vitorino Ramos, *On the Implicit and on the Artificial - Morphogenesis and Emergent Aesthetics in Autonomous Collective Systems,* INSTITUT D'ART CONTEMPORAIN, J.L. Maubant et al. (Eds.)

[12] De Jong, K., 2001, "*Evolutionary Computation: A Unified Approach*", MIT Press

[13] Grefenstette, John J., 1986, "*Optimization of control parameters for genetic algorithms*". IEEE Transactions on Systems, Man, and Cybernetics, SMC-16(1), 122-128

[14] Saunders Co., 1959, "*Fundamentals of Ecology*", 2nd ed., Philadelphia.

[15] Steels, L. (1994). *"The Artificial Life Roots of Artificial Intelligence"*. Artficial Life, 1, 75-110.

[16] open-ended simulation game of evolving species. url:http://www.pbm.com/~lindahl/fchain.html

[17] Ferretti, S. 2005, "*Interactivity Maintenance for Event : Synchronization in Massive Multiplayer Online Games",* University of Bologna, PhD Thesis

[18] Chia-chun Hsu, Jim Ling, Qing Li and C.-C. Jay Kuo., 2003, "*On the Design of Multiplayer Online Video Game Systems"*, University of Southern California, Los Angeles, SPIE proceedings series

[19] Bartlett, R. 2004, "*A categorisation model for distributed virtual environments"*, Univ. of Western Sydney, Australia; From *Parallel and Distributed Proœssing Symposium*, 2004. Proceedings. 18th International on page(s): 231

[20] MiMaze webpage. url:http://www-sop.inria.fr/rodeo/MiMaze/

[21] Gautier L., Diot C., 1997. *MiMaze, a Multiuser Game on the Internet*, INRIA

[22] Stytz M. R., "*Distributed VirtualEnvironments*", IEEE Computer Graphics and Applications, Vol. 16, No. 3, 1996, pp 19-31.

[23] Wilkinson B., Allen M., 2005, *Parrallel Programming*, 2nd Ed. Pearson Education, Inc.

[24] R. D. Williams R. D., 1990. "*Performance of Dynamic Load Balancing Algorithms for Unstructured Mesh Calculations"*. Technical Report C3P913, Institute of Technology, Pasadena, Califronia.

[25] Hapala M., 2006, "*Programming techniques for the development of massive multiplayer on-line games"* Bachelor thesis Czech Technical University,Prague

[26] Stackless python webpage url:http://www.stackless.com/

[27] *Minon of Mirth* webpage url:http://www.prairiegames.com/index.php