ECOSYSTEM-SIM: A VIRTUAL ECOSYSTEM SIMULATOR

By

RODERICK F. SIMPSON

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE

UNIVERSITY OF FLORIDA

2001
Copyright 2001

By

Roderick Simpson
This work is dedicated to my family for all their love and support.
ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Doug Dankel, for his guidance and encouragement throughout this project. I would also like to thank Dr. Sharleen Simpson, my mother, for providing ideas, support, and exemplary editorial skills.
TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>CHAPTERS</th>
<th>page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>iv</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>viii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>ix</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>x</td>
</tr>
<tr>
<td>1 INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Simulation-Oriented Games</td>
<td>2</td>
</tr>
<tr>
<td>1.1.1 Creatures</td>
<td>2</td>
</tr>
<tr>
<td>1.1.2 Black and White</td>
<td>3</td>
</tr>
<tr>
<td>1.1.3 The Sims</td>
<td>3</td>
</tr>
<tr>
<td>1.2 Real Life vs. Artificial Life</td>
<td>4</td>
</tr>
<tr>
<td>1.2.1 Real Life</td>
<td>5</td>
</tr>
<tr>
<td>1.2.2 Artificial Life</td>
<td>6</td>
</tr>
<tr>
<td>1.3 Digital Simulations</td>
<td>8</td>
</tr>
<tr>
<td>1.4 Thesis Outline</td>
<td>10</td>
</tr>
<tr>
<td>2 RELATED RESEARCH</td>
<td>11</td>
</tr>
<tr>
<td>2.1 Introduction</td>
<td>11</td>
</tr>
<tr>
<td>2.2 Biology</td>
<td>11</td>
</tr>
<tr>
<td>2.2.1 Growth Cycles</td>
<td>12</td>
</tr>
<tr>
<td>2.2.2 Carrying Capacity</td>
<td>13</td>
</tr>
<tr>
<td>2.2.3 Genetic Inheritance/Heredity</td>
<td>14</td>
</tr>
<tr>
<td>2.2.4 Gender</td>
<td>15</td>
</tr>
<tr>
<td>2.2.5 Food Preference</td>
<td>15</td>
</tr>
<tr>
<td>2.2.6 Color</td>
<td>16</td>
</tr>
<tr>
<td>2.2.7 Speed, Size, Hunger, Thirst, and Procreation Tendencies</td>
<td>17</td>
</tr>
<tr>
<td>2.2.8 Genetic Drift</td>
<td>17</td>
</tr>
<tr>
<td>2.3 Artificial Intelligence</td>
<td>18</td>
</tr>
<tr>
<td>2.4 Computer Games Programming</td>
<td>20</td>
</tr>
<tr>
<td>2.5 OpenGL</td>
<td>21</td>
</tr>
<tr>
<td>2.6 Windows Programming- MFC Library and The Document/View Model</td>
<td>22</td>
</tr>
<tr>
<td>2.6.1 The Microsoft Foundation Class Library</td>
<td>23</td>
</tr>
<tr>
<td>2.6.2 The Document/View Model</td>
<td>23</td>
</tr>
<tr>
<td>2.7 Conclusion</td>
<td>24</td>
</tr>
</tbody>
</table>
6.3.6 Advanced Evolutionary Enhancements ........................................................... 57
6.3.7 Ultimate Goal, World ....................................................................................... 57

LIST OF REFERENCES .................................................................................................. 58

BIOGRAPHICAL SKETCH ............................................................................................ 60
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-1:  This combination will result in 25% probability of an omnivore, and 75% probability of an herbivore.</td>
<td>15</td>
</tr>
<tr>
<td>2-2:  This combination will result in 100% probability of an herbivore.</td>
<td>15</td>
</tr>
<tr>
<td>2-3:  This combination will also result in 100% probability of an herbivore.</td>
<td>16</td>
</tr>
<tr>
<td>2-4:  This combination will result in 100% probability of the color red.</td>
<td>17</td>
</tr>
<tr>
<td>2-5:  This combination will also result in 25% probability of blue, and 75% probability of red.</td>
<td>17</td>
</tr>
<tr>
<td>3-1:  Keyboard commands to influence creatures.</td>
<td>33</td>
</tr>
<tr>
<td>3-2:  Keyboard commands to influence food and water objects.</td>
<td>34</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-1: Main Screen of EcoSystem-Sim.</td>
</tr>
</tbody>
</table>
As interactive computer simulations have evolved in recent years, a new trend in computer games has begun to emerge. In these new types of games, users participate by creating and caring for artificial life forms, in an artificial environment. These games include such commercial titles as Creatures and The Sims.

The goal of this thesis is to design a scaled down version of such a system. A user running the system creates customized creatures, which are placed into the ecosystem. During the course of their lifetimes in the ecosystem, these creatures respond to their needs for food, water, and procreation. EcoSystem-Sim supports herbivores, carnivores, and omnivores, all of which can be fully customized. The system also models several naturally occurring trends in nature, including out-of-phase exponential population expansion, and genetic drift.
CHAPTER 1
INTRODUCTION

As interactive computer simulations have evolved in recent years a new style of computer games has begun to emerge. In these new types of games, users participate in elaborate simulations of either real or fantasy life. Goals may range from creating and caring for artificial life forms, to playing the role of a god ruling over a population of villagers, to guiding a modern day family through life. These games include such commercial titles as Creatures, Black and White, and The Sims. Game creators are beginning to realize that the game playing public can be entertained by more than standard issue Shooter-type games. The fact that The Sims spent the end of 2000 and the better part of 2001 at the top of the sales charts is an indication that players are enjoying this break from the norm [Wal01].

One concept that is very different from most games is that there is no way to win these games; they are simply interactive. They simulate virtual worlds that exist and evolve, and the user interacts with life forms or objects in the world. How the user interacts with entities in these simulations influences the way that the world evolves. The worlds may grow and prosper, or they may fade and fall into decay. The creatures in these simulations have exceptional Artificial Intelligence (AI), and at times it almost seems as if they are alive. AI has a long history of trying to create artificial life on computers.

One of these programs is called PolyWorld, created by Larry Yeager from apple computer. PolyWorld is a simulation of life, in which creatures grow and evolve, and
learn. The program displays a continually evolving population of creatures. One of the most interesting aspects of PolyWorld is that it pays strict attention to the biological underpinnings of real ecosystems. Thus, the simulation supports some interesting traits such as evolution, and natural selection. However, this program was not designed as entertainment software, and is not interactive.

Enter EcoSystem-Sim. This software package is designed to bridge the gap between non-interactive ecosystem simulations such as PolyWorld, and the new paradigm of interactive simulation games.

1.1 Simulation-Oriented Games

The following sections will describe the games mentioned earlier in greater detail. These games include Creatures, Black and White, and The Sims.

1.1.1 Creatures

In the game Creatures, the player becomes the caretaker of a small race of new life forms called Norns. Norns look like small furry little dog-teddy-bear creatures. The player has to train these creatures, teach them to use machines, care for them when they are sick, and help them defend themselves against the evil race of Grendals [Cre01]. From birth through death, the player must teach them language, how to identify objects, and eventually, breed them. The first few generations of Norns not very smart, but with each generation their brains get bigger, and therefore they become noticeably smarter.

As the game progresses, and the player teaches the Norns names for things, such as “carrot,” they begin to teach each other these names. This can be great if they are teaching each other the correct things, however they often learn incorrectly, and thus teach all their little Norn friends the wrong things. To try to correct behaviors, the player
can punish or reward the Norn for its actions. If it is punished, it will think twice before repeating the action it was punished for. If it is rewarded, it is likely that it will repeat the action. This type of learning is an ongoing process as each new generation of Norns is born. As stated earlier, there is no way to actually win this game, however, the interaction with these creatures makes the process very enjoyable.

1.1.2 Black and White

Black and White is a game that is unique, in that the player assumes the role of a God for a village. The player starts the game by trying to convert the villagers to loyal followers. Once the villagers worship the player as their god, the player summons them to a Citadel. At this Citadel, the villagers dance around praising the player’s name. This dancing generates the life force, which is the essential ingredient for magic spells that the player can cast. These magic spells can be used to perform various miracles, such as summon rain for the villager’s crops. So that the player can interact more intimately with the villagers, the player is given a physical presence in the game, in the form of a creature.

This creature can take one of several forms, such as a lion or a bull, and is influenced by the player’s actions. With these tools, the magic spells, and the creature, the player is able to rule over the village and ward off enemy gods. As play progresses, the player gains control of more villagers, and can do battle with other gods via the creatures. Like Creatures, there is no real way to win this game; there are only different states of game-play.

1.1.3 The Sims

The Sims is another game that can only be played, and never won. Spending the end of the year 2000 and most of 2001 at the top of the sales charts, The Sims has been a
well-received game. The idea was to take the micromanagement of a whole city to the smallest possible level, the individual. The setting is a fictitious suburban neighborhood containing about 12 lots. Players create families to populate the houses, and then lead them through daily life. This includes the normal tasks of everyday life: using the toilet, sleeping, going to work, and cooking meals. Even the most mundane tasks are interesting and sometimes even exciting.

The player has several different career paths to choose for their Sim, some of which are police officer, athlete, doctor, or politician. To advance at work, a Sim must maintain good relationships with the other Sim-Neighbors in their free time. If the user does not help his/her Sim-person well enough the Sim could end up dying. However, there is a setting in the game, which will allow the Sim to act autonomously to a certain degree. Since a Sim’s motivations are driven by the various parameters of their personality, the Sim will take action if necessary. In other words, if a Sim is hungry, and the player is not feeding it, it will eat. However, Sims might miss work if not encouraged to go. This level of involvement and interaction with the people in the game creates a very fun game, taking game play and design to a new level.

1.2 Real Life vs. Artificial Life

Questions and debate surround the question of life. What is life anyway? Can artificial life – life created by humans – actually cross the threshold into real life? What does biology have to say about all this? The following section is a brief discussion of these issues, as put forth by various sources.
1.2.1 Real Life

Biology is the science of life, yet there is no comprehensive and generally accepted definition of what constitutes being alive. The problem is that any one quality that is common to all living things may very well show up in some non-living things as well. For example, one trait common to all living things is growth. Yet, there are non-living things that grow such as fire and crystals [Kro00]. The revelation is that living things cannot be described in terms of a single characteristic. Living organisms are complex entities that are not simply the sum of their parts. To become life, there must be an incredibly complex, ordered interaction between all facets of an organism. Because of these emergent properties, biologists have been presented with a difficult problem: life is a fundamentally intangible quality that defies simple definition [Aud02].

In practice, however, biologists describe that which is life by using a complex set of qualities that all life must possess. According to the online molecular biology notebook [Hor01], living organisms all share some similar traits: Growing, Moving, Eating (ingesting matter, metabolizing it, then its excretion), and Breeding. In his *Fundamentals of Physical Geography* [Pid01], Michael Pidwirny adds that Organisms tend to be complex and highly organized. Organisms also tend to regulate their bodies and other internal structures to certain normal parameters; they are homeostatic. Pidwirny also states that life undergoes mutation, and natural selection. In other words, it adapts and evolves in step with external changes in the environment.

In the college biology textbook *Biology: A Guide to the Natural World* by David Krogh, the following is a list of those qualities [Kro00].

All living things:

1. Can assimilate energy and use it.
2. Can respond to their environment.
3. Can maintain a relatively constant internal environment (homeostasis).
4. Possess an inherited information base, encoded in DNA that allows them to function.
5. Are composed of one or more cells.
6. Evolved from other living things.
7. Highly organized compared to inanimate objects.

What is interesting to note is that this definition of life focuses only on carbon-chain chemistry. For example, in her new textbook, Biology: Life on Earth 6th edition, Teresa Audesirk writes “the molecules of life are called organic molecules, meaning that they contain a framework of carbon”[Aud02]. Tom Ray of the University of Delaware wrote an essay titled Evolution, Ecology and Optimization of Digital Organisms [Ray01a], where he notes that biology is not restricted to the study of carbon-based life; it is simply that it has been the only kind of life that has been available to study. Thus, theoretical biology has long faced the fundamental obstacle that it is impossible to derive general principles from single examples. This conundrum is only exacerbated by the fact that researchers are beginning to explore the viability of artificial and non-carbon based life.

1.2.2 Artificial Life

Humans have long been desirous of the creation of artificial life. From Mary Shelly’s Frankenstein to Commander Data of the television show Star Trek – The Next Generation, we humans have expressed our desire to create life by means other than procreation or reproduction. With the emergence of very fast computers and advanced
robotics, these ideas may not be too far out of reach. In fact, a new field of study has
been created to explore these dreams in an academic capacity. “Artificial Life (‘AL’ or
‘Alife’) is the name given to a new discipline that studies ‘natural’ life by attempting to
recreate biological phenomena from scratch within computers and other ‘artificial’
media” [Alf01 pg.7].

DNA and the decoding of the human Genome have begun to open new doors in
our quest to discover what life is. DNA is a nucleic acid, into which is encoded the
instructions for assembling a wide assortment of proteins from a limited number of
smaller building blocks [Sta00]. An interesting by-product is that we are beginning to
realize that we are a lot closer to the family of machines than we once thought. Every
one of us is really just an information processor, transcribing the organic code of DNA.
Arguably, our bodies are just “a gene’s way of making other genes [Joh00]”.

As humans strive to understand the nature of what it means to be alive, a natural
extension of that process is an attempt to actually create life. Researchers have tried to
create life in different forms, such as computer programs and robots. However, there are
biological roadblocks such as the requirement that all life be constructed of DNA. How
do computer scientists and robotics engineers overcome this barrier? Is it a viable
obstacle, or can life exist that is not constructed of DNA?

From the perspective of artificial life, it is difficult to answer the question of what
it means to be alive. The line between alive and inanimate is not a line so much as a
vague gray area. At the extreme ends, there are examples with which everyone can
agree. A person is alive; a rock is not. In the middle of these extremes, the distinction is
not as clear. Viruses are considered by many Biologists to be alive, but others disagree
with that notion. So, how and where can A-life researchers find their place in this field? Perhaps a different view of life and what is alive is needed.

In his book, *Artificial Life* [Lev92], Steven Levy argues for a hierarchy of living things. He states that some things are more alive or have more life than others. At the top of this hierarchy would be creatures such as humans and dogs, things that are the most alive. In the next tier might be things that are clearly alive, but less mobile than animals, such as trees and vegetation. The following tier would be composed of things like bacteria. In the next lower tier would be viruses, whose claim to life is in question. In the lowest tier would be things like rocks, automobiles, fire, and the economy. Following this description, Levy notes that organisms from artificial ecosystems and Commander Data “would fall somewhere between Chevrolets and the flu” [Lev92: pp. 6-7].

**1.3 Digital Simulations**

Creating artificial life on a computer is an exciting prospect. Can genuine life actually coexist with word processors and compilers inside of that little tan box? Have researchers come close to achieving this feat? What does it mean to be alive, and what is the definition of life? What are the moral implications and responsibilities of creating such life?

Digital simulations of artificial life were brought into the mainstream of A-Life researchers in 1994. Tom Ray introduced a program called Tierra at a Massachusetts Institute of Technology (MIT) conference on Artificial Life [Hel98]. Built in 1990, the Tierra C source code was intended to create a virtual computer. The architecture of its operating system was designed in such a way that the basic machine code was evolvable.
By randomly flipping bits or swapping segments of code between algorithms, the code would be mutated, yet still remain functional enough of the time for natural selection to improve it [Ray01b]. Ray termed his program a *computational ecosystem*, and opened the doors for many more to follow [Hel98].

Larry Yeager of Apple computer created another such Digital Ecosystem, which he called PolyWorld. PolyWorld was very different from Tierra in that instead of the code itself being alive, PolyWorld modeled a closed ecosystem. Organisms as well as free growing food inhabited this world. There were also several impassible barriers that divided the world into various sections to simulate a varying landscape. These organisms would engage in various activities such as eating, fighting, mating, and moving around.

The idea of this simulation was to press the start button and watch it develop into a thriving ecosystem. Larry built a self-starting mechanism into PolyWorld to make sure that a self-sustaining ecosystem would emerge. This meant that if the population of the ecosystem reached certain critical parameters, the program would generate organisms to keep the system going. For example, if some of the population was to die off, and there were fewer than a specified number of organisms, new organisms would be generated. Conversely, a maximum number of creatures can be specified, to keep the program computationally tractable [Yea94].

The PolyWorld program was a strong point of reference for this thesis. The goal of this thesis was to take a digital simulation like PolyWorld, and gear it towards the entertainment arena.
1.4 Thesis Outline

EcoSystem-Sim was created to be an entertaining and educational game based on a closed ecosystem. Like the games mentioned earlier, the ultimate goal of this project is to create a very engaging software package based on a closed system. The software that was written for this thesis is the foundation for such an undertaking. The EcoSystem-Sim package consists of two parts. The first is a back-end ecosystem simulation engine, which functions as the kernel for complete entertainment package. Second is a front-end graphical user interface (GUI) to display the current state of the system, and to allow the user to interact with the system.

Entertainment software packages created today typically have very large budgets, many employees, and many months to finish. For a thesis, this was not an option. To this end, the focus of this thesis is on the core-processing unit of the package. A simple front end and several user interfaces were developed to visualize what is happening inside of the core-processing unit. This back-end is designed to maintain the state of the ecosystem, and all creatures and food in it. In addition, the processing units for all creatures and food objects are housed in the core unit. Given the proper budget and resources, this core unit would be packaged with a new, graphics intensive front-end.

The following chapter outlines some of the research and technologies that that support EcoSystem-Sim. Next in Chapter 3, the front-end user interface is discussed, followed by the back-end core-processing unit in Chapter 4. This is followed in Chapter 5 by some of the results that were obtained from various experiments, and finally a summary of this thesis is presented in Chapter 6.
CHAPTER 2
RELATED RESEARCH

2.1 Introduction

Creating the EcoSystem-Sim software package necessitated input from several fields of study and technologies, which are discussed in this chapter. Fields of study included Biology, Artificial intelligence, and Game Programming. The technologies necessary were Windows programming libraries based on MFC, the Document/View model, and OpenGL.

Because EcoSystem-Sim models an ecosystem, it was necessary to draw on various ideas from the field of Biology. Creating the creatures and supplying them with primitive intelligence and instincts meant looking to the field of Artificial intelligence. EcoSystem-Sim is designed to be an entertaining and educational package. Thus, it was also necessary to incorporate game programming techniques and design. Next, the package was created in C++ using the Microsoft Foundation Class Library and their Document/View model. Finally, the rendering of the data was done using the graphics API OpenGL.

2.2 Biology

EcoSystem-Sim makes is an attempt to generate a virtual model of a primitive ecosystem. Even a primitive ecosystem is a complicated, interconnected conglomeration of actions and reactions. Also, because one of the primary goals of EcoSystem-Sim is to
educate, it is important to present accurate information. In order to model software such that it functions and achieves a balance, it was necessary to look to the field of Biology. Several important principles from biology were used to help to create a working model. These included growth/consumption rates, carrying capacity, genetic inheritance, and genetic drift.

2.2.1 Growth Cycles

Populations may exhibit several types of growth cycles. Typically, these will be regular cycles that will consist of a period of rapid growth, followed by a period of rapid decline. According to Audersirk [Aud02], growth rates of populations typically follow an exponential curve, or “J-Curve” during the expanding phase of the population life cycle, and fall off just as quickly. The time between expansion and contraction varies from population to population.

For example, Audersirk [Aud02] notes that lemming populations follow a 4-year cycle. Every 4th year there occurs a period of maximum expansion. This causes conditions to become unfavorable, and the next year the population falls off to its lowest point. The next two years, the population increases only a small amount, until finally the 4th year again displays the maximum amount of growth.

Another example noted was that of algae. It undergoes a boom/bust cycle that occurs every summer. During the rest of the year algae simply survives. During this time, the nutrients that it feeds on are replaced. In the beginning of the summer conditions become favorable for growth and the population explodes. At some point the nutrients are depleted, and the population enters the bust phase of the cycle, and falls off just as quickly as it grew.
The final example noted is that of two populations that are directly dependent upon each other. These populations have an out-of-phase relationship to one another, moving in a sine/cosine cyclical relationship. The specific example that was put forth by [Aud02] was one that was conducted using Bean weevils, and Branconid wasps in a laboratory in a closed system. The wasps lay their eggs on the weevil larvae, which provide food for the newly hatched wasps. A large weevil population ensures a high survival rate of wasp offspring, increasing the predator population. This in turn causes the weevil population to plummet as the wasps prey on them for food. A much smaller weevil population means the wasps have fewer larvae upon which to lay eggs. Thus, the wasp population drops, and the weevil population can once again thrive.

EcoSystem-Sim attempts to provide the facility to recreate each of these examples, by simply adjusting some of the initial parameters. The results of these experiments are given in a later chapter.

2.2.2 Carrying Capacity

Carrying capacity is the term used to describe the maximum population size that an ecosystem can support indefinitely. Two distinct types of resources determine whether a population stabilizes at or below this point. The first type of resource is renewable, and replenished by natural processes. These include things such as light, water, and nutrients. The second resource, space, is non-renewable. This is especially true with closed systems such as EcoSystem-Sim. In systems such as the earth, organisms can relocate to other areas if demands on renewable resources are exceeded. Most of the time this means moving to a less suitable location where population size could be reduced because of less favorable conditions [Aud02].
In general, population size is the result of a balance between two factors: biotic potential, and environmental resistance. Biotic potential is the maximum rate at which a population could increase, assuming ideal conditions. These ideal conditions should allow for a maximum birth rate, and a minimum death rate. Environmental resistance is the growth limits set by living and non-living things. These limits are enforced by the availability of food and space, competition with other organisms, and certain interaction among species such as predation [Aud02].

In EcoSystem-Sim, these limits are also relevant. For example, the availability of food is relevant to both herbivores and carnivores. There are only a limited number of plants for the herbivores, and a limited number of other creatures for the carnivores. Like a real ecosystem, creatures in the EcoSystem-Sim world are also in constant competition with other organisms. This includes like creatures competing for the same food sources, and herbivores confronting the issue of predation.

2.2.3 Genetic Inheritance/Heredity

EcoSystem-Sim uses basic fundamentals of genetic inheritance to determine the traits of the offspring of two Creatures. Genetic inheritance, or heredity, is the process whereby the offspring of a species takes on the traits of its parents. Individuals carry two genes for each trait, one from the mother and one from the father. One of these two genes will be dominant over the other. Variations of the gene relating to the same trait are called alleles. The dominant allele will mask the other, called the recessive allele. For example, if the father gives a tall allele of the height gene, and the mother gives a short allele, the offspring will be tall. This is because tall is dominant and short is recessive [Aud02]. EcoSystem-Sim has the following traits, which are passed from one generation to the next:
2.2.4 Gender

Like most animals, the Creatures in EcoSystem-Sim are either Male or female, and both a male and a female is required for procreation. The gender of the offspring is simply selected by random selection with 50% probability of a male, and 50% probability of a female.

2.2.5 Food Preference

There are three types of creatures in EcoSystem-Sim: herbivores, carnivores, and omnivores. These creatures could possess the following genes:

Herbivores: HH or HO.

Omnivores: OO.

Carnivores: CO CC.

The following Punnett Squares show the possible outcomes of the mating of two herbivores, and the resulting probabilities.

Table 2-1: This combination will result in 25% probability of an omnivore, and 75% probability of an herbivore.

<table>
<thead>
<tr>
<th></th>
<th>H</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>HH</td>
<td>HO</td>
</tr>
<tr>
<td>O</td>
<td>HO</td>
<td>OO</td>
</tr>
</tbody>
</table>

Table 2-2: This combination will result in 100% probability of an herbivore.

<table>
<thead>
<tr>
<th></th>
<th>H</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>HH</td>
<td>HH</td>
</tr>
<tr>
<td>H</td>
<td>HH</td>
<td>HH</td>
</tr>
</tbody>
</table>
Table 2-3: This combination will also result in 100% probability of an herbivore.

<table>
<thead>
<tr>
<th></th>
<th>H</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>HH</td>
<td>HO</td>
</tr>
<tr>
<td>H</td>
<td>HO</td>
<td>HO</td>
</tr>
</tbody>
</table>

The allele for herbivore is dominant, and thus it will mask the gene for omnivore, which is recessive. Because of the nature of the pairings, the mating of two herbivores can result in another herbivore or an omnivore, but not a carnivore. Likewise, the mating of two carnivores can produce either a carnivore or an omnivore, but not an herbivore. The mating of two omnivores will always produce an omnivore. If an omnivore mates with either a carnivore or an herbivore, any of the food types are possible. The probabilities will be based on what traits the parents have. Carnivores and herbivores will never mate together.

2.2.6 Color

A creature’s color is determined much the same way as food preference. Each creature has a color gene, which consists of two alleles: one for the father’s color, and one for the mother’s color. By default, red is dominant, and blue is recessive. Any new colors that a user of the program introduces into the system are dominant by default. Thus, the same type of selection process can be used to determine color. The following are the Punnett Squares for color:
Table 2-4: This combination will result in 100% probability of the color red.

<table>
<thead>
<tr>
<th>R</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>RR</td>
</tr>
<tr>
<td>B</td>
<td>RB</td>
</tr>
</tbody>
</table>

Table 2-5: This combination will also result in 25% probability of blue, and 75% probability of red.

<table>
<thead>
<tr>
<th>R</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>RR</td>
</tr>
<tr>
<td>B</td>
<td>RB</td>
</tr>
</tbody>
</table>

Additionally, if both parents are RR, then the creature has 100% probability of being red, and if both parents are BB, then the creature has 100% probability of being blue. EcoSystem-Sim determines the remaining traits in a different manner.

2.2.7 Speed, Size, Hunger, Thirst, and Procreation Tendencies

Determining the speed of a newborn creature is performed in the following manner. Either the mother or the father is chosen with equal probability. Then, that creature’s speed is used plus or minus 10%. In other words, if the mother was chosen, and had a speed of 7, the child would have a speed of 6, 7, or 8. The same type of system is used to determine the values for size, hunger, thirst, and procreation tendencies.

2.2.8 Genetic Drift

Genetic drift is the process whereby an entire population adopts one random genetic change. An example of this would be the spots on a cheetah, or the stripes on a
tiger. Genetic drift typically does not occur in large populations, but rather in very small ones.

An example cited by [Aud02] uses amoebas, which could be either red or blue. This trait is controlled by alternate alleles of a single gene. To start, two populations are assembled. The first consists of 10,000 members half blue and half red. The second is also half blue and half red, except it consists of only 4 members. The amoebas reproduce using binary fusion. One generation of the 10,000-member group will produce 5000 offspring, whereas the 4-member group will produce 2 offspring. The offspring have equal chances of being either red or blue. Statistical probability says that with a large sample like 10,000, the red and blue groups will be fairly evenly distributed. However, the two offspring from the smaller group could easily end up all blue or all red. It is comparable to flipping two coins. The chances of getting two heads or two tails are high [Aud02].

EcoSystem-Sim is also susceptible to genetic drift. Because the populations in the system are typically between 30 and 100, there is a high probability that all creatures in a system could take on certain characteristics. For example, like the experiment described above, in some simulations that were run, the entire population became the same color. Also, populations were wiped out because there were only creatures of the same sex left. These results are also described in a later chapter.

2.3 Artificial Intelligence

The creatures in EcoSystem-Sim are all autonomous entities. Each individual creature is responsible for deciding where it wants to go, what it wants to eat, and so
forth. The creatures respond to stimuli that they receive from the environment around them, and they also respond to their own internal needs. To this end, the field of Artificial Intelligence was the source some of the methodologies used in the software. These included decision-making heuristics and search techniques.

As each creature in the EcoSystem-Sim program goes about its daily routine, various search processes are utilized to help the creature choose what it wants to do. The creature may decide to eat, drink, run away, procreate, or just do nothing. These decisions are made by evaluating a compliment of state data against a set of heuristics. For example, each creature has a hunger threshold. When that threshold is reached, the creature decides that it must find food. However, if the creature’s pain threshold is reached, it will abort the search for food so that it may run away. This could happen if a creature was feeding from a plant, and another larger creature also decided to feed from the same plant. This heuristic process is also true for what food source the creature will pursue.

When a creature is hungry, it must decide which food source is best to go after. For omnivores, food sources include both other creatures and plants. For carnivores and herbivores, the food sources include only other creature or only plants respectively. To narrow the decision of which food source is the best, the creature must evaluate a set of heuristics against the results of various searches. These include distance away from food sources and accessibility of those food sources.

Consider the case for an omnivore. As the creature surveys the landscape around it for its prospective lunch, it has some decisions to make. Perhaps there is a plant nearby, but with a large creature feeding from it. If there is also a small creature roaming
around further away, it might be best to attack that small creature. Also, creatures that are faster (have a higher speed value) will be willing to go after creatures that are farther away, whereas a slower creature may not.

In order to gather this information, the program must search the data structures that hold information about all creatures and all plants in local proximity. It is important to consider the creature’s field of view, to make sure that the creatures cannot see things that are too far away. It is also necessary to limit the resolution of detail of things that are further away. In other words, the further away a creature gets from some food source, the less that it knows about it. Thus, the creature considers smaller and smaller subsets of information until it arrives at a decision. In this way, the creatures of EcoSystem-Sim use a combination of heuristics and search algorithms to make decisions and locate resources.

2.4 Computer Games Programming

Because the main focus of EcoSystem-Sim is entertainment, several game-programming techniques were utilized in its creation. These techniques include the use of collision detection, tracking, and animation.

Collision detection is one of the basic gaming technologies. Did Pac-man touch the power-up pellet? Because EcoSystem-Sim models a virtual world, there are physical interactions between the creatures. It is necessary to be able to determine if there are creature/plant collisions, or creature/creature collisions. For example, if a creature is trying to attack another creature, it is necessary to determine if and when the predatory creature catches its prey. This is accomplished by comparing the special distributions of both creatures. The same principles hold true for creatures trying to eat plants, and creatures in the pursuit of procreation.
Once a creature has selected a creature to pursue, it must be able to track and follow it. The tracking algorithm that is implemented in EcoSystem-Sim is based on continual evaluation of the subject being tracked [Par00].

Initially, the pursuing creature locates the prey, and using triangulation, computes a trajectory that will intercept the prey. As the prey continues to move, the predator will update and modify its trajectory. As the chase continues, both creatures will increase their speed to maximum. If the predator is fast enough to catch the prey, and the prey is weaker and smaller than the predator, the predator will inflict damage upon it. If the damage is not sufficient to kill the prey, the predator will continue to attack until the prey has been killed. Engaging in this combat also affects the energy and hunger levels of the predator.

Almost all games make use of some type of moving objects or characters in the game. EcoSystem-Sim displays all of the creatures as they move around in their virtual world. To make these actions visible and interpretable to the player, it is necessary to animate the creatures and food over time. The goal is to display a real-time view of the creatures in the ecosystem, and their interactions with one another. Animation occurs when the creatures on the screen are continuously erased and then redrawn. This page flipping is done using a double-buffered system, which is explained in the next section.

2.5 OpenGL

OpenGL is a 2D and 3D graphics application-programming interface (API). Since the API was introduced in 1992, it has grown to become the industry standard as a stable, multi-platform, easy to use, well documented, graphics package. Although OpenGL is an open-source initiative, additions and updates to the standard are tightly
controlled. OpenGL runs on every major operating system and every major windowing system [OGL 2001]. The major advantage to using OpenGL is the performance gains that can be realized.

The function of OpenGL in the EcoSystem-Sim package is to display the current state of the simulated environment. For this task, simple squares are used to represent the creature and food objects. This is due to the lightweight nature of the front-end. During the course of a simulation, it may be necessary to display as many as 4,000 creature objects. The use of OpenGL greatly increases the performance of the application when compared with the standard Windows GDI drawing commands.

To make the actions of the creatures interpretable to the player, it is necessary to animate the creatures and food over time. This animation is done by continuously erasing and redrawing the area of the screen that contains the creatures. A double-buffered system is used to improve performance and make the animations smoother. At any given time, one buffer is being displayed, while the other is being drawn to. At regular time intervals, the buffers are swapped, and the new buffer is posted to the screen [Woo99].

2.6 Windows Programming- MFC Library and The Document/View Model

Windows programming is a colossal empire. It consists of a vast landscape of code spread across thousands of libraries and packages. Much of this real estate has been consolidated to form the Microsoft Foundation Class Library (MFC Library). The beauty of the MFC Library is that it removes some of the drudgery from developing Windows applications by encapsulating many of the tedious, yet necessary parts of an application.
without sacrificing speed. Using the MFC Library also allows for the use of the program model called the document/view model [MSDN 2001].

The decision to use the Document View model is an architectural one, based on the features the application needed to support. There was a need to represent the data in both character form, and visual form. It was necessary to use an architecture that provided this functionality, and the document view architecture supports multiple views of the same document particularly well [MSDN 2001].

2.6.1 The Microsoft Foundation Class Library

The MFC Library consists of numerous classes that are thin wrappers for high-level Application Programming Interfaces. An object-oriented wrapper for Win32 automates many routine-programming tasks. Additionally, an application that is implemented using the MFC Library has a multitude of built-in features. These include toolbars, status bars, and File Save/Open functions in the drop-down menus of the application. For EcoSystem-Sim, this facilitated several features such as saving the state of a simulation, as well as adding some basic controls necessary to run the application.

2.6.2 The Document/View Model

The primary objective of the Document/View architecture is to separate data from their visual representation. The way the user actually views and manipulates that data becomes independent from the way the data are actually stored. In this paradigm, there are two main objects, the Document and the View. The job of the document object is to load and store the application’s data. The view object, which is just another type of window, allows the user to see the data on the screen and interact with it. Multiple view objects can display the data contained within a single document object. In this way, the underlying architecture of the application is decomposed into very logical pieces.
The EcoSystem-Sim software provides multiple views of data, which are continuously being modified. These views consist of the graphical display of the creatures and the readout of the numbers internal to the creatures. By splitting the components which need to access this data into distinct separate classes, the overall view of the system is much more logical.

2.7 Conclusion

EcoSystem-Sim was a collaboration of several very different fields of study, coupled with various technologies. It was a challenging and exciting prospect to bring together the fields of Biology, Artificial intelligence, and Game Programming. Although the software that resulted from this union does not create actual life, it is more interesting and diverse because of these sources. The next chapter will guide the reader through the operation of the EcoSystem-Sim software package, and the following chapter will discuss the internal workings of the software.
CHAPTER 3
ECOSYSTEM SIMULATOR - USER PERSPECTIVE

3.1 Introduction

The EcoSystem-Sim software application was designed to be a fun and interesting interactive experience. With that aim in mind, several features were added to help first time users get up and running quickly. For the more advanced user, more features linger in the wings to enhance the simulation experience. This chapter focuses on describing how to start the simulation, as well as how to create new creatures and add them to the simulation.

3.2 Start-up

Running EcoSystem-Sim can be as simple as launching the program or much more involved, depending on how much interaction the user desires. When the program is initially launched, the user is presented with a simple dialog inquiring if the user wants to run a custom simulation or one of the available, stock, preset configurations. If the user chooses a stock simulation, a second dialog box appears to allow the user to choose the one desired. If the user chooses a custom simulation, the simulation simply starts, with no creatures in it. The user then proceeds to use the “Create Creature” and “Create Food” menu options to populate the simulation. In both types of simulations, the user can always add more creatures or food to the simulation by using the “Create Creature” or “Create Food” menu items.
3.3 Main Screen

When EcoSystem-Sim initially starts, it appears as a Windows application. See Figure 3-1. Across the top there is a traditional Windows’ drop-down menu. The menu consists of the following items: File, Creature, Food, and Help. Underneath the drop-down menu is the toolbar. The toolbar is composed of the following buttons: New, Open, Save, Create Creature, and Create Food. At the bottom of the screen is a current event window displaying messages when events such as a birth or a death occur. Finally, on the right hand side of the application window is the status area. This section displays various state information in real-time. The following sections discuss each of these items in greater detail.

Figure 3-1: Main Screen of EcoSystem-Sim.
3.3.1 Drop-Down Menus/ Toolbar buttons

All of the EcoSystem-Sim commands can be accessed via the drop-down menus, or the toolbar buttons. The commands are as follows:

- **New**: Starts a new simulation. Can be accessed in the file drop-down menu or by the leftmost button on the toolbar.

- **Open**: Opens a saved simulation. Can be accessed in the file drop-down menu or by the second button on the toolbar.

- **Save**: Saves the current simulation. Can be accessed in the file drop-down menu or by the third button on the toolbar.

- **Create a new Creature**: Opens the Create Creature dialog box to allow the user to configure a custom creature and add it to the simulation. This option can be accessed via the Creature drop-down menu or by the fourth button on the toolbar.

- **Create new Food**: Opens the Create Food dialog-box to allow the user to configure a custom food object and add it to the simulation. This option can be accessed via the Food drop-down menu or by the last button on the toolbar.

3.3.2 Current Event Window

The user watches the current event window to stay abreast of the latest developments in the system. The window is located at the bottom of the screen and alerts the user when certain exciting events have occurred. These events include one creature eating another creature and when the last member of a food class perishes. Other important events that are relayed include creatures dying from starvation and births of
new creatures. A status window for up to the second statistical updates is described in the next section.

3.3.3 Status Panel

As a simulation progresses, state information contained in the system is updated continuously and displayed in the status panel. This panel is located in the upper right hand corner of the screen. The statistics displayed are as follows: the current number of creatures, the total number of creatures that have existed in the current simulation, the number of herbivores, the number of carnivores, the number of omnivores, the number of creatures that have died and why, and the number of creatures in each of the three color groups. Figure 3-3 shows a screenshot of this panel.

Figure 3-2: The Simulation Info Panel.
3.4 Create Creature Dialog

Creating a new creature is a very easy operation in EcoSystem-Sim. The Create Creature dialog box (see Figure 3-2) is accessed via the Creature drop-down menu on the main screen or by clicking on the “Create Creature” button on the toolbar. This dialog box allows the user to customize the new creature and then introduce it into the ecosystem. The Create Creature dialog box has been separated into eight numbered sections to make Creature creation easy. The user steps through the conveniently labeled fields filling in the required data. Once the form is complete, the user simply presses the “Create Creature” button in the last step, and the creature is inserted into a random location in the simulation.

Step 1: Creature’s Name

Every creature that is created manually, and not through procreation of creatures, has a unique name. This feature is only enabled so users can track their creatures.

Step 2: Gender

This option allows the user to select the gender of their creature.

Step 3: Food Preference

What the creatures eat in this virtual ecosystem falls into three categories: Herbivore, Carnivore, and Omnivore. Herbivores eat plant objects, which are dispersed throughout the environment. Carnivores eat other creatures in the field. Omnivores eat plants, or pursue other creatures. If the user chooses Omnivore, a slider can be adjusted to give the creature a stronger preference for either meat or plants.
Step 4: Social Type

The creatures in this simulation can be solitary creatures, or pack-type creatures. This option lets the user choose the type for the creature.

Step 5: Parameters

The parameters that can be set include speed, size, hunger, thirst, reproduction, aggression, strength, and fear. Each of these parameters affect how the creature behaves and how it interacts with its simulated world.

Step 6: Creature’s Color

Users can adjust the red, green, and blue sliders to select a custom color for their creatures. When the creature is inserted into the system, the creature can be identified by its color.

Step 7: Create Creature

The user should use this section to either cancel the Create Creature process or to proceed with the creation of the creature, inserting it into the ecosystem.

3.5 Interactivity in EcoSystem-Sim

EcoSystem-Sim was designed to be an interactive and entertaining software package. In the introductory chapter, several games were discussed that have brought a new type of gaming paradigm to the market. These interactive simulations merge a unique combination of artificial intelligence and interactivity. In this section we discuss how EcoSystem-Sim incorporates this interactivity.
3.5.1 Selecting and Influencing a Creature

All creatures in EcoSystem-Sim can be selected and influenced via keyboard input. To do this, the user must first put the system into creature selection mode, by choosing the select creature option from the creature drop-down menu or by pressing the select creature option from the toolbar. The mode is displayed in the mode status panel on lower left-hand corner of the screen. Once the system is in select creature mode, the user can click on any of the creatures in the system.

Once a creature has been selected, its vital statistics are displayed in the object view panel on the right hand side of the screen (see Figure 3-4). This information includes the following: creature ID, gender, food preference, current goal, health, age, hunger, reproduction, thirst, fear, strength, size, speed, and location. All of these
parameters are updated in real time and show the most current information for the creature.

To increase the depth of user interactivity, influencing the creature is possible once they have been selected. The user does this through keyboard input. The possible user commands include changing the creature’s goals, and influencing where it moves. The specific commands for influencing the creatures in EcoSystem-Sim can be seen in Table 3-1.

It is important to note that the creatures in EcoSystem-Sim have a very established hierarchy of needs, and so a user command is more of a request than an actual command. Thus, if a creature is hungry and the user tries to get it to eat, it may go into eat mode. However, the creatures in EcoSystem-Sim are self-preservationists and are very concerned for their own survival. Thus, the creature may decide that thirst is a higher priority than food, and go look for water instead.

This behavior is similar to the video game The Sims, described in Chapter 1. In this simulation, the user can try to get a Sim to read a book, but if the Sim is too tired and wants to go to bed, it will complain and go to bed anyway. In both The Sims and EcoSystem-Sim however, the majority of the time, the user’s commands will be heeded. This is necessary to give the user the feeling that their commands are doing something. Users can also interact with food and water objects in EcoSystem-Sim, which is discussed in the next section.
Figure 3-4: The Object Viewer Panel.

Table 3-1: Keyboard commands to influence creatures.

<table>
<thead>
<tr>
<th>Commands</th>
<th>Keyboard hot keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>Look for Food</td>
<td>&quot;E&quot;</td>
</tr>
<tr>
<td>Look for water</td>
<td>&quot;W&quot;</td>
</tr>
<tr>
<td>Look for mate</td>
<td>&quot;R&quot;</td>
</tr>
<tr>
<td>Move left</td>
<td>&quot;Left arrow&quot;</td>
</tr>
<tr>
<td>Move right</td>
<td>&quot;Right arrow&quot;</td>
</tr>
<tr>
<td>Move up</td>
<td>&quot;Up arrow&quot;</td>
</tr>
<tr>
<td>Move down</td>
<td>&quot;Down arrow&quot;</td>
</tr>
</tbody>
</table>
3.5.2 Selecting and Interacting with Food and Water Objects

To select either a food or a water object, the user must put the system into either food selection mode or water selection mode, respectively. This is accomplished by using either the drop-down menus or the buttons on the toolbar. The user is able to see what mode the system is in by looking at the mode display window in the lower left hand side of the screen. Once the system is in the proper mode, the user is able to view the properties of food or water objects in the object view panel on the right hand side of the screen. The user can then increment or decrement the quantity of food or water, or delete it altogether. Table 3-2 shows the keyboard commands for food and water objects.

<table>
<thead>
<tr>
<th>Commands</th>
<th>Keyboard hot keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>Increment amount</td>
<td>&quot;I&quot;</td>
</tr>
<tr>
<td>Decrement amount</td>
<td>&quot;D&quot;</td>
</tr>
<tr>
<td>Delete object</td>
<td>&quot;R&quot;</td>
</tr>
<tr>
<td>Move left</td>
<td>&quot;Left arrow&quot;</td>
</tr>
<tr>
<td>Move right</td>
<td>&quot;Right arrow&quot;</td>
</tr>
<tr>
<td>Move up</td>
<td>&quot;Up arrow&quot;</td>
</tr>
<tr>
<td>Move down</td>
<td>&quot;Down arrow&quot;</td>
</tr>
</tbody>
</table>

3.6 Conclusion

EcoSystem-Sim was designed to be a user-friendly application. One of the goals of this project was to create a software application that could be run by a user with no previous experience. The application was designed to start with only a few clicks of the mouse, and then gradually expose more features to the user. These features include
creating custom simulations and creating custom creatures and food objects. The next chapter discusses the internal architecture of EcoSystem-Sim.
CHAPTER 4
ECOSYSTEM SIMULATOR - IMPLEMENTATION PERSPECTIVE

4.1 Introduction

The EcoSystem-Sim package was implemented using the Document/View Model with several supporting classes. This chapter discusses the general object relationships, followed by an overview of key class inheritance structures. The section on general object relationships is a discussion of how objects are created at runtime, and how those objects interact with one another. The second section describes how objects in this virtual world are derived. Also discussed are some of the specific functionality those objects export.

4.2 Object Relationship Overview

EcoSystem-Sim consists of nine distinct object entities, each with their own specific purpose. These objects are as follows: EcoSystemSimDoc, CreateCreature, Creature, CreateFood, Food, CreateWater, Water, Statistics, and EcoSystemSimView. An overview of these objects and their relationships can be seen in Figure 4.1. This section discusses these objects and how they interact with each other during runtime.

4.2.1 EcoSystemSimDoc

EcoSystemSimDoc is the main controller class. All activity that occurs during runtime is dispatched and directed by EcoSystemSimDoc. For example, when a user
Figure 4.1: The Object Relationship Diagram - shows the interactions of the various objects in EcoSystem-Sim.

wants to create a new creature, it is this class that handles the message from the drop-down menu or button. EcoSystemSimDoc then instructs the CreateCreature object to display its dialog box, and create a new creature. The CreateCreature object passes EcoSystemSimDoc a pointer to the newly created creature object. Figure 4.1 displays this relationship. The black circles on the lines that lead into the Creature, Food, and Water objects indicate a one to many relationship. In other words, there can be many
Creature objects, Food objects, and Water objects, but only one EcoSystemSimDoc object.

EcoSystemSimDoc is also responsible for allowing creatures to live and for cleaning up after creatures die. At regular time intervals, EcoSystemSimDoc gives each of the active creatures some time to live and execute their code. When a creature dies, EcoSystemSimDoc frees that memory and removes the creature from its list of creatures. EcoSystemSimDoc also keeps track of all food and water objects in the same manner.

EcoSystemSimDoc also keeps track of the statistics object. As the game progresses, various statistics are generated, which EcoSystemSimDoc passes to the Statistics object for safekeeping.

Finally, EcoSystemSimDoc calls the EcoSystemSimView object to display all information to the screen. The EcoSystemSimView object is given pointers to the information that it needs. This information includes creature, food, and position information, as well as various statistics.

4.2.2 CreateCreature

CreateCreature is the class that is responsible for creating new creatures. This includes both custom creatures and those created via procreation. The central creature creation unit is the same in both cases; the source of the information is different.

When a create custom creature message is received, the CreateCreature object invokes the CreateCreature dialog box. After the user steps through the dialog and enters all the relevant information, that information is sent to the central creature creation unit to be processed. In the case of procreation, the EcoSystemSimDoc object sends the new creature information directly to the central creature creation unit. This unit allocates memory, and then uses the information to initialize the various data structures in the new
creature object. Once complete, the CreateCreature object passes a pointer to the new
data structure back to the EcoSystemSimDoc object.

4.2.3 Creature

Each creature in EcoSystem-Sim is allocated its own Creature object. It is in this
object that all state information pertaining to an individual creature is contained.

4.2.4 CreateFood/ CreateWater

These objects are similar in design to the CreateCreature object. They both have
their own dialog boxes, which are invoked by the user. Once all information has been
entered into the dialog box, the objects send the information to their core creation units to
be processed. After new objects have been created, pointers to those objects are passed
back to the EcoSystemSimDoc object.

4.2.5 Food/Water

Like the creature objects, each food or water object is also allocated its own
object. These objects contain all state information pertaining to that entity’s lifecycle.

4.2.6 Statistics

The statistics object is responsible for keeping track of all relevant data as it is
generated. Some of this information is gathered as it is created. This includes events
such as when a creature is born, or when one dies. This information could include such
items as what type of creature was born (omnivore, herbivore, or carnivore), or how a
creature died (from hunger, by being eaten, etc.). Other information is gathered at regular
time intervals. This information includes how many creatures are alive, how many are
carnivores, how many are herbivores, how many creatures have each of the carnivores
eaten, and so forth. These statistics are used to determine trends in a given simulation.
Some of these trends are discussed in experimental results section, in the next chapter.
4.2.7 EcoSystemSimView

EcoSystemSimView is the object that is responsible for displaying the current state of the simulation to the screen. This object is handed pointers to objects that contain information necessary to create the display. Thus, it can create a visual representation of what is occurring in the simulation at any given time.

4.2.8 Conclusion

The objects in the EcoSystem-Sim architecture are tightly coupled, yet modular in nature. By having the EcoSystemSimDoc object as the main control module, the rest of the hierarchy can be activated centrally. Additionally, because of the true modular nature of the system, it is a trivial operation to replace any of the objects that inhabit the ecosystem. The inheritance relationships of these objects are discussed in the next section.

4.3 Class Relationships

The objects that inhabit EcoSystem-Sim (food, water, and creatures) are all derived from the WorldObject class. This common base class exports several fundamental properties and methods that are used by each of the specialty classes. Additionally, since Carnivores, Herbivores, and Omnivores, are specific types of Creatures, they all extend the Creature class.
4.3.1 WorldObject Class

WorldObject is the base class for all entities in EcoSystem-Sim. It contains the common, basic information needed by all objects. These include properties and methods for determining location, age, active status, size, color, and name. By having all of the entities derive from this common class, a significant amount of code did not have to be rewritten for each object.
4.3.2 Food Class

The Food class is directly derived from the WorldObject class. This class contains the properties and methods necessary to track and incur growth of the object. Additionally, this class contains code to track when a creature is eating the food object.

4.3.3 Water Class

The Water class, also derived from the WorldObject class, is very similar to the Food class in its functionality. Like the Food class, the Water class also contains code to implement growth in the form of rain. This rain is generated randomly by the EcoSystemSimDoc object, and comes in the form of a function call to the food object. Finally, the Water class also exports functionality to keep track of when a creature is consuming some of the object.

4.3.4 Creature Class

Creature objects contain the state information of the individual creatures in the simulation. When a creature is born or created via the custom interface, it is associated with one of these objects. To create a Creature object, the EcoSystemSimDoc object sends a “create” message to the CreateCreature object. The CreateCreature object then instantiates the Creature object and initializes its parameters. When the Creature dies, it sends a “terminate” message back to the EcoSystemSimDoc object, which deletes the creature and cleans up the allocated memory. In this section, some of the responsibilities and actions of the Creature class are discussed. These include maintaining state, how a creature chooses its movements.

Each Creature object is responsible for maintaining its own state. This information includes numerous parameters such as location in 2D space, health, strength, size, and gender. It also includes a myriad of threshold parameters such as hunger, thirst,
reproduction, aggression, and fear. As the creature lives, it uses these parameters to help it decide what it wants to do at any given time.

Deciding when to move and where is also the responsibility of each individual creature. The state that the creature is in will determine where that creature goes and how that movement is calculated. There are six states that a creature can be in at any given time, and they each have a priority. These states are as follows:

1. Looking for water.
2. Looking for food (herbivore or carnivore).
3. Looking for a mate.
4. Running away from danger.
5. Doing nothing (just wandering around).

Thus, looking for water has the highest priority, whereas looking for food is the next highest priority. For example, if the creature becomes hungry, and changes from the doing nothing state, to the looking for food state, it will trigger several processes in the creature. First the creature will locate the type of food that it requires. It will then move towards that food. If the creature’s internal state changes such that it becomes excessively thirsty before it is able to eat, its state will change from looking for food, to looking for water. At that point the creature will begin the process of locating a water object, and moving towards it. Thus, each creature follows this prioritized need structure.

4.4 Conclusion

EcoSystem-Sim is built on a sound foundation of solid architectural principles. A significant portion of the design time invested in the creation of EcoSystem-Sim was devoted to architectural design. These solid design principles allowed for EcoSystem-
Sim to function as it was intended and to perform specific simulations. These scenarios and their results are discussed in the following chapter.
CHAPTER 5
EXPERIMENTAL RESULTS

5.1 Introduction

This chapter addresses some of the results that have been achieved by EcoSystem-Sim. EcoSystem-Sim was designed to reproduce trends normally seen in populations, in an accurate manner. Accurate reproduction of these features was necessary to show that EcoSystem-Sim is a credible ecosystem simulator. This chapter discusses these trends, and how they relate to their naturally occurring counterparts. Trends that are discussed are the exponential growth curve of populations and the boom/bust growth cycle, out-of-phase growth cycles of two populations, and genetic drift.

5.2 Exponential Growth Curve of Populations and the Boom/Bust Growth Cycle

During the testing phase of EcoSystem-Sim, several different population scenarios were run. These included an all herbivore population, an all carnivore population, and a mixed population. All scenarios produced similar results in terms of population expansion.

The Englishman Thomas R. Malthus first proposed a mathematical model for population growth in 1798. Commonly called the natural growth model or exponential growth model, it has become a basis for the modeling of biological populations (see Figure 5-1.). This model assumes that a population grows at a rate proportional to itself. In other words, the larger the population, the higher the rate of growth. This model also assumes unchecked growth and that growth is not constrained by environmental factors.
such as disease or famine [Moo01]. However, when these factors are considered, a boom/bust growth cycle begins to emerge.

This boom/bust growth pattern begins with an exponential growth of a population. Once a population becomes too large, the ecosystem can no longer support it and fall-off occurs. Typically, this fall off is just as abrupt as the expansion phase of the cycle. Because of damage done to the ecosystem, a considerable amount of time may have to pass before the population can rebound from its lows.

In Chapter 2, it was noted that during the summer months, algae underwent a dramatic population expansion. Then, at the end of the summer the population would fall off again, and the algae would merely survive the remainder of the year. This decrease in the algae population was due to resource depletion. It takes the rest of the year, when the algae are dormant, for the resources it needs to be replenished. This same type of resource depletion occurs in EcoSystem-Sim. When the population of herbivores becomes so large that plants are eaten down as soon as they grow, the plants go into a dormant stage. This dormant stage takes long enough that the herbivore population falls off significantly. Eventually, the plants start to grow again, and the herbivore population rebounds. Results from this simulation can be seen in Table 5-2.

5.3 Out-of-phase Relationship

When two populations are directly dependent upon each other, an out-of-phase relationship can occur. The nature of this type of relationship was initially discussed in Section 2.2, and is elaborated on here. It was noted by [Aud02] that a perfect sine/cosine relationship does not actually occur in nature. She notes that while it is very apparent
Figure 5-1: The Exponential Growth of the Herbivore Population.

Figure 5-2: Seasonal Boom/Bust Cycles.
that two populations influence each other, in nature there are simply too many factors affecting population changes. The Bean weevils/Branconid wasps example discussed in Section 2.2 was conducted in a laboratory in a closed system. In other words, there were no other species involved in the simulation. It is this type of environment that is being replicated in EcoSystem-Sim.

It is interesting to note that a two species mathematical model was developed to demonstrate this predator-prey relationship. This model, called the Lotka-Volterra model, was proposed in 1925 by the American biophysicist Alfred Lotka and the Italian mathematician Vito Volterra [McK95]. With EcoSystem-Sim, the idea was to approximate this model without using their specific formula. The creatures in EcoSystem-Sim are born because of reproduction, and they die because of hunger, because they are eaten, or because of old age. Additionally, there is a prominent random element in EcoSystem-Sim that is not present in the Lotka-Volterra mathematical model. This random element comes both from the choices the creatures make, and from what traits their offspring inherit.

This random element means that the population does not always behave in a manner that fits the model exactly. When the simulation begins and the user chooses to run the out-of-phase growth simulation, the ecosystem is started with a set of creatures. Extensive testing was conducted to determined what set of creatures was necessary to produce the out-of-phase model the most consistently. Table 5-3 shows the results of one 100-minute run of the program.

In this example, the herbivore population fluctuates more drastically than the carnivore population. This could be due to the fact that the herbivore population is
directly connected to the plant objects in the simulation. The plant objects take a certain amount of time to grow back once they have been eaten down. Thus, when the herbivore population reaches a certain number (about 82), the plant objects have all been eaten down and take a while to grow back. It is at this point that the herbivore population goes into a freefall until the food has a chance to grow back. The carnivore population gradually follows the herbivore population. However, since their food source (the herbivores) is always present, their population count does not fluctuate as wildly.

![Carnivore vs. Herbivore populations](image)

**Figure 5-3: The Exponential Growth of the Herbivore Population.**

5.5 Genetic Drift

As stated in Chapter 2, genetic drift is the process whereby an entire population adopts one genetic change. EcoSystem-Sim was implemented to incorporate this type of event into its simulations. Table 5-4 shows the results of one simulation where genetic drift occurred. In this example, the gene for the color blue was marginalized and subsequently made obsolete by the gene for the color red.
Each creature in EcoSystem-Sim carries two genes for color, one from each parent. There are three possible colors that a creature could be: red, blue, or a custom color. The gene for red is dominant over the gene for blue, and any gene for a custom color is dominant over both genes for red and blue. For example, since the gene for blue is recessive, a creature only has the blue trait, if it receives genes for blue from both the father and the mother. However, if the offspring receives just one gene for red it will be red.

When a creature mates, there is an equal probability that either one of its two genes for color will be passed along. Thus, if two red creatures mate, and each carries one gene for red, and one gene for blue, there will only be a 25% chance that the creature would be blue (see the Punnett Square in Table 2-5). It is this disparity in the probabilities, 25% vs. 75%, which is the source of catalyst for genetic drift.

An additional example of this same type of genetic drift can be achieved using a custom creature. To do this, a user must create a creature with two genes for a custom color, and insert it into the system. If this creature is sufficiently healthy, its genes will propagate through the system, and push out other recessive genes.

Genetic drift does not occur in every simulation however, since it is a product of probability. One of the primary requirements for genetic drift to occur is that the population size must be small. There is no specific number for genetic drift to occur. It is simply that the smaller the number of offspring, the higher the probability that only the dominant genes will survive.
Figure 5-4: Genetic Drift in a Population.

5.6 Conclusion

This chapter discussed some of the possible characteristics of simulations run under EcoSystem-Sim. Any of these trends may be experienced during the course of running the application. Certain pre-created simulations, which may favor one trend over another, can be loaded when EcoSystem-Sim first starts. Optionally, any of these pre-set scenarios can be run from the simulations drop-down menu. These simulations were designed to show the capabilities of the system, and prove that the system is capable of supporting certain characteristics of a real ecosystem. The next chapter summarizes the EcoSystem-Sim project and discusses where it could go in the future.
CHAPTER 6
SUMMARY AND FUTURE WORK

6.1 Summary

Interactive simulations have become very popular forms of entertainment in the last few years. Games that have been introduced into the market recently, such as The Sims, and Creatures, have had a significant impact on what the game-buying public perceives as entertaining. In general, these games have moved away from traditional gaming themes to more varied types of interactive entertainment.

EcoSystem-Sim is a prototype application that was written in this new style of game programming. The application is a proof-of-concept for an advanced, interactive, ecosystem simulation application. Users can design and customize an ecosystem to suit their needs and desires. A variety of configurable parameters are available for users who wish to create custom creatures, food, and water objects.

Once an ecosystem has been built, the user can then interact with all elements of the system. Users can nurse their ecosystems to a healthy thriving state, or let them fall into decay. To add an additional level of personalization, users can even name their creatures, and create unique colors for them. Once a user has created and populated an ecosystem, they can save it to disk, and come back to it at any time.

Although EcoSystem-Sim is entertainment software, and not a scientific simulation, it was important to adhere to the scientific principles of biology and Artificial Intelligence. Modeling genetic theory and population expansion as accurately as possible
was paramount in EcoSystem-Sim. Thus, it was important to be able to produce various trends while running simulations, such as genetic drift and out-of-phase population expansion, as accurately as possible. This level of reality makes the interactivity of the game much more interesting.

### 6.2 Contributions

EcoSystem-Sim merges ecosystem simulations, and interactive game programs to create an interactive ecosystem simulator. There have been attempts in the past to create virtual ecosystems, as noted in chapter 2, but none of these systems was designed to be entertaining and interactive. Alternatively, there have been commercially available interactive simulations, as noted in chapter 1, but none of these has focused on modeling an ecosystem.

EcoSystem-Sim is the integration of these two ideas into one complete package. As a prototype for a potentially commercially viable product, it has a foot in both the entertainment and education arenas.

### 6.3 Future Work

This section discusses some of the ideas for future extensions of EcoSystem-Sim. Because the underlying nature of this application is entertainment, it is important to recognize what aspects of the product would need to be enhanced to bring it into the commercial arena. These needs are as follows: An enhanced graphical user interface, modularity and clearly defined interfaces, Group/pack behavior, AI enhancements, Physical barriers and landscape variations, Advanced evolutionary enhancements, and Aggression and Attack mode enhancements.
6.3.1 Enhanced Graphical User Interface

Advanced graphics is a critical component in the entertainment software market today. Users expect to see very intricate and attractive graphics. If EcoSystem-Sim were to be released as a commercial product, everything from the representations of the creatures to the buttons and display would need to be upgraded. Graphics of this quality are very expensive and take a lot of time to create, and thus, far beyond the scope of this thesis.

6.3.2 Modularity and Clearly Defined Interfaces

The architecture was designed in such a way that the front end and the back end are separate modules. The intention of the basic design of the graphics front-end, EcoSystemSimView, was to allow it to be decoupled from the rest of the application, and replaced with a commercial grade product. However, because it is impossible to anticipate what parts of the code would be kept, and what would be scrapped, the interfaces have not been clearly defined. However, the software is object-oriented in nature. Once commercial requirements could be established, the object-oriented structure of the application would assist greatly in creating standardized interfaces, and modular units.

6.3.3 AI Enhancements

A significant amount of work went into creating the intelligence for the creatures in EcoSystem-Sim. However, given a significant amount of time, a number of enhancements could be added to the intelligence algorithms. Some of the areas where these enhancements could take place are as follows: line of sight and barrier detection, territorial behaviors, mate selection, and group or pack roaming and hunting activities.
**Line of sight and barrier detection:** When a creature examines its world, the current algorithms currently give it 360-degree vision. It would be interesting to limit this to 180 degrees, with less accuracy towards the periphery of the creature’s vision. This enhanced vision algorithm could also lead into the creatures having the ability to detect barriers or objects that would limit their mobility in the world.

**Territorial behaviors:** Currently, when a creature wants to look to see if there is any food to eat, the creature simply examines all of the food objects in the area and chooses the closest one that has not already been eaten. What would be much more interesting would be if creatures could stake out a section of their world as their territory. Then only that creature and possibly its mate(s) would eat food from that section. If food got scarce, the creatures could invade other creature’s territories, and thus expand or migrate their own territory.

**Mate selection:** When a creature wishes to reproduce, it chooses its mate simply by finding the closest creature that is of the opposite sex, and which it can catch. It would be an interesting feature if mates could be taken on a more permanent basis. In other words, once creatures choose each other as their mates, they keep them permanently, or until one dies. This type of mate selection would naturally lead to groups or packs in the ecosystem.

**Group or pack roaming and hunting activities:** The idea of having creatures working together for survival is a very exciting idea. This would bring a whole new dynamic element to the application. Users could create creatures and add them to competing groups, thereby setting the stage for conflict and interaction between groups. Part of the enhancements to the algorithms would be to add flocking patterns, and
hunting patterns to the group’s motions. For example, when a pack of carnivore creatures is chasing prey, they could participate in group-hunting patterns that would be more interesting than simple pursuit.

6.3.4 Physical Barriers and Landscape Variations

With the addition of barrier detection to the AI of the creatures, the next logical step would be to make the landscape more interesting by adding naturally occurring barriers such as rivers, canyons, hills, or valleys. This could also support the group or pack tendencies that were also discussed in section on AI enhancements. These physical barriers could promote the creation of colonies, which are separated by these physical barriers. Once the population size of the creatures in these colonies got too large, and their existing food sources ran out, they could attack other colonies on other landmasses.

6.3.5 Aggression and Attack Modes

EcoSystem-Sim does not currently support any types of attack modes for the creatures. Creatures do attack each other in the sense that carnivores can eat other creatures. However, once an attacker catches its prey, the two creatures are simply evaluated to determine which one is larger and stronger. If the larger, stronger one is the attacker, it eats the prey. If the stronger one is the prey, the attacker simply goes to look for another creature.

It might be more interesting to create a more involved description of these attacks. This does not mean that blood and gore are necessary. Rather that the attack is broken down to a series of strikes and counter strikes, in which each creature sustains damage until one dies. This would also help to separate the strong from the weak, and promote evolutionary tendencies.
6.3.6 Advanced Evolutionary Enhancements

As a simulation progresses, it would be interesting to integrate Spontaneous mutations, evolutionary changes (length of creatures, new color, etc.). The kinds of mutations that could occur would only be limited to the imagination of the programmers. By adding this type of feature to the system, a truly robust package would be created.

6.3.7 Ultimate Goal, World

EcoSystem-Sim was designed to be the foundation and prototype for what could be a very advanced, intricate and robust system. The end product is one that very closely models a real ecosystem. The title of the software would be World, and would start at the beginning of life on the world. The user would coax primitive life forms out of the primordial ooze, and assist and influence them as they evolve. As the simulation progressed, life forms in the simulation would evolve into increasingly more complicated creatures. Each simulation would produce new and varied creatures, depending on the participation and interaction of the user.

This thesis has presented a prototype application that was designed to be a proof-of-concept for an advanced, interactive, ecosystem simulation application. It affords a wealth of functionality and features such as customizable creatures, easy to use interfaces, and the ability to influence any of the creatures in the system. With the ideas that have been discussed in this section, the EcoSystem-Sim is ready to evolve into commercial quality software application.
LIST OF REFERENCES


BIOGRAPHICAL SKETCH

Roderick Simpson was born in Bryan, Texas, on November 13, 1970. He received a bachelor’s degree in fine arts from the University of Florida in 1995. He returned to the University of Florida in 1998 to pursue a master’s degree in the Department of Computer and Information Science and Engineering. His research interests include all aspects of computer game programming, artificial intelligence, and digital simulations.