SOCIAL SCIENCE APPLICATIONS OF DISCRETE EVENT SIMULATION:

A DEVS ARTIFICIAL SOCIETY

by

Gordon Christopher Zaft

A Thesis Submitted to the Faculty of the

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

In Partial Fulfillment of the Requirements
For the Degree of

MASTER OF SCIENCE
WITH A MAJOR IN ELECTRICAL ENGINEERING

In the Graduate College

THE UNIVERSITY OF ARIZONA

2 0 0 1

STATEMENT BY AUTHOR

SIGNED: ___________________________________

APPROVAL BY THESIS DIRECTOR

This thesis has been approved on the date shown below:

_______________________ _______________________

Bernard P. Zeigler
Professor of Electrical and
Computer Engineering

Date

ACKNOWLEDGMENTS

I would like to thank my advisor, Professor Bernard P. Zeigler, and Dr. Hessam Sarjoughian for their support, patience, and unfailing good humor during the long effort to complete my research and thesis. Not many people would have been willing to engage in such a protracted endeavor, which makes their kindness even more remarkable.

Thanks are also due Dr. Salim Hariri for taking the time to be on my committee. His graciousness is much appreciated.

I would also like to thank Dr. Jennifer Croissant for challenging me to question, examine, and analyze more deeply. She opened a whole new world of social science for me.

My efforts to attain the highest quality writing were aided by Rhonda Knotts, Marilyn Martin, and Dr. Janalee O'Bagy, who puzzled through a quite-rough draft in order to give me invaluable feedback; for their time, talent, and friendship I am truly grateful. Of course, the blame for any remaining faults is mine alone.

I would be remiss if I did not thank my employers at both Artisoft and NewMonics for their encouragement and support during the course of my graduate school career. In particular, Eileen Daily at Artisoft and Randy Rorden at NewMonics gave more help than they probably realized.

Special thanks are due to Maria Lorentzen for her support and encouragement, first to start my graduate studies, and then to complete them. Key parts of my research were completed while a guest in her home.

Finally, a debt of gratitude is due my family. Their combination of encouragement, support, and skepticism proved strangely beneficial.

DEDICATION

For my mother, who showed me the way.

TABLE OF CONTENTS

TABLE OF CONTENTS — Continued

TABLE OF CONTENTS — Continued

LIST OF FIGURES

LIST OF FIGURES — Continued

LIST OF TABLES

ABSTRACT

Social scientists use artificial society simulations to explore complex behaviors that result from the interaction of agents. One such artificial society is Epstein and Axtell's Sugarscape simulation. In Sugarscape, agents are born, eat 'sugar', travel, reproduce, and die in a torus-shaped virtual world. Sugarscape uses simple rules to create a virtual society where agent interactions aggregate to form surprisingly complicated social structures.

This thesis presents XeriScape, a Sugarscape-style artificial society based on the Discrete Event System Specification (DEVS) formalism implementation in the Java language (DEVS-Java). DEVS-Java is a powerful tool that makes XeriScape a more efficient, flexible, and extensible artificial society simulation than Sugarscape. A number of experiments illustrate the capabilities and advantages of the DEVS-Java environment for artificial societies and other social science applications.

1 INTRODUCTION

XeriScape – used for a water-conserving method of landscaping in arid or semiarid climates
— *Merriam-Webster's Collegiate Dictionary*, Tenth Edition (1998)

1.1 Rationale

This thesis attempts the melding of two disparate worlds: social science research and object-oriented modeling and simulation. In the intersection of these worlds lies a new application of discrete event simulation technology, and specifically the Discrete Event System Specification (DEVS) developed by Zeigler [1]. This application is the software system called XeriScape.

Epstein and Axtell's Sugarscape project [2] received much attention in the popular press [3] [4] and among social scientists. While social science and high performance simulation are separate worlds, the attention Sugarscape received raised the possibility of applying DEVS technology to a Sugarscape-style simulation. This possibility is intriguing for two reasons: first, because DEVS technology has primarily been applied to problem spaces in the physical sciences, and second, because the power, flexibility and efficiency of DEVS could result in a very capable system. In applying DEVS to an artificial society, it would be possible to expand the DEVS problem space in a fruitful and interesting way – one that would be beneficial both to the social science community and to simulation scientists.

The name XeriScape reflects the Sonoran desert that is the setting of the University of Arizona. In Epstein and Axtell's work, sugar is the resource for which all the agents compete. As desert people, water is the resource that rules our lives and is the basis of our civilization.

1.2 Research Questions

The research questions explored in this thesis are:

- Can DEVS be applied to artificial society simulations?
- Can DEVS do artificial society simulations efficiently and quickly?
- Can a DEVS implementation answer other social science questions quickly and easily?
- Can a DEVS artificial society simulation provide a more general and extensible foundation for further research?
- Can a DEVS artificial society provide a feedback mechanism that allows agents to grow and learn, and to pass on what they have learned to their offspring?

1.3 Outline of the Thesis

Chapter 2 provides background information on artificial societies, and specifically on the work of Epstein and Axtell that is the starting point for this thesis. Chapter 2 also includes more information on modeling and simulation in general as well as on the DEVS formalism developed by Zeigler.

Chapter 3 discusses the XeriScape system simulation model, its structure and underlying concepts from the perspective of a social science model.

Chapter 4 examines the XeriScape system from a software simulation perspective. The DEVS-Java implementation of the XeriScape model is explained in some detail, and its structure is described.

Chapter 5 details the experiments undertaken with the XeriScape system and relates them to Epstein and Axtell's work. The results of the experiments are presented and discussed.

Chapter 6 presents conclusions about the XeriScape system and the experiments presented in this thesis; it re-examines the research questions presented in section 1.2 above and explores the answers presented in this thesis. Finally, Chapter 6 also outlines some possibilities for future extensions of the XeriScape system and further artificial society and social science research applications for DEVS.

Appendix A provides information on obtaining the XeriScape code and example configurations. Appendix B documents the configuration file format used by XeriScape. Appendix C offers some comparisons and contrasts between the XeriScape implementation and Epstein and Axtell's Sugarscape.

2 BACKGROUND

In this chapter, two important concepts are presented. The first concept is artificial societies, with specific emphasis on Epstein and Axtell's Sugarscape research. The second concept is modeling and simulation, focusing on Zeigler's DEVS formalism.

2.1 Artificial Societies

Artificial society simulations are an outgrowth and specialization of two more general simulation paradigms, artificial life and agent-based modeling. Artificial life simulations (the classic example is Conway’s Game of Life [5]) simulate an individual life span; while there may be multiple agents, they are usually considered individually. Agent based modeling involves a number of agents that interact in a simulated space of some kind.

2.1.1 Overview of Artificial Societies

An artificial society is distinguished from artificial life simulations by the emphasis on the communal aspect of the simulation; artificial societies focus not on individuals but on the society as a whole. They may be composed of non-intelligent agents or intelligent agents. One early artificial society is BioLand [6], a massively parallel distributed simulation of animal behavior that focused on exploring the capabilities afforded by the availability of a large number of processors.

Early social science attempts at artificial societies were not as sophisticated in their hardware and software. Some examples are Nowak and Latané’s work on emergence of social order [7] and Drogoul and Ferber’s simulation of an anthill [8]. The attempt to bring more sophistication to artificial societies is an ongoing one since the field is still immature. Such basic questions as “what is an agent?” are still being defined [9] and there is a continued effort to establish basic tools and techniques [10].

Some particular ways in which artificial societies differ from artificial life and other traditional simulations are given by Epstein and Axtell [2, pp. 14 - 17]:

- *Heterogeneous agent populations*. Instead of a population where each agent behaves the same as every other agent, artificial societies are composed of agents which have their own unique genetic makeup, cultural traits, and experiences.

- *Space distinct from agents*. Artificial societies have a spatial component that is distinct from the agents themselves. Agents interact with the environment and with each other.

- *Interactions according to local rules*. The interactions among agents and between the agents and the environment are determined according to rules that are local to the agents themselves.

2.1.2 Epstein and Axtell's Sugarscape

Epstein and Axtell developed the Sugarscape simulation over several years at the Santa Fe Institute in an attempt to bring a new methodology to social science simulations. They applied agent-based technology in an attempt to build an artificial society "from the bottom up." That is, instead of the previously common approach of imposing broad classes of rules on a simulation, they define simple low-level rules of behavior for the agents and observe what structures emerge:

> …the defining feature of an artificial society model is precisely that *fundamental social structures and group behaviors emerge from the interaction of individual agents operating on artificial environments under rules that place only bounded demands on each agent's information and computational capacity*.[2, p. 6, emphasis in original]

Thus the subtitle of their book, "Social Science from the Bottom Up." In order to grow their simulation their emphasis was primarily to define simple, readily understood rules for agent interaction.

Sugarscape uses the Swarm agent-based simulation system [11] [12]. Swarm combines object-oriented technology with agent-based modeling to produce a much more capable toolkit for social science simulation than previous toolkits [13] [14] [15]. Sugarscape was developed on a Macintosh computer, and is composed of approximately 20,000 lines of code.

The Sugarscape is made up of a 50×50 grid of cells that are the environment for the simulation. On this grid are agents which gather and consume sugar in order to live; they trade, reproduce, make war, form tribes, and so forth. They and their environment follow a few simple rules that govern their behavior. Epstein and Axtell adopt a simple notation for these rules, some 17 of which are described in their book. The rules that are applicable to XeriScape are listed below [2, pp. 182-183]:

$\mathbf{G}_{\alpha}$ Growback rule. In each cell sugar grows back at a rate of α units per unit time.

M Movement rule. Each agent looks as far as its vision permits in the four directions it can see, finds an optimum location, moves there, and collects all the resources available.

$\mathbf{R}_{[a,b]}$ Replacement rule. When an agent dies, whether by starvation or old age, it is replaced by a new agent with a given genetic endowment and random position. Agents have a lifetime randomly distributed in the range [a,b].

$\mathbf{S}_{\alpha\beta\gamma}$ Seasonal growback rule. Seasons vary between summer and winter, each season is γ time periods long. The growback rate is α units per time period in summer and α units per β time periods in winter.

P$_{\alpha\beta}$ Pollution rule. Pollution is produced at a rate of $\alpha s + \beta m$, where α is a multiplier and s is the amount gathered, β is another multiplier and m is the amount consumed.

D$_{\alpha}$ Diffusion rule. Pollution is diffused from a cell at a rate of α time periods.

S Sexual reproduction rule. If the agent is fertile and has sufficient resources, it attempts to mate with each of its neighboring agents if they are also fertile, have sufficient resources, and are of the appropriate sex, so long as there is an empty cell available for the child to be produced.

The Sugarscape model provides significant data visualization and data extraction/reduction facilities. These include the ability to call up multiple windows as the simulation is running to display such data as social networks of neighbors, Lorenz curves and Gini coefficients, age and wealth histograms, and much more. The model also has a graphical user interface that provides the user a visual display of the agents in the Sugarscape as the simulation runs.

2.2 Modeling and Simulation

2.2.1 General Concepts of Modeling and Simulation

Zeigler [1, pp. 27-32] identifies five seminal concepts of modeling and simulation. They are:

1. The *real system* – the system that we are attempting to model. For an artificial society, the real system is usually a past, present, or possible future human society.

2. The *experimental frame* – the set of circumstances that are of interest for the purposes of the simulation. For example, in XeriScape we are not interested in such things as race, religion, diet, hair style, etc., so they are excluded from our experimental frame. More broadly, the concept of experimental frame can refer to the configuration or set of components of the model (Ch. 5 uses the term in this sense). The model need only be *valid* (that is, an accurate representation of the real system behavior) within the experimental frame in order to be useful.

3. The *base model* – a model that is capable of accounting for all the behavior of the real system for all allowable experimental frames. Often it is not possible to actually create a base model due to its immense complexity. This is especially true in social science applications.

4. The *lumped model* – the relatively simple model constructed for the specified experimental frame. This is the model that will actually be simulated.

5. The *computer* – more generally, the computational device used to generate input and output from the model. This can also be usefully thought of as the *simulator*. It is important to keep in mind that the model is an abstraction,

separate from its potential implementation as a computer program. It is common to identify the model with its particular implementation (this shorthand is often used in this thesis) but they are, strictly speaking, distinct.

One other related, important concept is the cellular automaton. This concept had its origins with the great thinker John von Neumann [16] but was developed most fully by Wolfram [17]. Cellular automata consist of a grid of cells each having a set of values. The essential aspect of this concept is that the future values of each cell depend on the current value of the cell, <u>and</u> the current values of its neighbor cells. Thus, looking back to section 2.1.2 it becomes evident that the Sugarscape is a combination of cellular automata and agents.

2.2.2 Discrete Event Simulation

The DEVS formalism was developed by Zeigler in order to provide a solid foundation for discrete event simulation. Discrete event simulation is a technique that, as the name implies, is based not on time steps but on events. Where a discrete time simulation has a clock that steps in even increments (1, 2, 3 …), a discrete event simulation steps from event to event (A, B, C, …). Although discrete event simulations do have a clock, it is not constrained to step in even increments, but from the time of the first event to the time of the second event, and so forth.

The classic discrete event system specification [18, pp. 75-76] is a structure

$$M = \{X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta\}$$

Where:

X is the set of input values
S is the set of states
Y is the set of output values
$\delta_{int}: S \rightarrow S$ is the *internal transition* function
$\delta_{ext}: Q \times X \rightarrow S$ is the *external transition function*, where
$Q = \{(s,e) \mid s \in S, 0 \leq e \leq ta(s)\}$ is the *total state* set
e is the *time elapsed* since last transition
$\lambda: S \rightarrow Y$ is the output function
$ta: S \rightarrow R^{+}_{0,\infty}$ is the set of positive reals[1] from 0 to ∞

Briefly stated, at any given time the system is in some state s. If no disturbing event occurs, the system will stay in that state until time *ta* has elapsed; at which point the system will process its output ($\lambda(s)$) and then change to state $\delta_{int}(s)$ (i.e. an internal transition occurs). If an external event occurs, the system will change to state $\delta_{ext}(s,e,x)$.

DEVS models may be *atomic* (i.e. a single, independent model) or *coupled* (models hierarchically composed of other models). The particular strength of DEVS is that it has been shown to be closed under coupling, that is, if a model A is composed of models B, C, and D, and B, C, and D are valid DEVS models, then A is also a valid DEVS model. This ability to compose complex models from simple building blocks makes it possible to build very large models.

[1] DEVS-Java 1.5 time steps are integers, however, not real numbers.

The DEVS formalism has spawned a very large body of work and the extensive DEVS literature is impossible to summarize in this thesis. Two particular DEVS variations are worthy of note in this context, however, since they relate to XeriScape design choices.

The first of these is Barros' work on dynamic structure DEVS [19] [20]. Barros extended DEVS to include the ability for DEVS models to change their structure and still maintain an unambiguous network structure that was closed under coupling. This Dynamic Structure Discrete Event (DSDE) formalism was considered for the XeriScape system architecture but ultimately a static-structured DEVS architecture was chosen.

The second DEVS variation of interest is the work of Wainer and Giambiasi [21]. They described a variation of DEVS oriented toward cellular automata that allowed for transport and inertial delays between cells. Coincidentally, their work was conducted at the same time that XeriScape was being developed, and was not available for consideration until after the XeriScape design was complete.

3 THE XERISCAPE MODEL

3.1 Overview

The XeriScape model is composed of agents, cells, and rules that interact to define and guide the behavior of the XeriScape. Briefly stated, agents live in a space (the XeriScape) made up of cells. Their behavior, and the behavior of the 'environment' in which they exist, is regulated by a number of rules which interact to control the behavior of the agents and cells. XeriScape components are very general and are easily extensible.

The XeriScape Model

- Agents
- Cells
- Rules

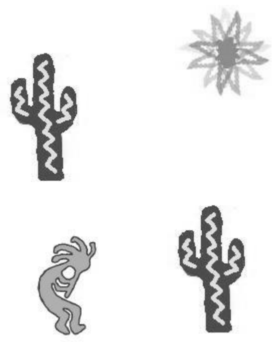
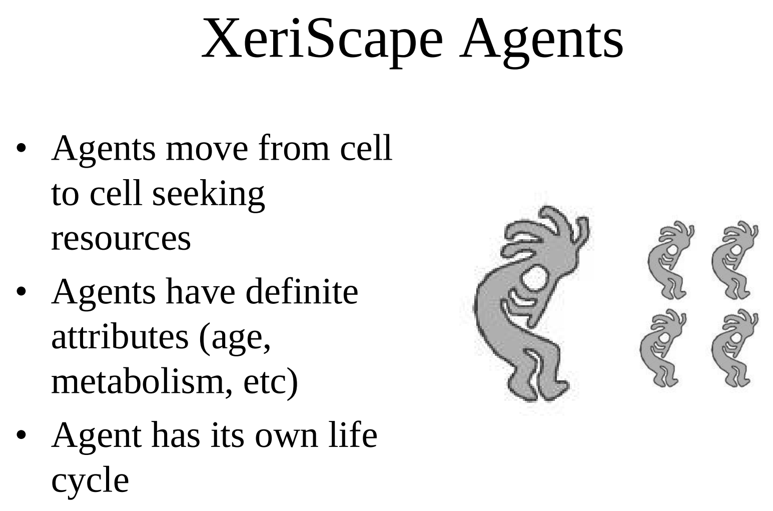


Figure 1. The XeriScape Model

3.2 Building Blocks

3.2.1 Agents

Agents are the dynamic part of the XeriScape. Their actions and interactions create the artificial society of the XeriScape. Agents move from cell to cell seeking resources such as water. Each agent is a unique individual with its own life cycle, attributes, history, and resources (see Figure 2).



XeriScape Agents

- Agents move from cell to cell seeking resources
- Agents have definite attributes (age, metabolism, etc)
- Agent has its own life cycle

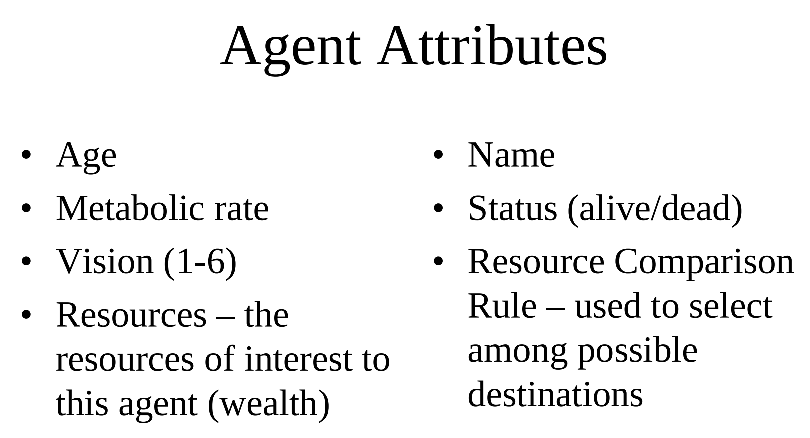
Figure 2. XeriScape Agents

For the purposes of this thesis, there are four different types of agents to consider – the basic agent definition (Agent) and its extension into three additional agent types: the

Finite Agent, the Polluting Agent, and the Gendered Agent. These four agent types share most properties in common.

3.2.1.1 Agent

Each Agent has a number of attributes that define it. The attributes that every agent has are shown in Figure 3 below. Agents move from cell to cell, seeking resources they need, gathering them and consuming them. Agents have a limited vision and movement range. Vision and movement are linked; a vision of *n* cells implies a movement capability of *n* cells as well. The agent has an unlimited lifespan; given enough resources, it is immortal. Each agent has its own rate of resource consumption (metabolic rate).



Agent Attributes

- Age
- Metabolic rate
- Vision (1-6)
- Resources – the resources of interest to this agent (wealth)
- Name
- Status (alive/dead)
- Resource Comparison Rule – used to select among possible destinations

Figure 3. Agent Attributes

The agent life cycle is shown in Figure 4 below.

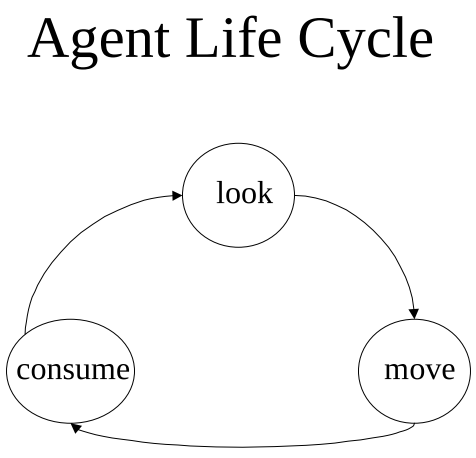


Figure 4. Agent Life Cycle

3.2.1.2 Finite Agent

The Finite Agent is an agent that has a definite, determined lifespan. Finite agents live to be a specific age that is determined when they are created; then they die.

Finite agents have one attribute in addition to the base attributes shown in Figure 3, namely, their maximum age (i.e., the age at which they will die).

3.2.1.3 Polluting Agent

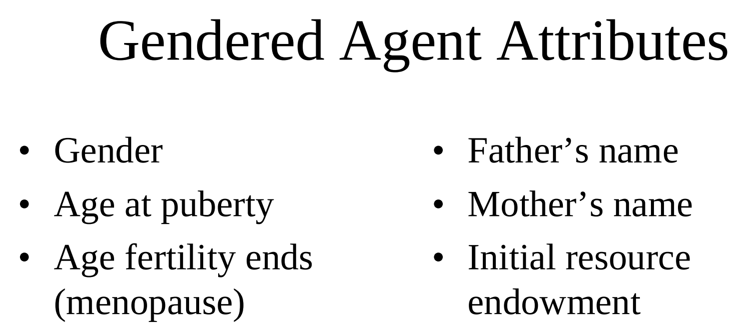
The Polluting Agent is a finite agent that creates pollution. Whenever the polluting agent gathers resources or consumes them, pollution is produced as a side effect. The amount of pollution produced is a multiplier of the amount of resources gathered or consumed and a constant determined when the agent is created.

The polluting agent is sensitive to the pollution it creates. When a polluting agent is deciding where to move, it tries to avoid moving to areas that are polluted. In making its movement decision the polluting agent takes into account the amount of pollution in possible destinations and the amount of water available; all other factors being equal, it will move to the site with the least pollution.

Polluting agents have two attributes in addition to the finite agent attributes: the resource-gathering pollution multiplier constant α, and the resource-consumption pollution multiplier constant β. These constants are used to compute the amount of pollution produced by the agent.

3.2.1.4 Gendered Agent

The Gendered Agent is a finite agent that has gender and the ability to reproduce.



Gendered Agent Attributes

- Gender
- Age at puberty
- Age fertility ends (menopause)
- Father’s name
- Mother’s name
- Initial resource endowment

Figure 5. Gendered Agent Attributes

Gendered agents have a number of attributes in addition to those of the finite agent. They include age at puberty, age at which fertility ends (menopause), and the names of the father and mother. These attributes are detailed in Figure 5.

The gendered agent has a modified life cycle as shown in Figure 6. After the standard look, move, and consume activities, the gendered agent will look to its neighboring cells to see if a suitable mate is available. If the gendered agent is fertile and one or more fertile neighbors is found, mating will occur with the first suitable partner found.

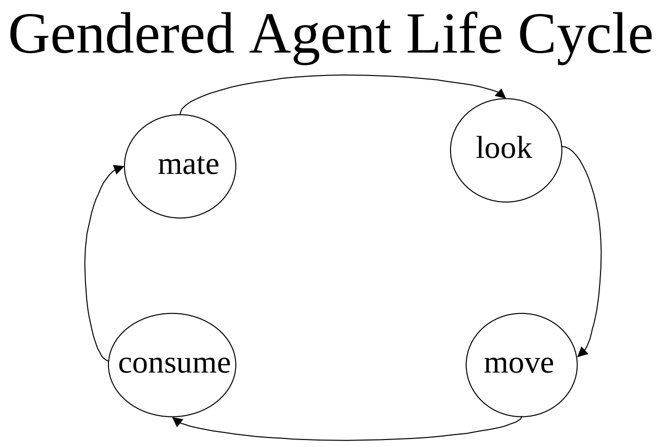


Figure 6. Gendered Agent Life Cycle

Gendered agent fertility is determined by a number of factors. First, the agent must be of childbearing age, i.e., older than puberty and younger than menopause. Second, the agent must have an amount of water resources (wealth) equal to at least half of the amount it had when it was born. These resources (half each from mother and father) are given to the child at birth. Children’s vision and metabolic rates are determined in a Mendelian genetic cross; while simplistic, it is a reasonable approximation. Thus, the child will receive either its mother’s or its father’s vision, and either its mother’s or its father’s metabolism.

The child’s inheritance from its parents consists solely of its genetic makeup and the amount of water it receives when it is born. There is no other inheritance mechanism for transfer of resources between generations currently implemented. The lack of an inheritance mechanism can be thought of as being equivalent to a 100% inheritance tax.

3.2.2 Cells

The cell is the basic unit of space in the XeriScape model. Each cell has four neighbors: north, south, east, and west (diagonals are not used in the XeriScape). This is called a *von Neumann* neighborhood (see Figure 7). The alternative *Moore* neighborhood is not used in XeriScape at this time.

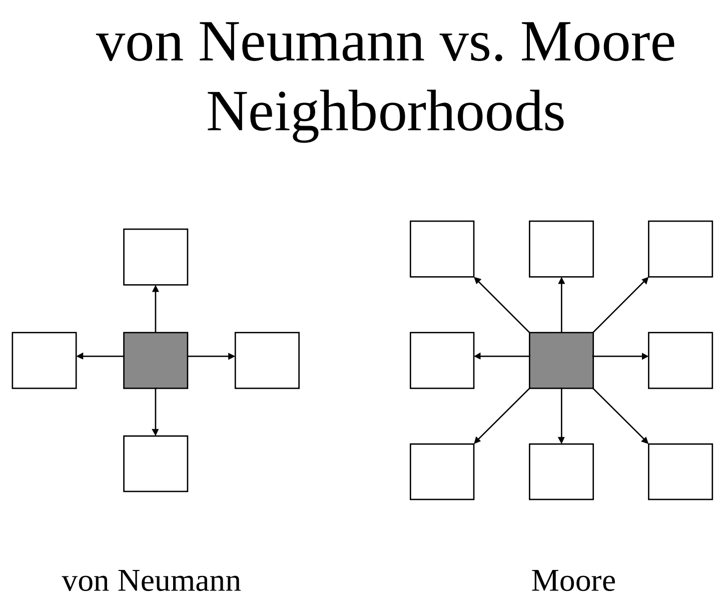
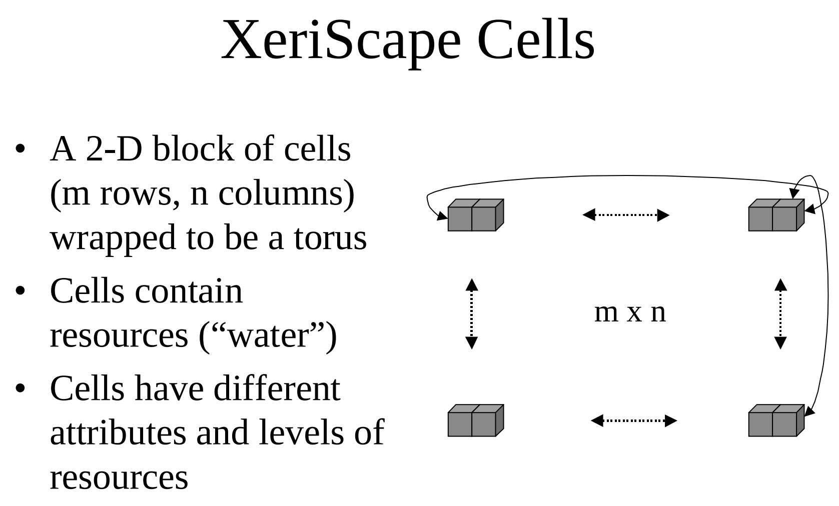


Figure 7. von Neumann vs. Moore Neighborhoods

Cells on the edge of the XeriScape wrap to the other side of the space, that is, the space is not really a flat, two-dimensional space but is instead a torus (doughnut shape) made up of two-dimensional elements (see Figure 8). Each cell is considered a point, that is, everything inside the cell is present to everything else.



XeriScape Cells

- A 2-D block of cells (m rows, n columns) wrapped to be a torus
- Cells contain resources ("water")
- Cells have different attributes and levels of resources

Figure 8. XeriScape Cells

Cells contain resources that are consumed and replenished over time. In the case of the XeriScape, the primary resource is water. Since the distribution of resources is not uniform over the XeriScape, some resource "peaks" and "valleys" exist. The rate at which resources are replenished is determined by a rule (see section 3.2.3).

Cells contain agents, and each cell can contain a pre-specified number of agents. In Sugarscape, cells could only contain 1 agent at a time, and so the XeriScape defaults to this behavior. XeriScape supports the ability to put an arbitrary number of agents in each cell, although this ability is not demonstrated in this thesis.

Cells can become polluted by the activities of the agents living in them. This pollution increases as agent gathering and consumption activity continues, but is dispersed gradually by a diffusion process. Note that pollution is dispersed, but never disappears from the environment, just as in the real world. This dispersal process averages the amount of pollution in a cell with the pollution level in its immediate neighbors.

The topology of the XeriScape simulation is built to resemble that of Sugarscape, with two “peaks” or concentrations (oases) of water, one at the top right and another at the bottom left of the simulation. See section 4.11 for an example and discussion of how this appears in the XeriScape graphical display.

3.2.3 Rules

Rules are the laws of the XeriScape. The variety of rules available and the ease with which new ones can be written is where the true power and flexibility of the XeriScape model originates. Rules control such elements as the initial distribution of agents in the XeriScape, the rate at which resources regenerate in a cell, the “genetic makeup” of each agent, the criteria used to compare possible destination cells, and much more. There are four basic rules for each simulation: a resource comparison rule, a resource initialization rule, an agent initialization rule, and an agent distribution rule.

3.2.3.1 Resource Comparison Rule

The Resource comparison rule (RCR) determines how an agent compares the resource values of a cell into which the agent is considering moving. In the simplest case, it involves a straight comparison of values, but in some cases, it is more complex. For example, the presence of pollution in a cell may make it a less desirable destination than another cell with less water but no pollution.

3.2.3.2 Resource Initialization Rule

The resource initialization rule (RIR) creates the basic landscape of the simulation. It determines the types of resources, the resource capacity and initial resource allocation for each cell. The RIR uses a sub-rule to determine the manner or rate at which resources are replenished in each cell. This is termed the growback rate. The growback rate is expressed in terms of units of resource per unit time. Note that it is also permissible to have an infinite growback rate, i.e., resources grow back to their full capacity immediately.

3.2.3.3 Agent Initialization Rule

The agent initialization rule (AIR) is responsible for creating new agents. The simulation calls the AIR when the simulation starts in order to generate the initial complement of agents. The AIR is also used to generate replacement agents when new agents are injected into the simulation, for example by sexual reproduction, or when replacement mode is on (see section 4.13).

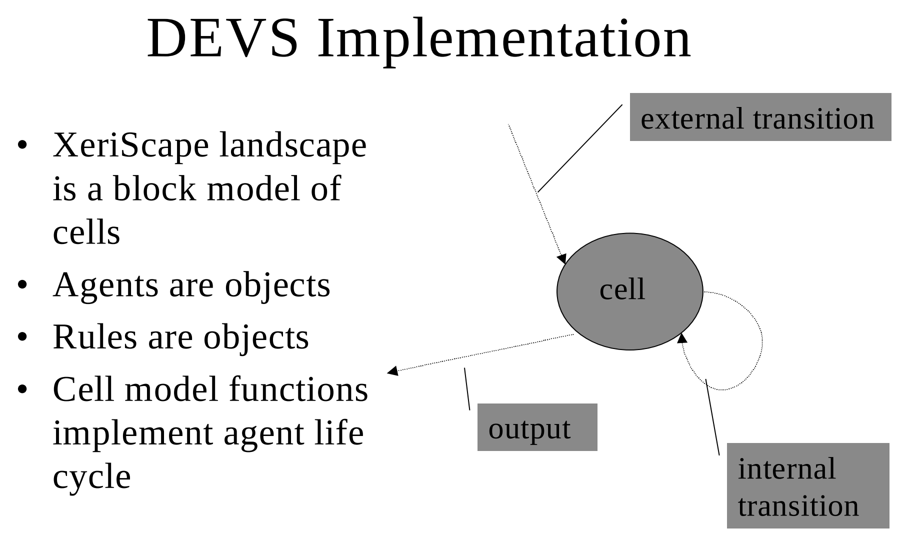
3.2.3.4 Agent Distribution Rule

The agent distribution rule (ADR) determines where newly-created agents are injected into the simulation. This distribution may be random, clustered, or organized in some fashion. Only a random distribution is used in this thesis.

4 THE XERISCAPE SYSTEM

4.1 Overview

The XeriScape system is implemented in the Java language, and it utilizes DEVS-Java 1.5 with some modifications and bug fixes.[2] Some parts of the XeriScape system are DEVS models (the block, the cell, and the transducer), while other parts are Java objects (agents, rules, and resources) (see Figure 9). In addition, a graphical user interface (GUI) allows the user to manipulate the models and extract data. The XeriScape system has two threads of execution: one for the simulation proper, and one for the graphical user interface.



DEVS Implementation

- XeriScape landscape is a block model of cells
- Agents are objects
- Rules are objects
- Cell model functions implement agent life cycle

Figure 9. DEVS Implementation

[2] DEVS-Java 1.5 is no longer available from the University of Arizona and is somewhat outdated. It may be made available along with the XeriScape code pending clearance from the University. See Appendix A for more information on obtaining XeriScape or DEVS-Java code.

4.2 Experimental Frame

The experimental frame for a XeriScape simulation run (Figure 10) is composed of five elements. These elements are the four rules that control the configuration as it runs, and the configuration file that controls the initial setup.

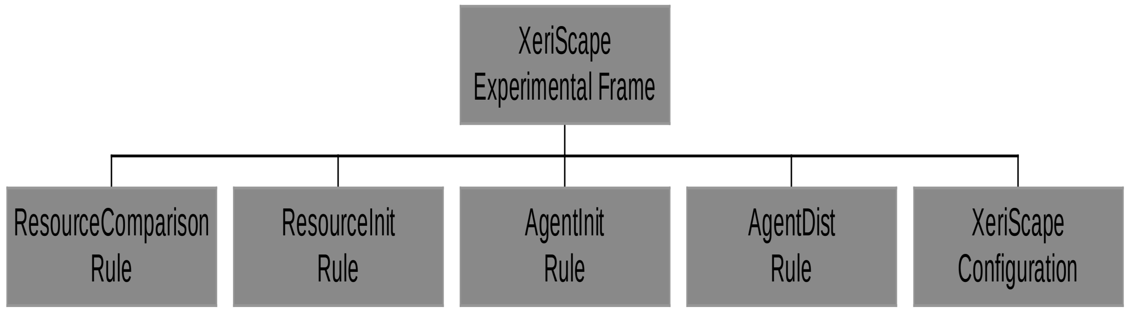


Figure 10. XeriScape Experimental Frame

4.3 Block Model

The XeriScape system uses a DEVS block model (`XSBlock`) to contain the individual cells. This block, a two-dimensional array of cells, is very simple and serves only to initialize the cells in the appropriate configuration and pass messages to/from the block. The size of the block is determined by the XeriScape configuration file (see Appendix B) at runtime. A 50×50 block size (2500 cells, the size used by Sugarscape) is used for all the simulation runs presented in Chapter 5. Other sizes can be selected, and the XeriScape system will scale itself accordingly.

4.4 Cells

The individual cells (each an instance of `XSCell`) are the heart of the XeriScape's DEVS model. Each cell extends the DEVS class `cell`. The cells implement the full set of DEVS transition functions (internal transition, external transition, output) and pass messages among themselves via a number of input and output ports (see Figure 11).

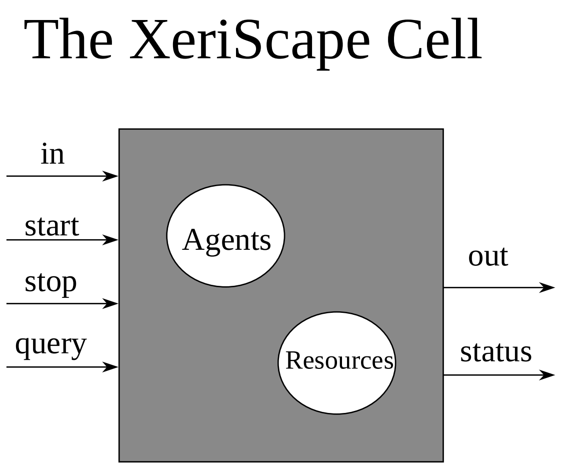


Figure 11. The XeriScape Cell

Because agents are not DEVS models, the cell itself implements the agent life cycle. While this may at first seem odd, the result is the ability to support a broad variety of agent types and agent behaviors with a unified cell model. Furthermore, because the model is static in structure (agents move, not cells) the result is a conceptually simple architecture that is straightforward and easy to visualize.

Since the cell has to support all the known types of agents, it must be able to determine what type(s) of agent(s) it contains; Java's reflection feature is used to determine this information while the simulation is running. For example, the cell determines if the agent is gendered, and if it is, the cell calls the agent's `reproduce()` function. Rather than sub-classing the cell model to match the different types of supported agents (i.e., having a cell class hierarchy that matches the agent hierarchy), XeriScape has one cell model that can support all of the agent types. This gives XeriScape the ability to mix different agent types in the same running simulation.

Figure 12 shows the cell state transitions. XeriScape cells have only two states: passive and busy. Cells enter the busy state whenever an external transition ("deltext") occurs; this corresponds to an agent entering the cell. The cell stays in the busy state as long as agents are present and/or its resources are below their capacity; the internal transition ("deltint") is used to implement the agent life cycle. The agent leaves the cell via the cell's output function ("out").

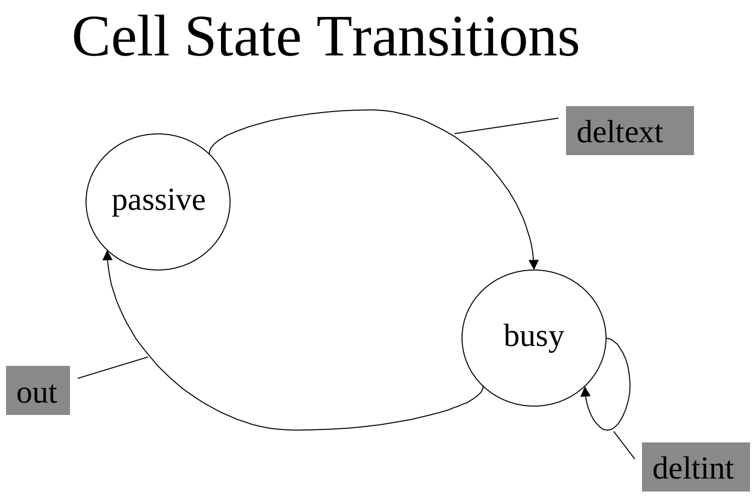


Figure 12. Cell State Transitions

The cell's `out()` function is used to send agents to new cells and to send status messages to the transducer.

4.5 Agents

The XeriScape agents (`Agent, FiniteAgent, PollutingAgent,` and `GenderedAgent`) are Java objects that extend the DEVS class `entity`. A number of different agents can be implemented to support different experiments. Java inheritance is used to make development of new types of agents simple and straightforward. Figure 13 shows the Java class relationships between the various agent types in XeriScape.

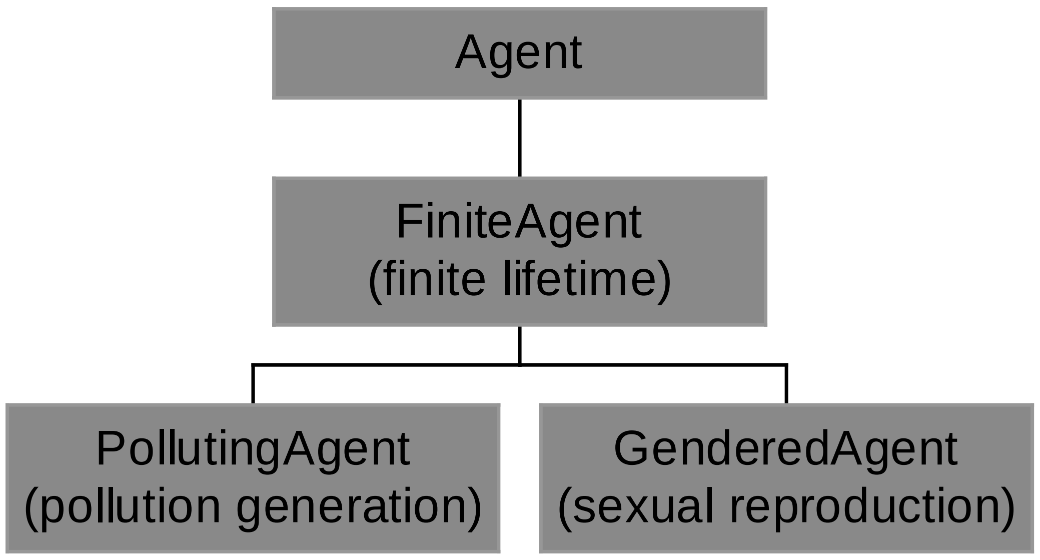
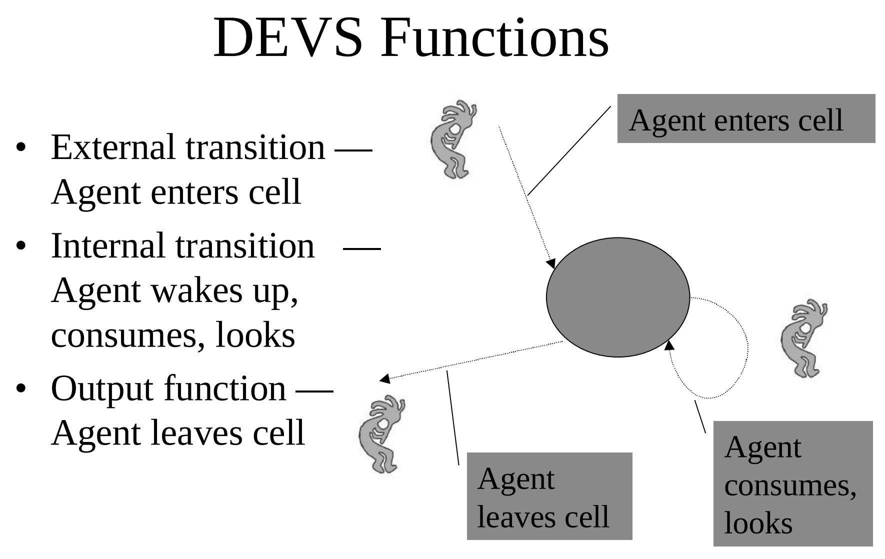

Agent
FiniteAgent (finite lifetime)
PollutingAgent (pollution generation)
GenderedAgent (sexual reproduction)

Figure 13. XeriScape Agent Class Hierarchy



DEVS Functions

- External transition — Agent enters cell
- Internal transition — Agent wakes up, consumes, looks
- Output function — Agent leaves cell

Agent enters cell
Agent leaves cell
Agent consumes, looks

Figure 14. DEVS Functions

As mentioned in section 4.4, the XeriScape cell implements the agent life cycle. Figure 14 above shows how the standard DEVS state transition functions correspond to the agent life cycle.

4.6 Rules

XeriScape rules are Java objects that implement decision-making capability for the simulation. They are created when the simulation is initialized. Section 3.2.3 discussed the various types of rules.

4.7 Resources

XeriScape resources are Java objects that represent various natural resources or items of interest to agents and cells. The most basic resource for XeriScape is water, however, other resources can be created and managed as well. For example, for the pollution experiment described in section 5.5, a new resource ("TCE", tricholoroethylene, an insidious pollutant) is created and used by the simulation.

4.8 Agent Slots

In order to regulate agent movement so that the agent capacity of any cell is not exceeded (i.e., so that no more agents are present in any cell than it allows), XeriScape includes a reservation system.. Whenever an agent is considering a move, it determines whether a potential destination has a free slot. If the agent decides to move to that cell, it reserves the slot so that another agent cannot take its place.

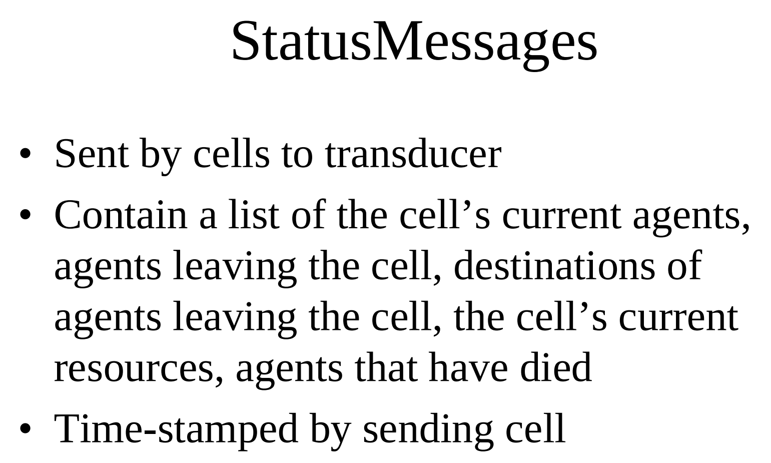
This implementation detail is transparent to the user. However, it is of interest because it is one area that would need extensive modification to make the XeriScape system capable of executing in multiple threads, such as in a distributed computing environment. One possibility would be to move the slot reservation management function from the transducer, where is currently resides, to the individual cells. Slot information could then be transmitted in the messages transmitted from cell to cell (see section 4.9.2).

4.9 Messages

XeriScape uses two different types of Java objects to convey data within the simulation: status messages and cell queries.

4.9.1 StatusMessage

`StatusMessage` objects (see Figure 15) are sent from each cell's output function and received by the transducer. These objects report the address of the generating cell, the live agents in the cell, the resources available in the cell, agents that are leaving the cell, agents that have died, and a time stamp.



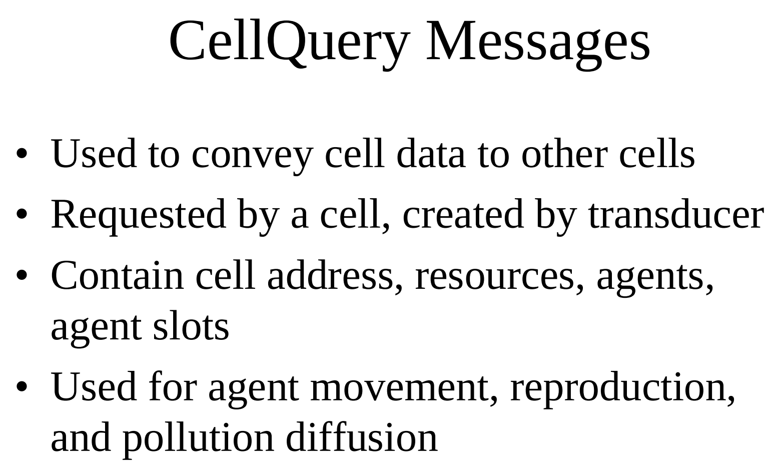
StatusMessages

- Sent by cells to transducer
- Contain a list of the cell’s current agents, agents leaving the cell, destinations of agents leaving the cell, the cell’s current resources, agents that have died
- Time-stamped by sending cell

Figure 15. StatusMessages

4.9.2 CellQuery

`CellQuery` objects are created upon request by the transducer. They report the number of agent slots and the resources available for a specified cell. These objects are used to determine an agent’s destination when moving; they are also used in the reproductive process by gendered agents and for diffusion of pollution (see Figure 16).



CellQuery Messages

- Used to convey cell data to other cells
- Requested by a cell, created by transducer
- Contain cell address, resources, agents, agent slots
- Used for agent movement, reproduction, and pollution diffusion

Figure 16. CellQuery Messages

4.10 Transducer and Data Extraction

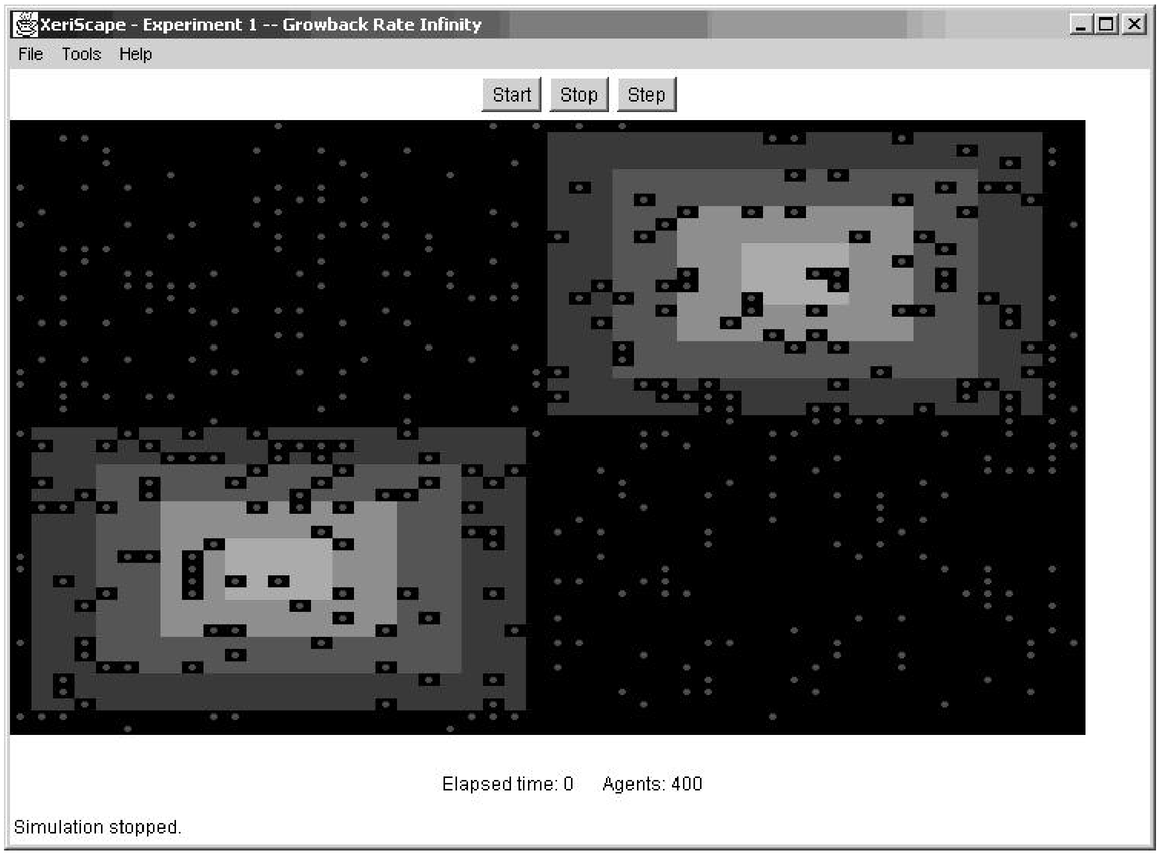
In order to allow the user to analyze the simulation's workings and record data as the simulation runs, XeriScape includes a transducer and a data logging system.

The transducer (`XSTransd`) is implemented as a DEVS-Java model. The transducer’s work is primarily done in its external transition function, which processes incoming status messages and does extensive reporting and data analysis. The transducer’s output function is used to inject new agents into the simulation when the replacement mode is on (see section 4.13).

XeriScape includes a logging function to extract data as the simulation runs. An instance of the `Logger` class is used to log events to the Java console, a data file, or both. A verbosity threshold (QUIET, NORMAL (default), or VERBOSE) can be set, and individual log entries are also prioritized by verbosity level. Only log entries with a verbosity level greater than or equal to the threshold are logged.

4.11 Visualization

XeriScape implements a visualization system that provides a display similar to Sugarscape, as shown in Figure 17 below.



Elapsed time: 0 Agents: 400

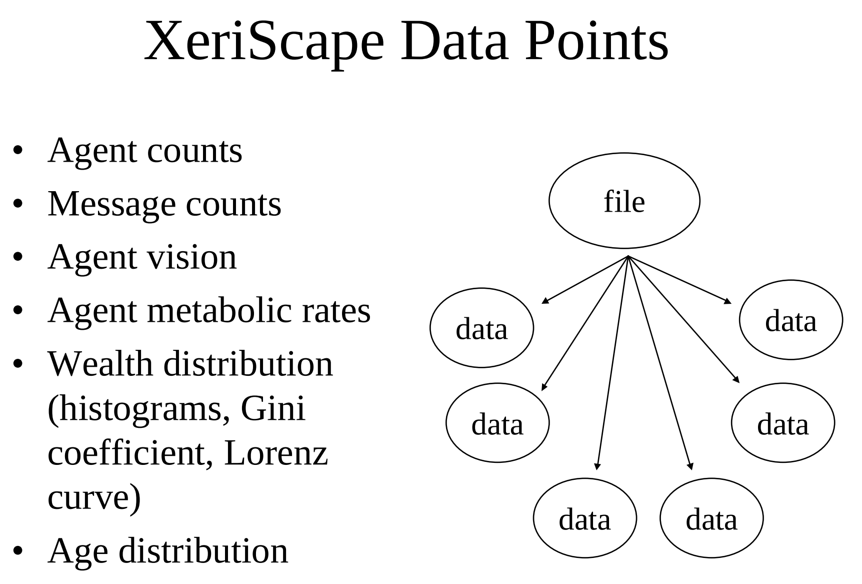
Simulation stopped.

Figure 17. XeriScape Graphical User Interface

The XeriScape GUI displays the simulation results and provides basic controls to start, stop, and step the simulation; load, create, and save configuration files; and change the time step size. In Figure 17 the dots represent agents. The background of each cell is colored to represent its water resource level: a black background indicates no water, a lighter shade of yellow a low water level, and the brightest shade of yellow represents the highest water level. The panel below the visual display indicates the elapsed simulation time and the number of live agents currently in the simulation. The status bar at the lower left shows whether the simulation is running, stepping, or stopped. The title bar at the top of the display indicates the name of the configuration being run (in Figure 17, "XeriScape - Experiment 1 -- Growback Rate Infinity").

4.12 Data Reduction

To reduce the data produced by the logger to a meaningful format, a simple data reduction tool called `ProcessLog` has been written to support XeriScape. This tool reads in a data file produced by the logger and processes it into comma-delimited data files for several data points (see Figure 18). The reduced data is considerably more compact than the raw data files and only includes data that is always extracted regardless of the verbosity level requested.



XeriScape Data Points

- Agent counts
- Message counts
- Agent vision
- Agent metabolic rates
- Wealth distribution (histograms, Gini coefficient, Lorenz curve)
- Age distribution

file
data
data
data
data
data
data
data

Figure 18. XeriScape Data Points

4.13 Replacement Mode

As mentioned in section 4.10, the XeriScape simulation has a replacement mode that can be turned on or off. When the replacement mode is on, agents that die are replaced by new agents inserted into the simulation in a random position. These new agents are generated using the same agent initialization rule that was used to generate the initial complement of agents. This mode is demonstrated in Experiment 3 (section 5.3).

5 EXPERIMENTS AND RESULTS

The experiments detailed below illustrate the operation of the XeriScape system in a number of different ways. Experiments 1 through 6 show that the XeriScape system can generate results similar to those of Sugarscape. These experiments are based on simulation runs described in Epstein and Axtell.

Experiments 7 and 8 show that the XeriScape system's flexibility and extensibility allow it to easily do things Sugarscape cannot. These experiments show the enhanced capabilities that implementation in the DEVS-Java environment provides for artificial society simulation.

For each experiment, a brief description of the experiment setup is given, followed by a diagram of the experimental frame for the experiment and sample GUI screen captures of the simulation run (see section 4.11 for an explanation of the GUI visualization). Finally, a discussion of the data produced by the experiment and the conclusions that may be drawn is provided.

5.1 Experiment 1 — Basic Agents with Instantaneous Growback

5.1.1 Setup

The first experiment uses the basic agent class, a random spatial distribution of agents in the environment, and a resource initialization rule that allows resources to be replenished instantaneously. The initial number of agents is 400.

5.1.2 Experimental Frame

The experimental frame for Experiment 1 is shown in Figure 19 below.

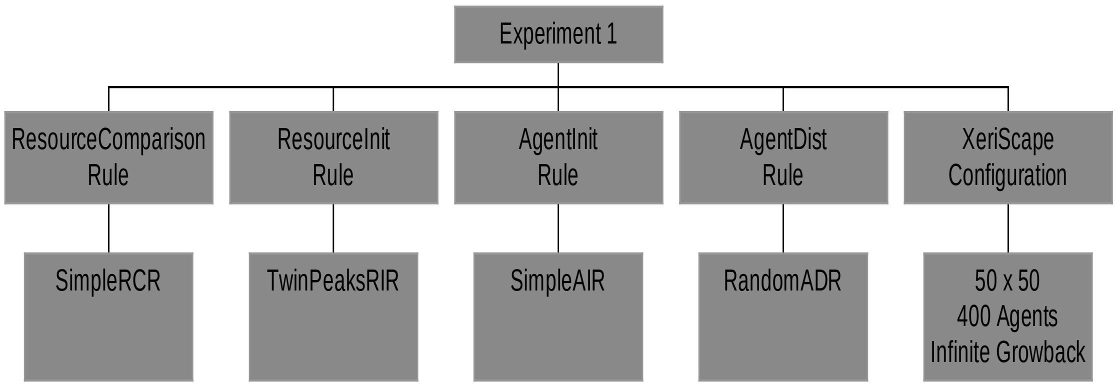
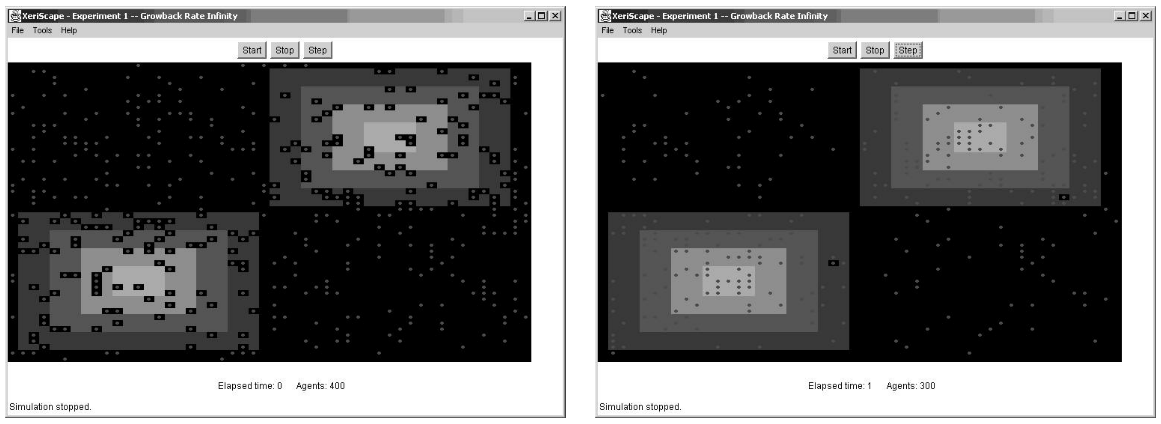


Figure 19. Experiment 1 Frame

5.1.3 Screen Captures

Figure 20 and Figure 21 show screen captures from the Experiment 1 simulation run.


XeriScape - Experiment 1 -- Growback Rate Infinity
File Tools Help
Start Stop Step
XeriScape - Experiment 1 -- Growback Rate Infinity
File Tools Help
Start Stop Step

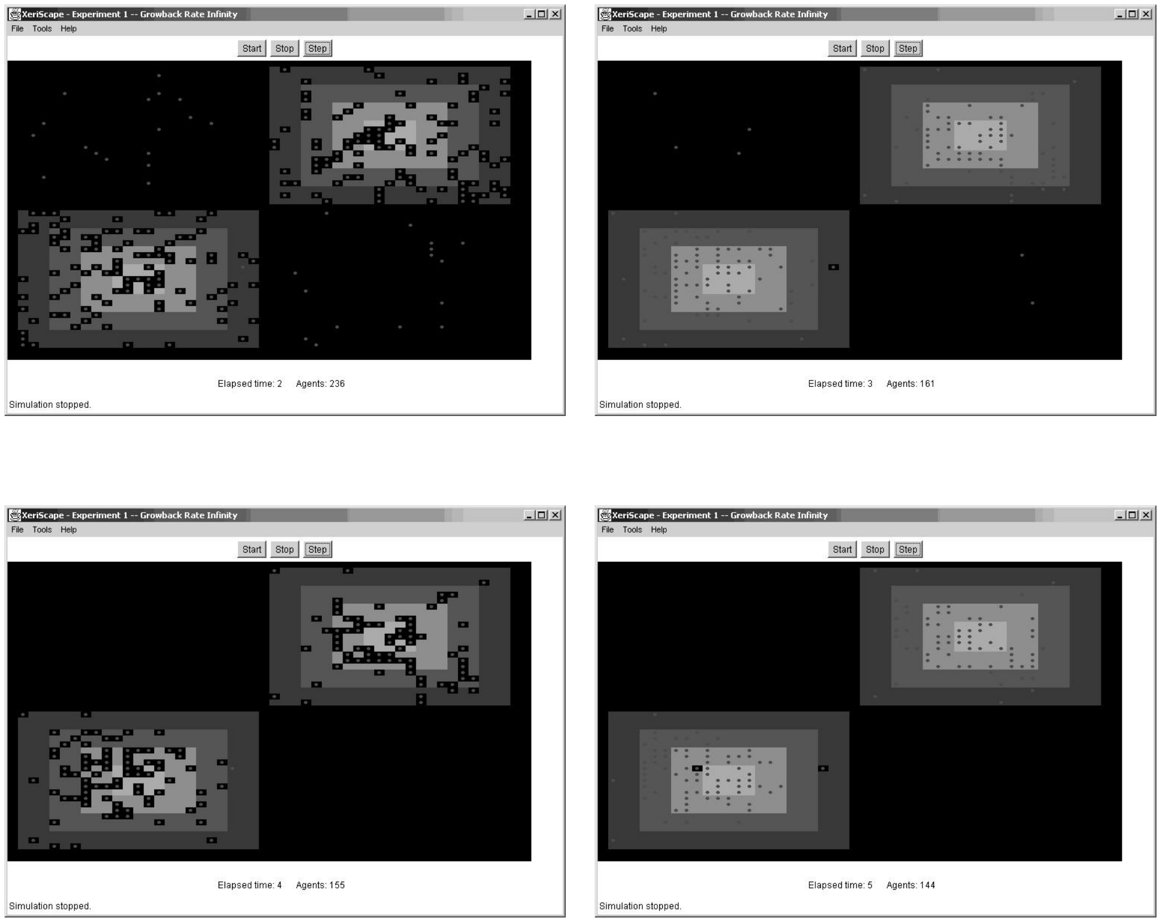
Elapsed time: 0 Agents: 400

Simulation stopped.

Elapsed time: 1 Agents: 300

Simulation stopped.

Figure 20. Experiment 1 Screen Captures



Elapsed time: 2 Agents: 236
Simulation stopped.

Elapsed time: 3 Agents: 161
Simulation stopped.

Elapsed time: 4 Agents: 155
Simulation stopped.

Elapsed time: 5 Agents: 144
Simulation stopped.

Figure 21. Experiment 1 Screen Captures (continued)

5.1.4 Discussion

Looking at Figure 20 and Figure 21, the following story can be told: a random distribution of agents is born on the XeriScape. Those close to the water oases migrate to the areas of greatest water availability. Those born in the desert die from thirst more or less rapidly depending upon their individual metabolic rates and how much water they had initially.

Particularly interesting to note in this experiment, and throughout the XeriScape simulation runs, is the effect of the von Neumann neighborhood on the behavior of the agents. Since the agents cannot see diagonally, they tend to find themselves on water resource "tiers," each of which has an equal level of resources. There may be a richer source of water on the diagonal, but they cannot see it and thus never move there.

Experiment 1 illustrates the concept of carrying capacity, that is, the population that the environment can support over time. Figure 22 shows a graph of agent population over time. Clearly, the Experiment 1 simulation has a carrying capacity of roughly 100 agents.

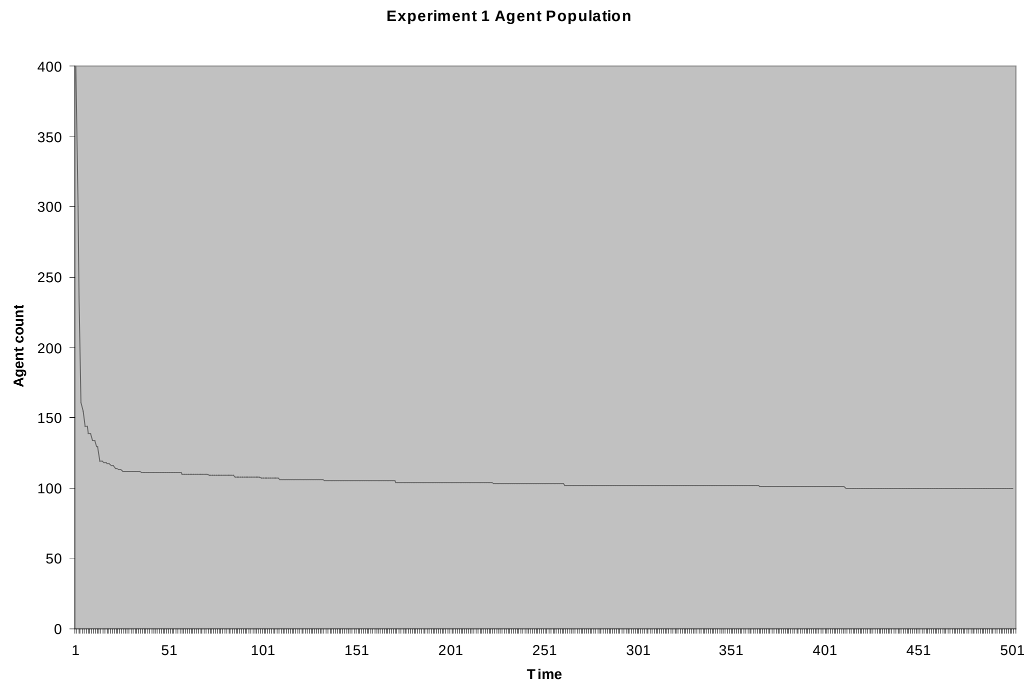


Figure 22. Experiment 1 Agent Population

5.2 Experiment 2 — Basic Agents with Timed Growback

5.2.1 Setup

Experiment 2 is identical to Experiment 1, except that instead of resources growing back instantaneously (i.e., an infinite growback rate), they grow back at a rate of 1 unit per time period.

5.2.2 Experimental Frame

The experimental frame for Experiment 2 is shown in Figure 23 below.

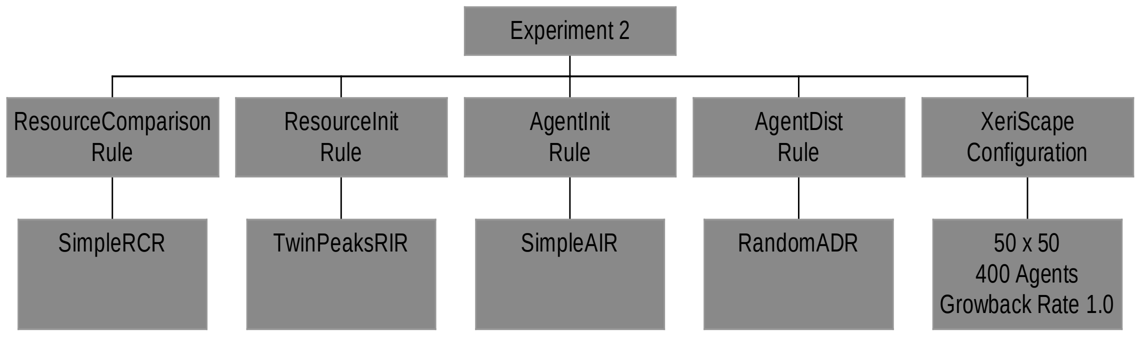
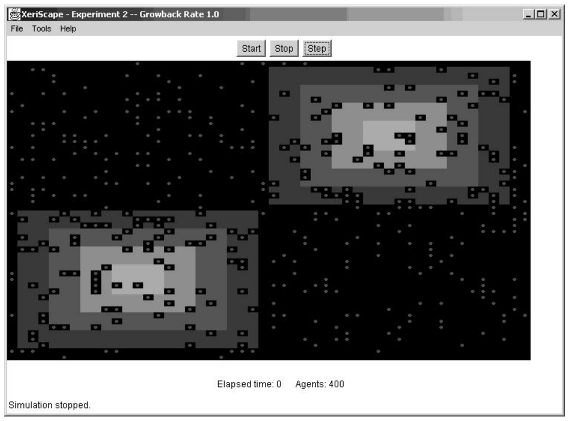


Figure 23. Experiment 2 Frame

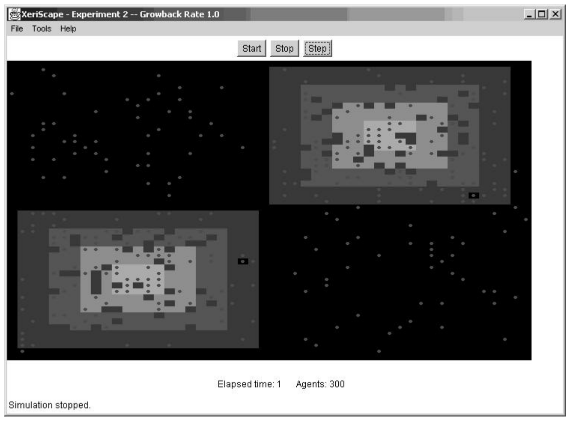
5.2.3 Screen Captures

Figure 24 shows screen captures of the Experiment 2 simulation run.



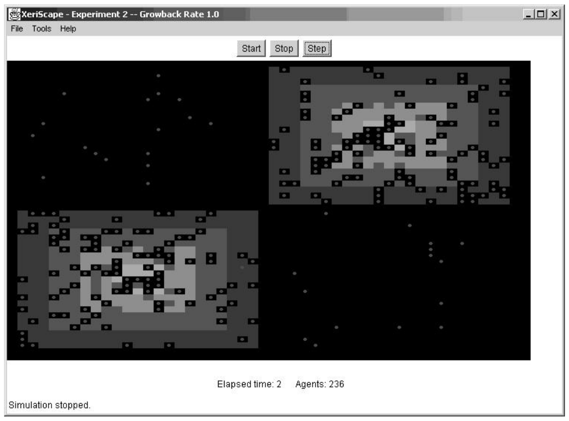
Elapsed time: 0 Agents: 400

Simulation stopped.



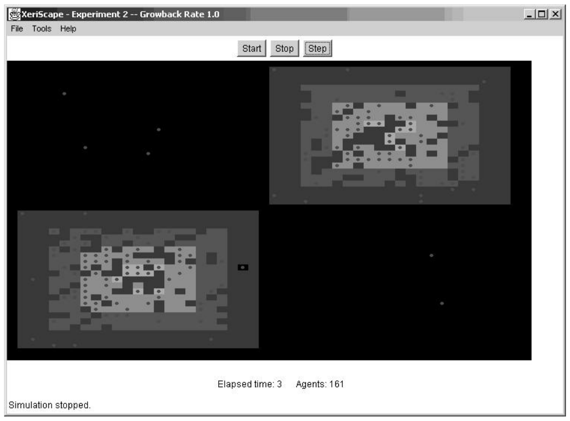
Elapsed time: 1 Agents: 300

Simulation stopped.



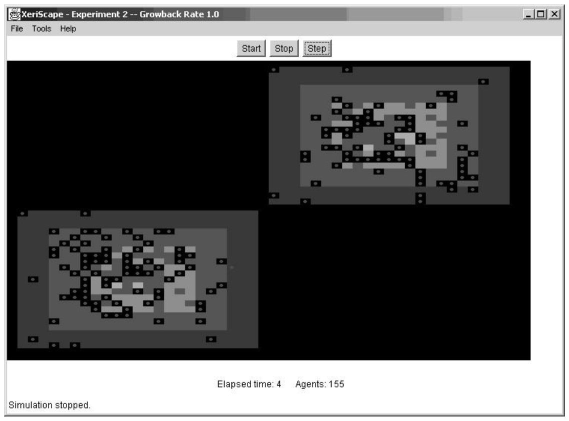
Elapsed time: 2 Agents: 236

Simulation stopped.



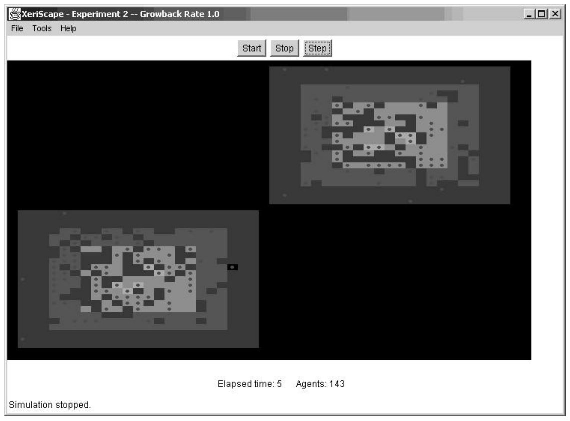
Elapsed time: 3 Agents: 161

Simulation stopped.



Elapsed time: 4 Agents: 155

Simulation stopped.



Elapsed time: 5 Agents: 143

Simulation stopped.

Figure 24. Experiment 2 Screen Captures

5.2.4 Discussion

Experiment 2 builds on Experiment 1. In Experiment 2 we examine the principle of *selection*. While selection is usually thought of in terms of sexual reproduction, selection also occurs whenever agents are forced to respond to their environment or die. The Darwinian concept of “survival of the fittest” plays out in Experiment 2.

Reason tells us that those agents with the lowest metabolisms should stand the best chance of survival, all other things being equal, because they can make the most of their resources. Reason also tells us that those agents with the keenest sight should have a greater chance of survival than those with shorter vision. Both of these insights are borne out in the Experiment 2 results. In Figure 25 we see that mean agent metabolism decreases rapidly at the start of the simulation, then plateaus at a level of approximately 1.68. Similarly, Figure 26 shows that mean agent vision increases rapidly, then plateaus at a level of roughly 4.08.

It is important at this point to recall that agent vision and agent mobility are the same metric, that is, agents can move as far as they can see. Thus not only does the society select for higher vision but also for agents which can travel farther and thus gather more resources.

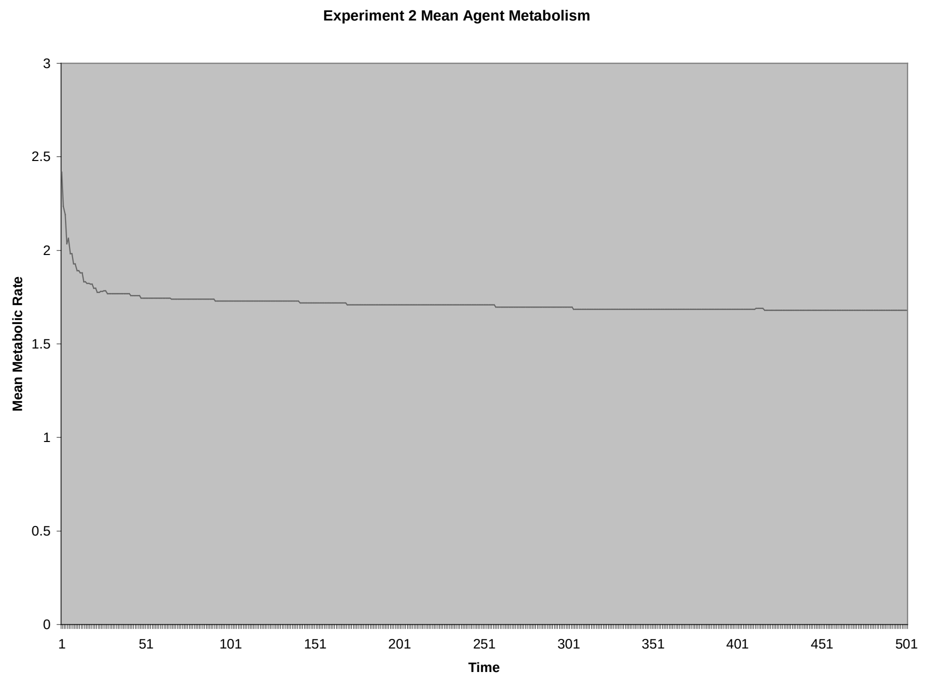


Figure 25. Experiment 2 Mean Agent Metabolism

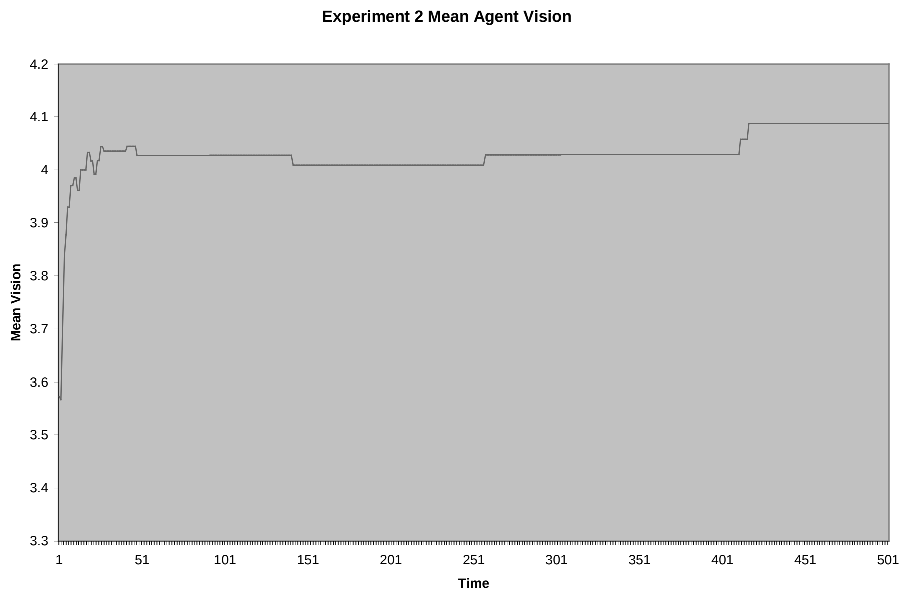


Figure 26. Experiment 2 Mean Agent Vision

5.3 Experiment 3 — Finite Lifetimes

5.3.1 Setup

In this experiment agents have a finite lifetime. This lifetime is established at the agents' creation and is random in a predetermined range (in this case, it varies from 60 to 100). When the agent dies (whether by starvation or old age), it is replaced by a new agent of age 0 that is injected into the simulation at a random location. This replacement is called replacement mode (see section 4.13). The number of agents in this experiment was lowered to 125 to approximate the carrying capacity of the XeriScape.

5.3.2 Experimental Frame

The experimental frame for Experiment 3 is shown in Figure 27 below.

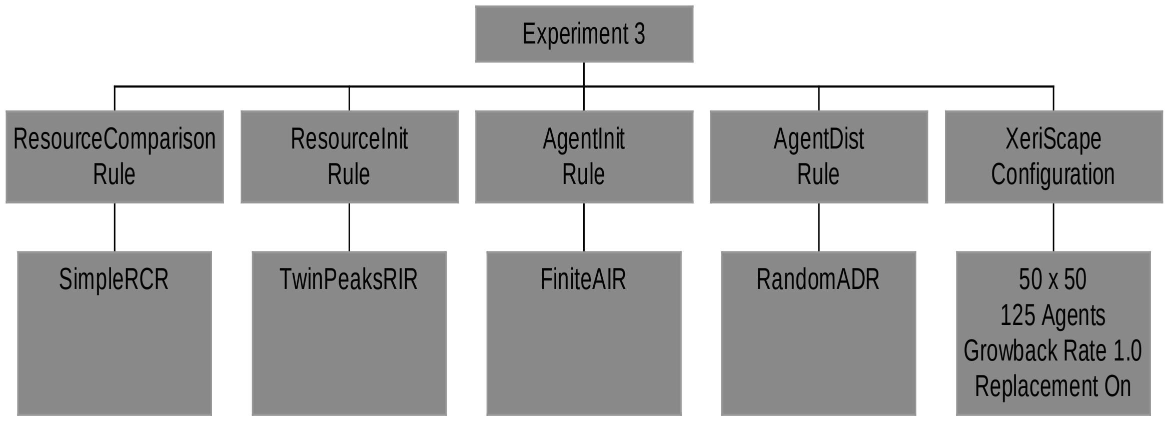
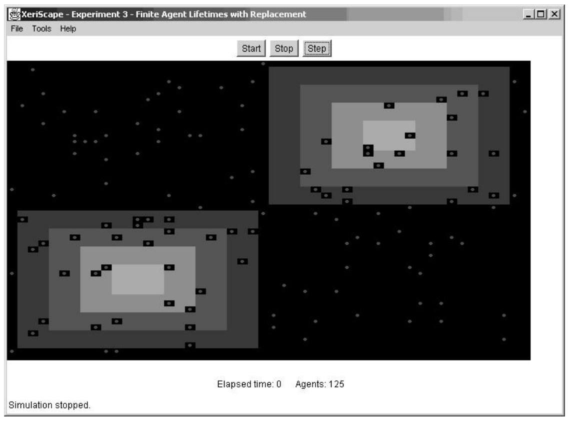


Figure 27. Experiment 3 Frame

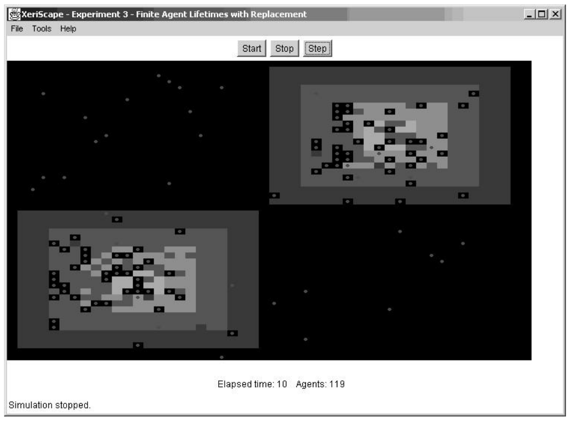
5.3.3 Screen Captures

Figure 28 shows the screen captures from Experiment 3.


XeriScape - Experiment 3 - Finite Agent Lifetimes with Replacement
File
Tools
Help
Start
Stop
Step

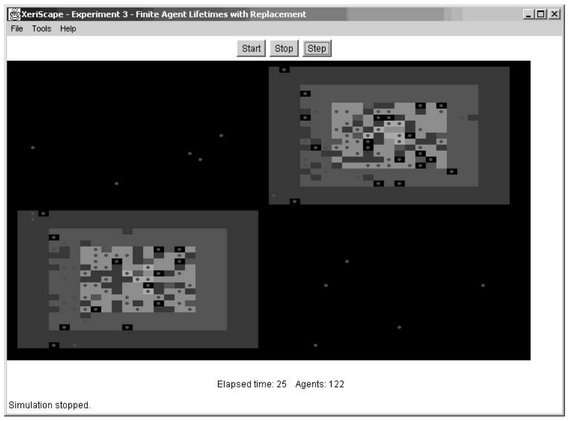
Elapsed time: 0 Agents: 125

Simulation stopped.


XeriScape - Experiment 3 - Finite Agent Lifetimes with Replacement
File
Tools
Help
Start
Stop
Step

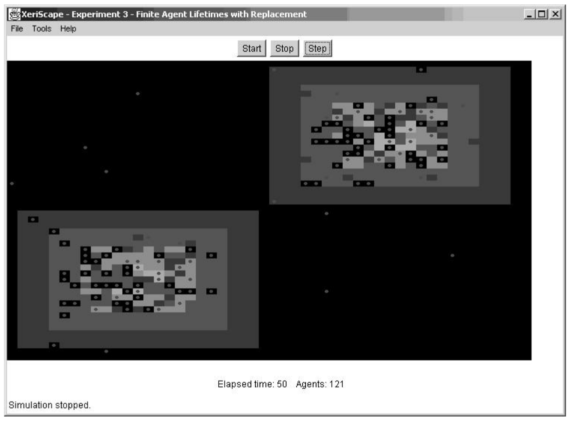
Elapsed time: 10 Agents: 119

Simulation stopped.


XeriScape - Experiment 3 - Finite Agent Lifetimes with Replacement
File
Tools
Help
Start
Stop
Step

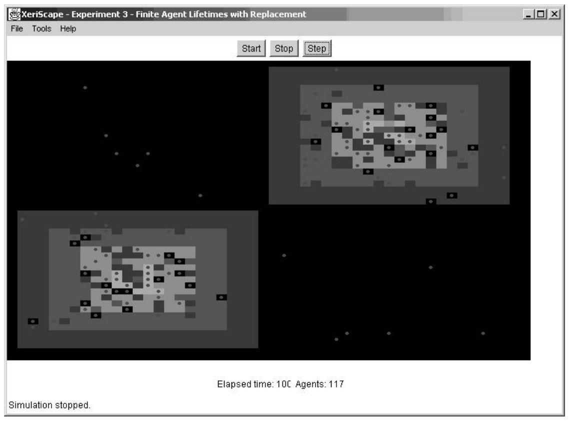
Elapsed time: 25 Agents: 122

Simulation stopped.


XeriScape - Experiment 3 - Finite Agent Lifetimes with Replacement
File
Tools
Help
Start
Stop
Step

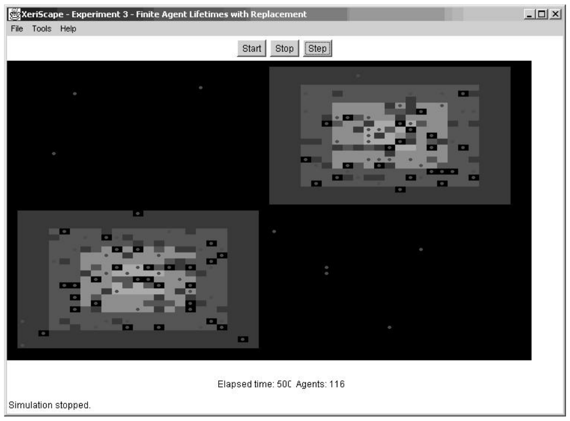
Elapsed time: 50 Agents: 121

Simulation stopped.


XeriScape - Experiment 3 - Finite Agent Lifetimes with Replacement
File
Tools
Help
Start
Stop
Step

Elapsed time: 10(Agents: 117

Simulation stopped.


XeriScape - Experiment 3 - Finite Agent Lifetimes with Replacement
File
Tools
Help
Start
Stop
Step

Elapsed time: 50(Agents: 116

Simulation stopped.

Figure 28. Experiment 3 Screen Captures

5.3.4 Discussion

Experiment 3 provides a platform for examining water resource (wealth) distributions and how they change over time. Figure 29, Figure 30, Figure 31, and Figure 32 show histograms of wealth distribution in the XeriScape at times 0, 5, 10, 25, 100, and 500. In these histograms, the vertical bars represent the number of agents for each decile of the total wealth of the simulation. In general we can see from the histograms that as time goes on agent wealth becomes more concentrated. The top decile of the societal wealth is held by fewer and fewer individuals. This is due to the fact that longer-lived agents which are born in resource-rich areas of the XeriScape can accumulate relatively great wealth, especially if they have a low metabolism and a high vision.

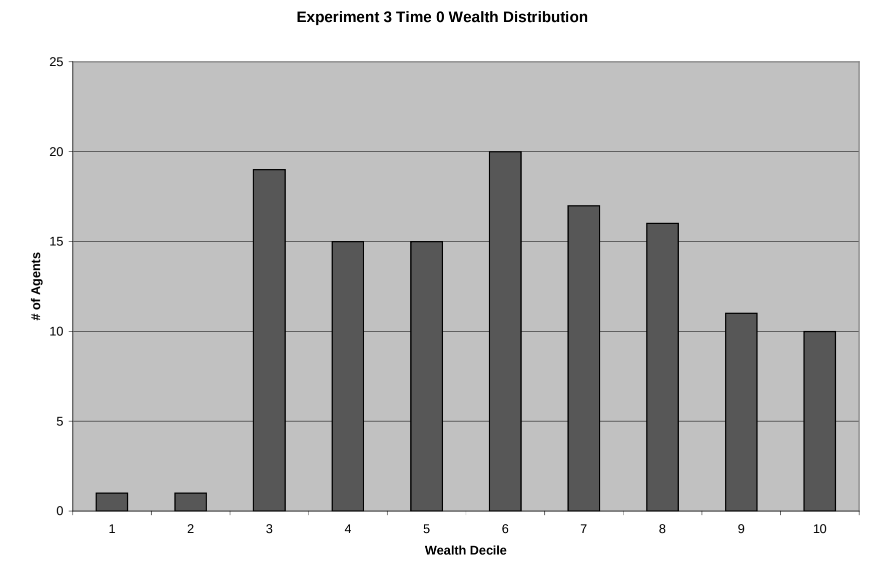
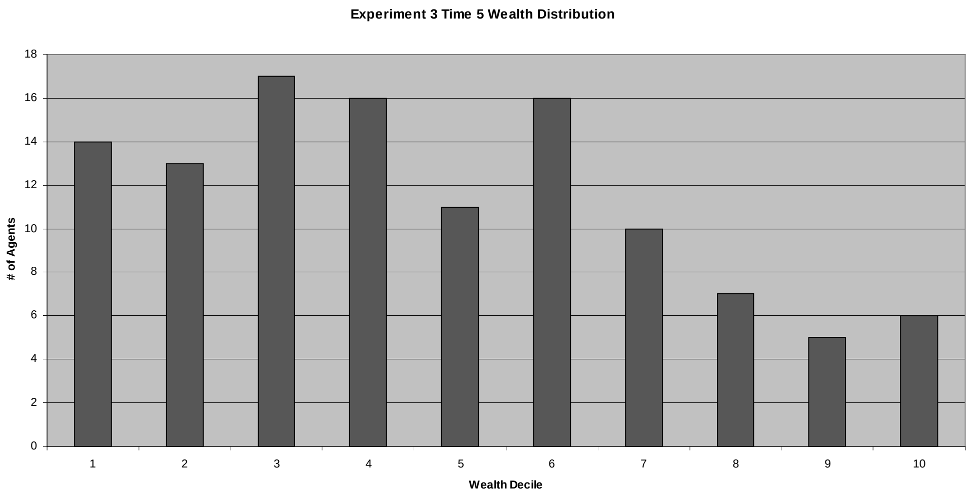


Figure 29. Experiment 3 Wealth Histograms



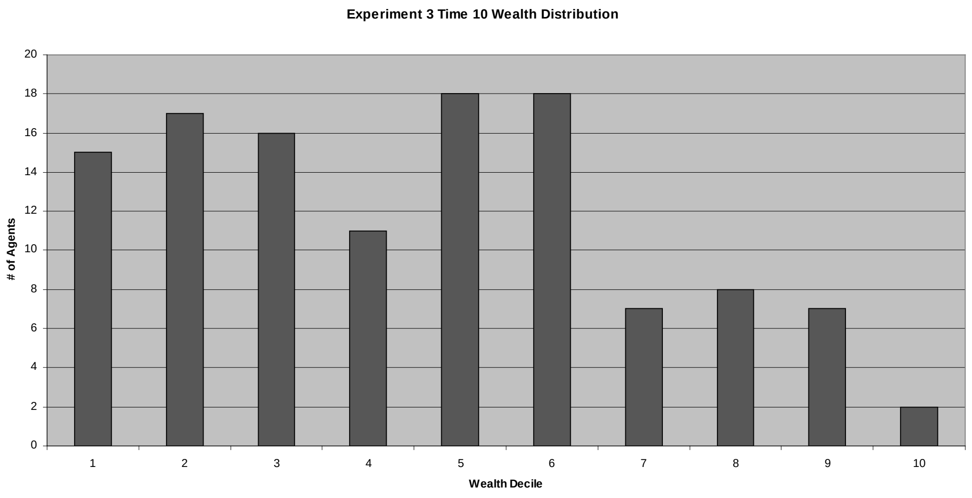
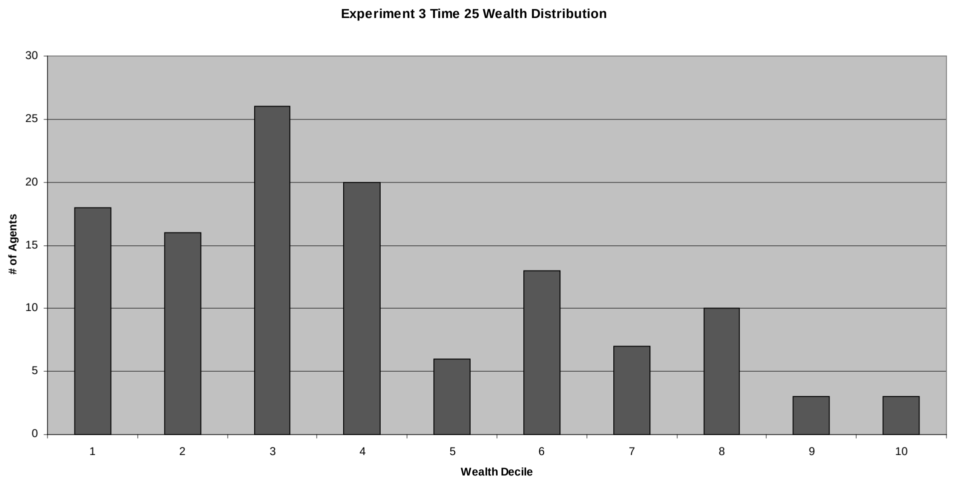


Figure 30. Experiment 3 Wealth Histograms (continued)



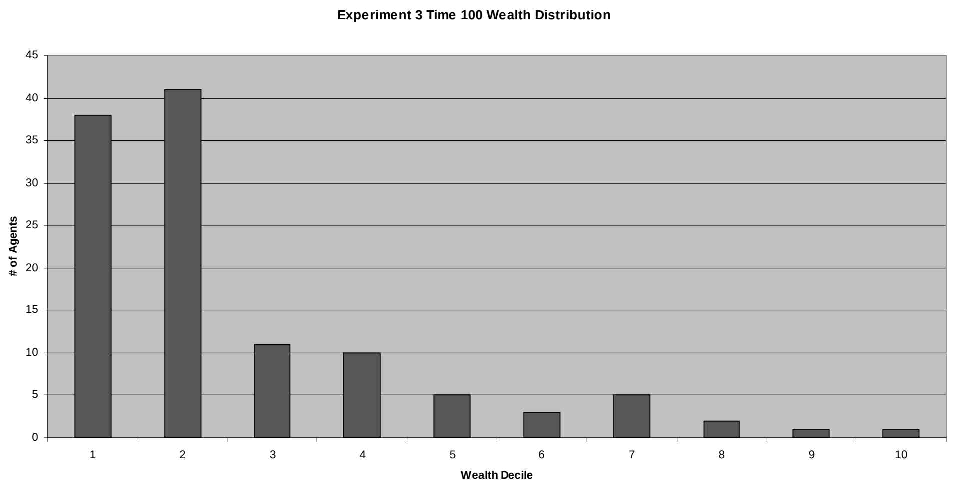


Figure 31. Experiment 3 Wealth Histograms (continued)

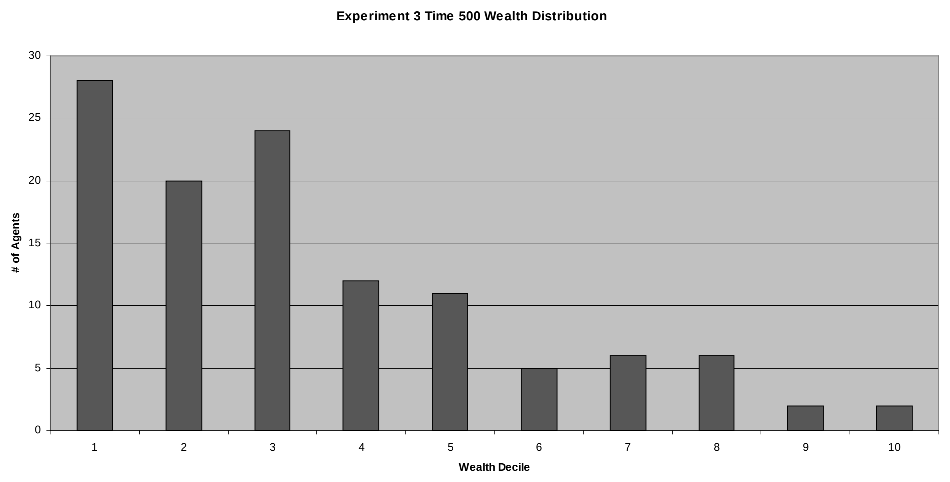
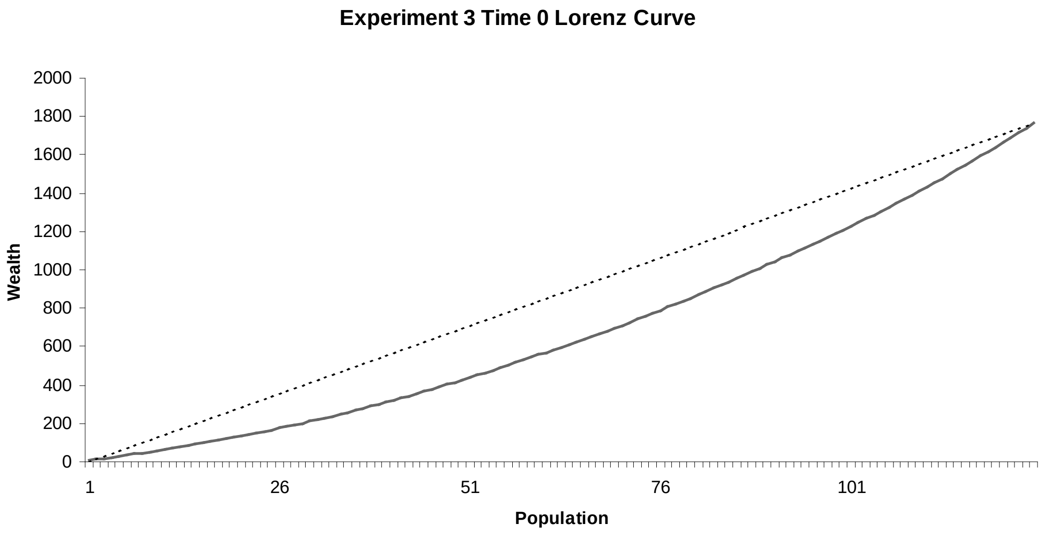


Figure 32. Experiment 3 Wealth Histograms (continued)

Also instructive are the Lorenz curves for the same sample times (see Figure 33, Figure 34, and Figure 35). The Lorenz curve [22] is one measure of wealth distribution in a society; a perfectly equitable wealth distribution would be a straight line at a 45° angle. The more the curve falls short of the 45° line, the more unequal the distribution. The horizontal axis represents the agent population, ordered from poorest to wealthiest. The points on the curve are the amount of wealth of that agent summed with the agents that are poorer, that is, the fraction of wealth owned at that fraction of the population.



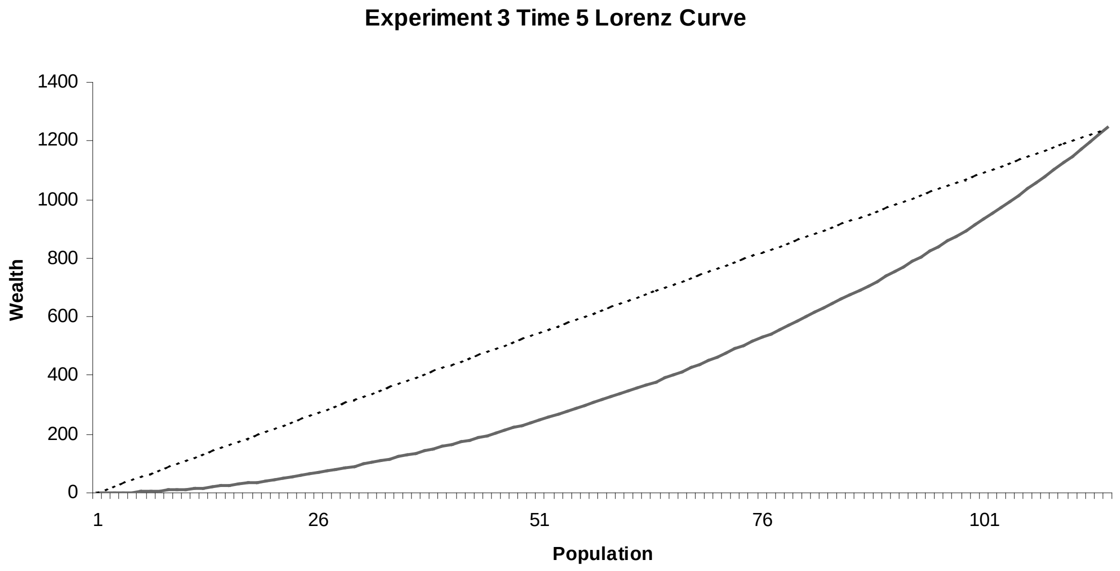
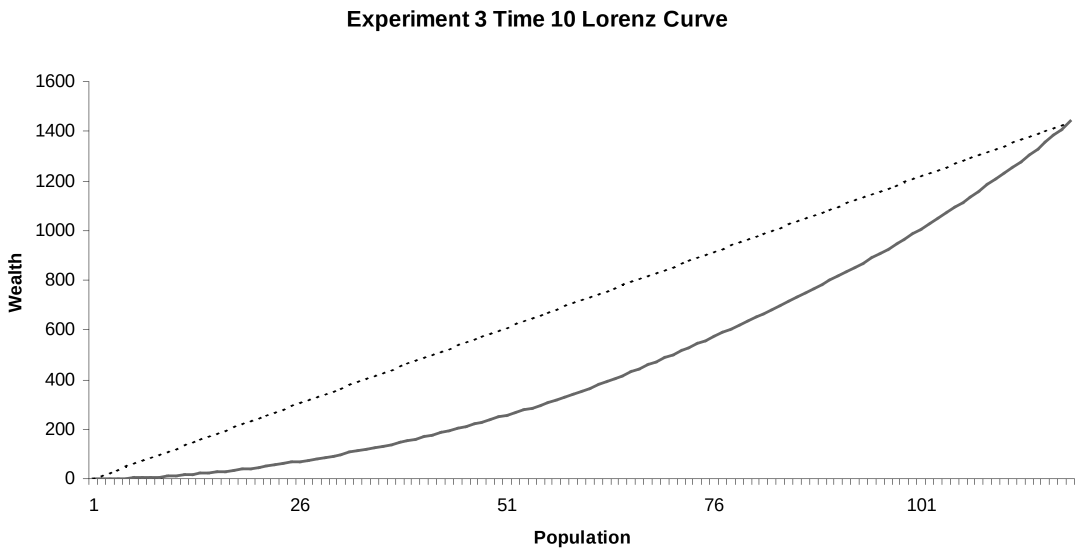


Figure 33. Experiment 3 Lorenz Curves



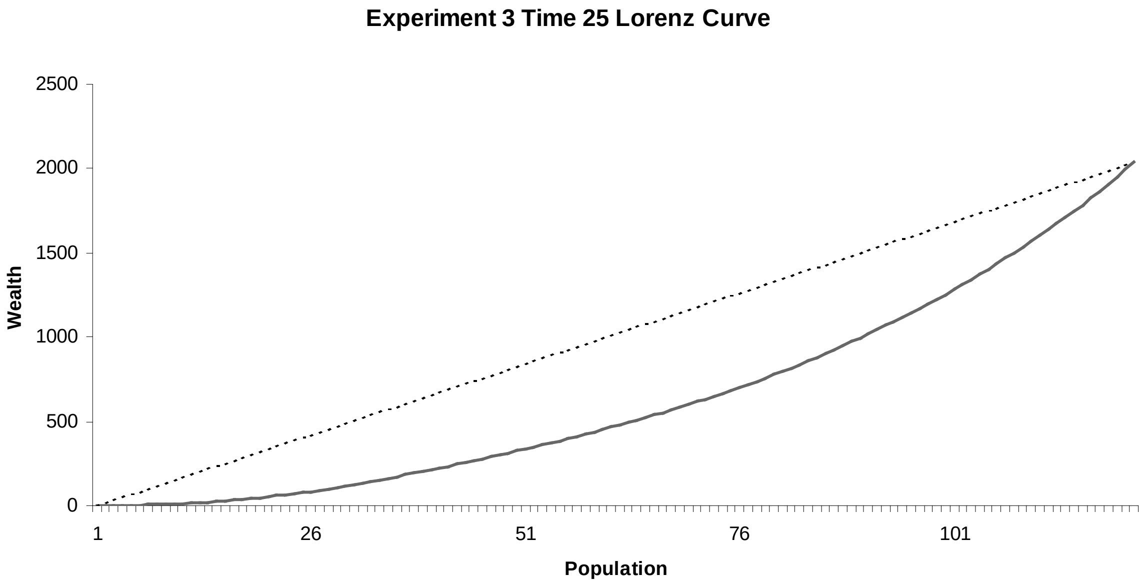
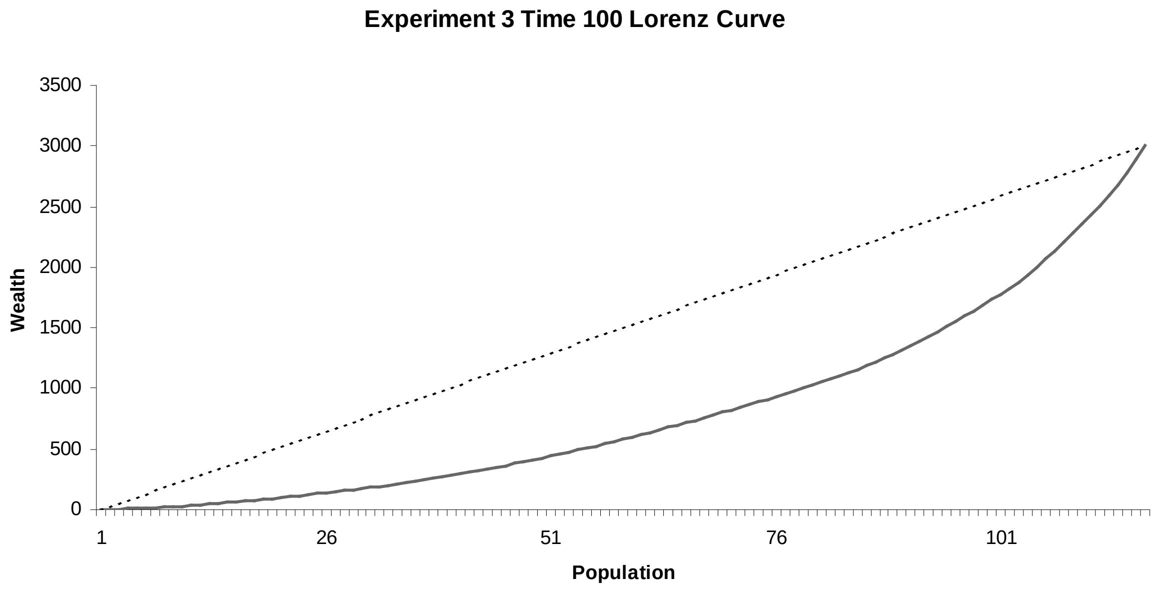


Figure 34. Experiment 3 Lorenz Curves (continued)



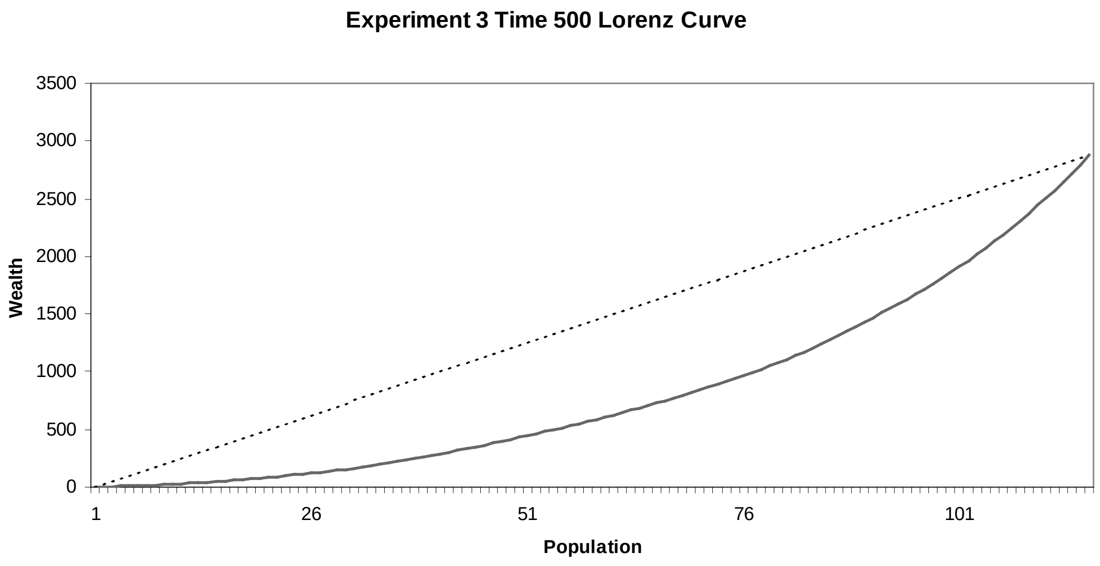


Figure 35. Experiment 3 Lorenz Curves (continued)

Figure 36 shows the Gini coefficient for Experiment 3 as a function of time. The Gini coefficient [23] is used to gauge the equity of wealth distribution in a society (0.0 is perfectly equitable; 1.0 is perfectly inequitable). The Gini coefficient is calculated as the ratio of the area between (1) the 45° line and the Lorenz curve, and (2) the area below the Lorenz curve. Figure 36 shows that the wealth distribution rapidly becomes more skewed, then settles into a range from roughly 0.4 to 0.5.

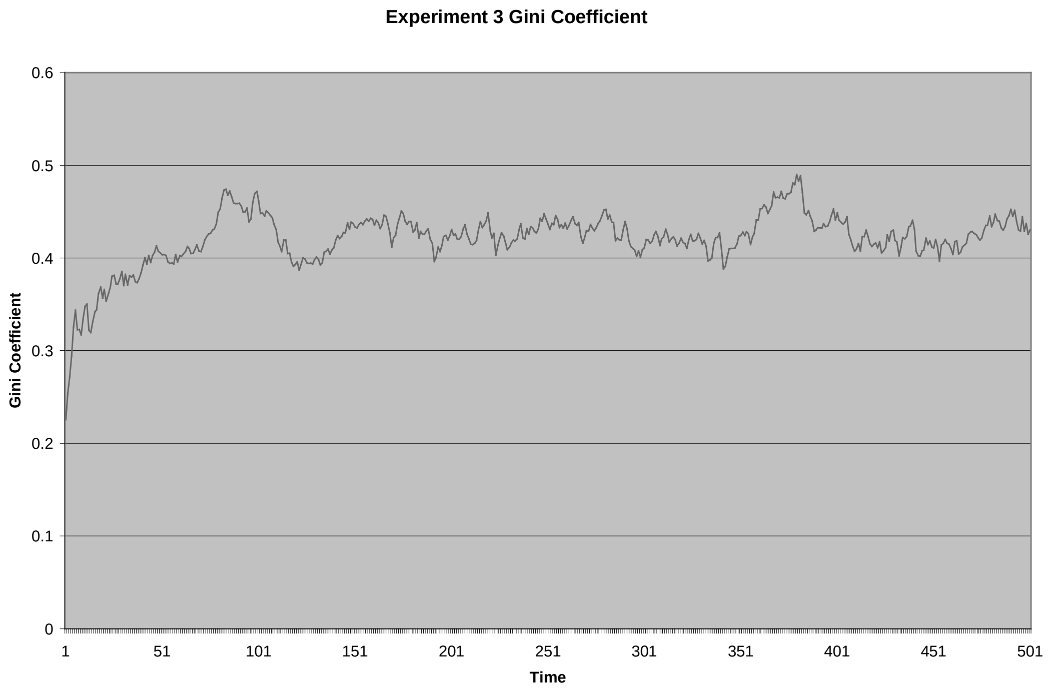


Figure 36. Experiment 3 Gini Coefficient

The wealth distribution is a statistic of note for social science and economics because it is considered a leading indicator for a number of issues: societal stability, capital formation, economic efficiency, and others. The ability to change agent rules and readily see the effect on wealth distribution is a key reason artificial societies are a useful research tool.

5.4 Experiment 4 — Seasonal Variations

5.4.1 Setup

Experiment 4 introduces seasonal variations into the XeriScape. The XeriScape is divided into two hemispheres, north and south. Each hemisphere alternates between a monsoon season and a dry season; when it is monsoon season in the northern hemisphere, it is dry in the southern, and vice versa. During monsoon season, the water resource is replenished at a normal rate. During the dry season, the water resource is replenished at a greatly reduced rate, ⅛ of the monsoon rate. Each season lasts approximately 50 cycles. In other respects, this experiment resembles Experiment 2.

5.4.2 Experimental Frame

The experimental frame for Experiment 4 is shown in Figure 37 below.

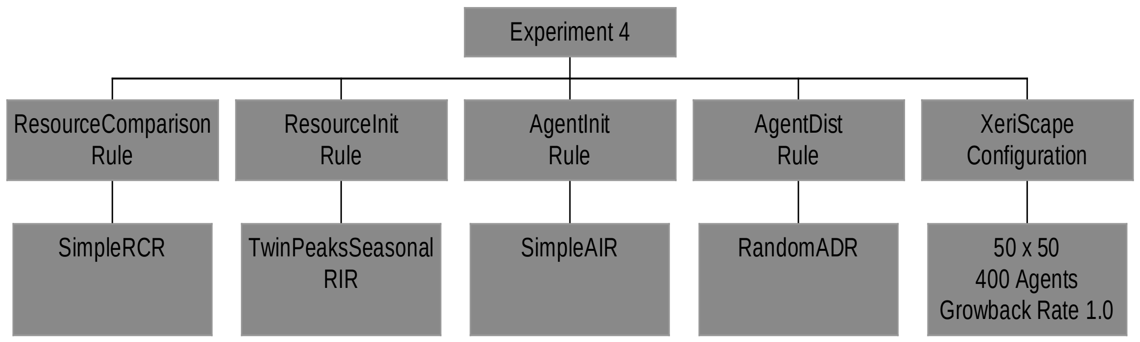
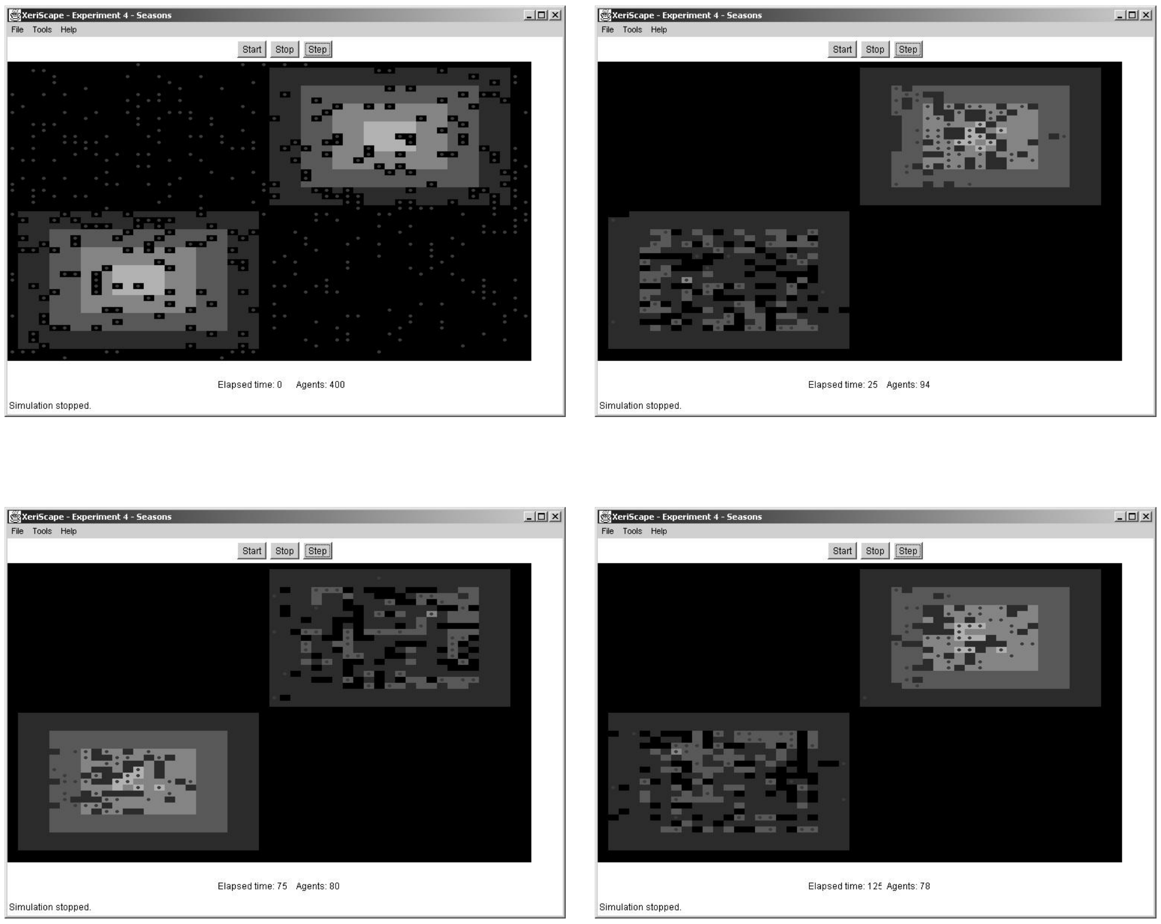


Figure 37. Experiment 4 Frame

5.4.3 Screen Captures

Figure 38 and Figure 39 show the screen captures from Experiment 4.



Elapsed time: 0 Agents: 400

Simulation stopped.

Elapsed time: 25 Agents: 94

Simulation stopped.

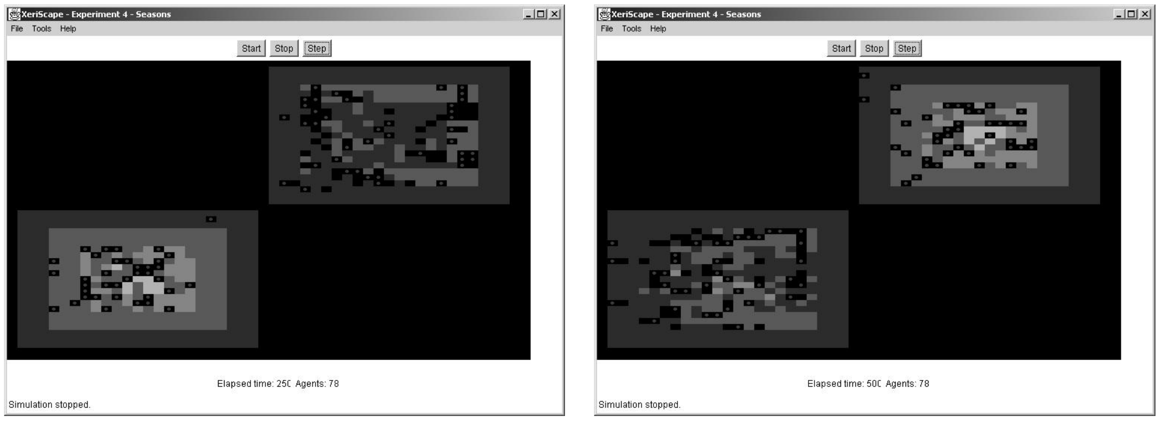
Elapsed time: 75 Agents: 80

Simulation stopped.

Elapsed time: 125 Agents: 78

Simulation stopped.

Figure 38. Experiment 4 Screen Captures


XeriScape - Experiment 4 - Seasons
File Tools Help
Start Stop Step
XeriScape - Experiment 4 - Seasons
File Tools Help
Start Stop Step

Elapsed time: 250 Agents: 78
Simulation stopped.

Elapsed time: 500 Agents: 78
Simulation stopped.

Figure 39. Experiment 4 Screen Captures (continued)

5.4.4 Discussion

Experiment 4 shows agents reacting to their environment. The agents outside the oases die off rapidly, and the stress of the seasonal variations results in even more agent mortality than in Experiment 2. Selection favors a low metabolic rate for the best chance of survival.

One might expect that agents would migrate with the seasons, seeking the more abundant water resources of the rainy season. Their limited vision rules this out, and the von Neumann neighborhood's lack of diagonals means that they cannot see the other oasis.

5.5 Experiment 5 — Pollution

5.5.1 Setup

This experiment explores the role of pollution in altering agent behavior and the environment. The experiment starts with 400 agents. As in Experiment 3, these agents have a finite lifetime that is randomly fixed between 60 and 100 time steps. Unlike

Experiment 3, these agents are not replaced when they die. A new rule, the TwinPeaksPollutionResourceInitRule, is used to set up the XeriScape cells. A new PollutingAgentInitRule creates these new agents (see section 3.2.1.3). The agents use a new rule, PollutionResourceComparisonRule, to determine their movements. According to this rule, agents will move to cells with the highest ratio of water to pollution.

5.5.2 Experimental Frame

The experimental frame for Experiment 5 is shown in Figure 40 below.

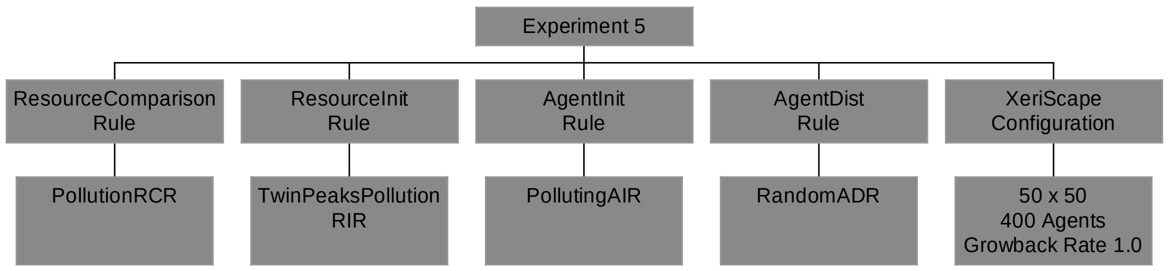
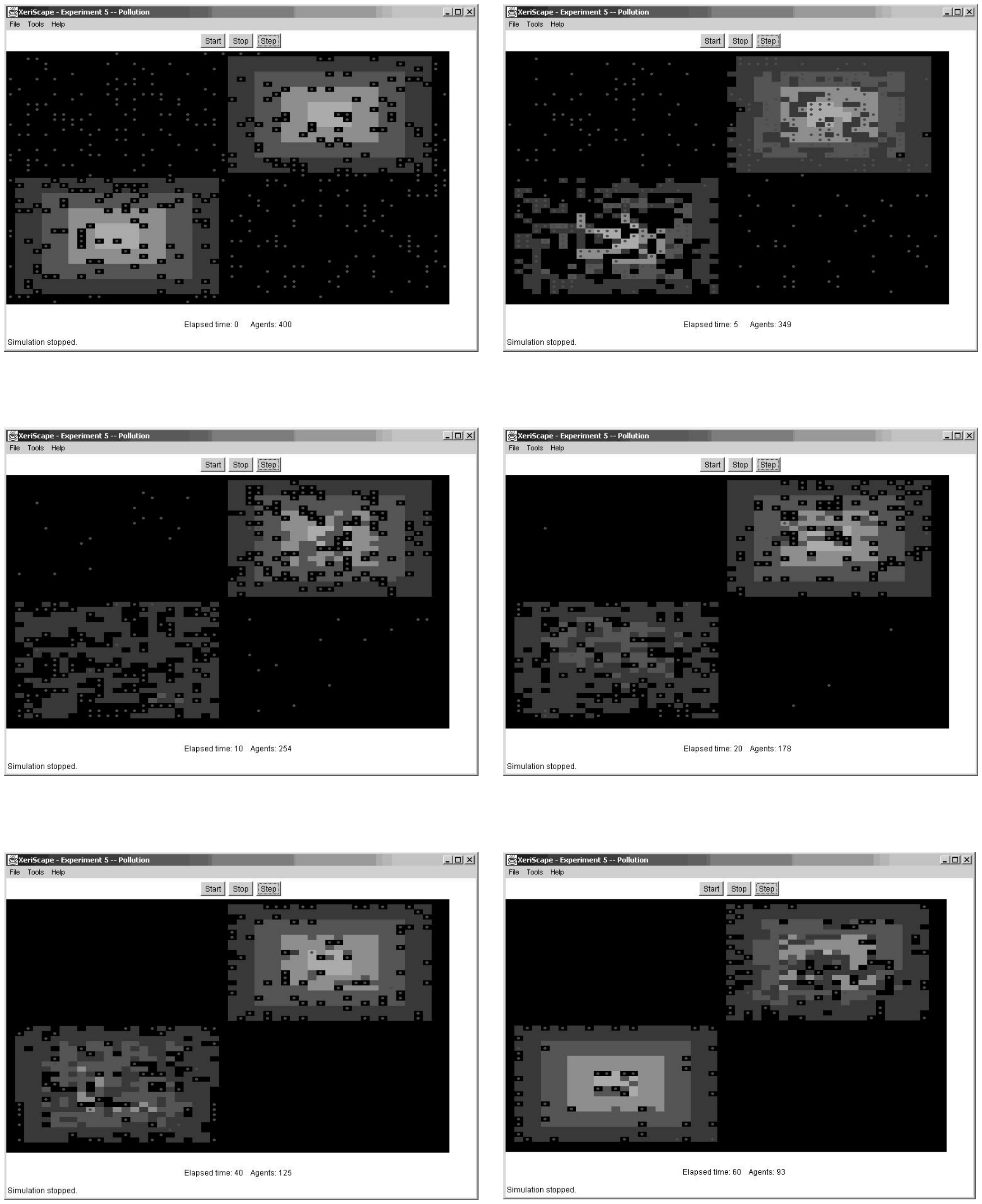


Figure 40. Experiment 5 Frame

5.5.3 Screen Captures

Screen captures from Experiment 5 are shown in Figure 41 below.



Elapsed time: 0 Agents: 400

Simulation stopped.

Elapsed time: 5 Agents: 349

Simulation stopped.

Elapsed time: 10 Agents: 254

Simulation stopped.

Elapsed time: 20 Agents: 178

Simulation stopped.

Elapsed time: 40 Agents: 125

Simulation stopped.

Elapsed time: 60 Agents: 93

Simulation stopped.

Figure 41. Experiment 5 Screen Captures

5.5.4 Discussion

Experiment 5 demonstrates agents reacting to their environment in a clear and simple way. As Figure 41 illustrates, as time progresses pollution levels in the oases build up so that, except for a few hardy souls, most agents flee for less inviting but cleaner areas. Since the replacement mode is off (see section 4.13), by the time of the last screen capture many of the agents have died.

5.6 Experiment 6 — Sexual Reproduction

5.6.1 Setup

Experiment 6 introduces the GenderedAgent class. Agents grow, consume, and gather as before. If they are fertile, have enough resources and are adjacent to an agent of the opposite sex who is also fertile, they also mate and produce a child. Agents will mate once every cycle if they have enough resources and a suitable partner is available. As explained in section 3.2.1.4, children receive a Mendelian cross of their parent's metabolism and vision attributes. The simulation starts with 400 agents randomly distributed (roughly 50% male and 50% female, randomly generated).

Other than their initial endowment of water, children receive no inheritance from their parents in this setup. This can be considered the equivalent of a 100% inheritance tax; there is no transmission of wealth between generations. The resources that an agent has gathered when it dies disappear from the environment.

5.6.2 Experimental Frame

The experimental frame for Experiment 6 is shown in Figure 42 below.

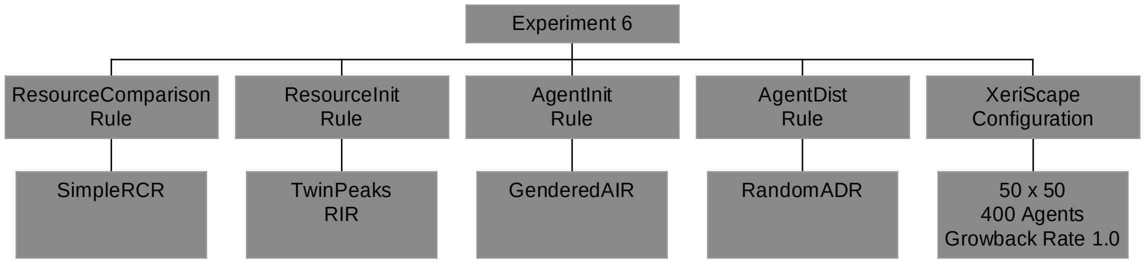
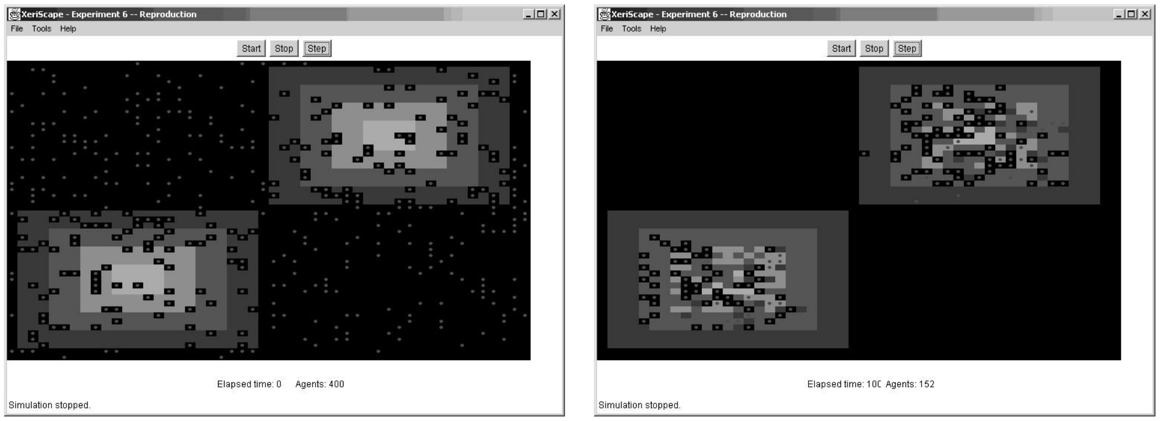


Figure 42. Experiment 6 Frame

5.6.3 Screen Captures

Experiment 6 screen captures are shown in Figure 43 and Figure 44 below.


XeriScape - Experiment 6 -- Reproduction
File Tools Help
Start Stop Step
XeriScape - Experiment 6 -- Reproduction
File Tools Help
Start Stop Step

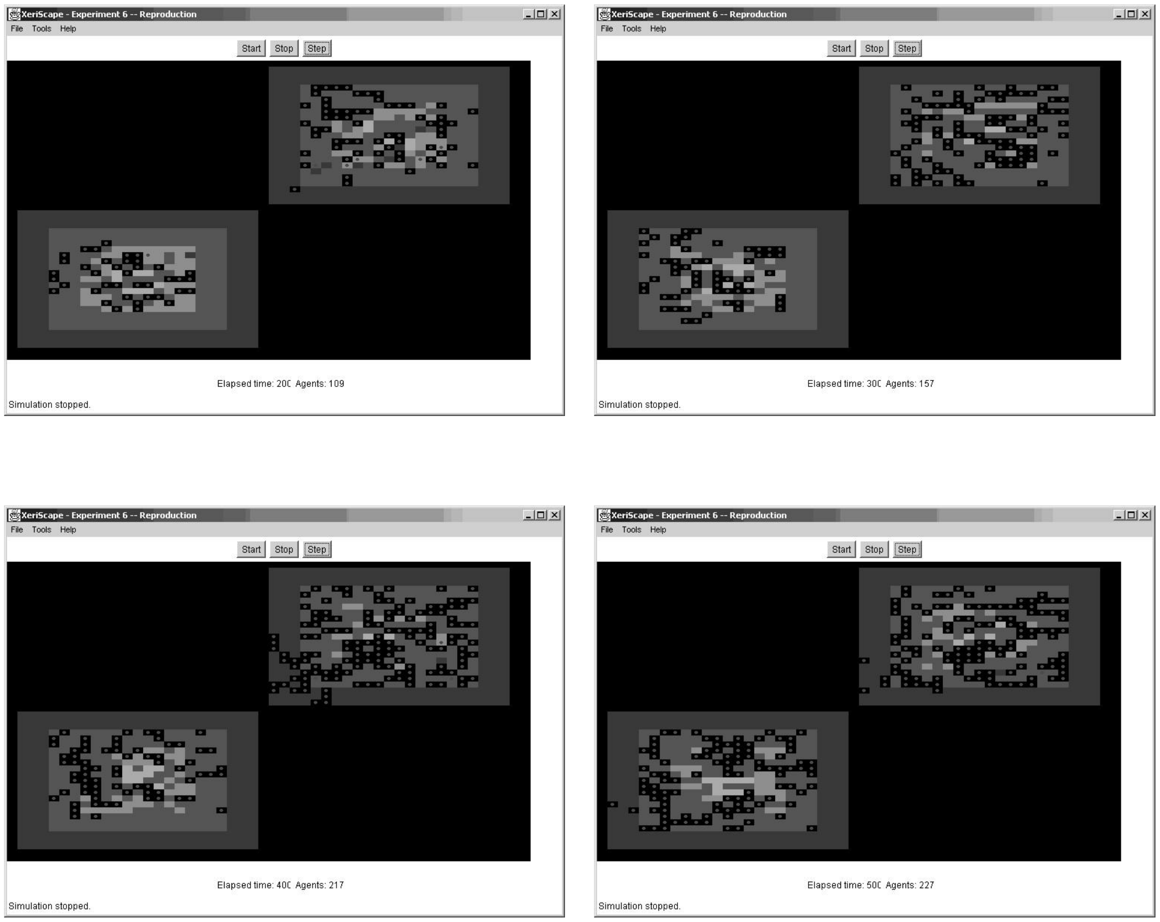
Elapsed time: 0 Agents: 400

Simulation stopped.

Elapsed time: 10(Agents: 152

Simulation stopped.

Figure 43. Experiment 6 Screen Captures



```
Elapsed time: 20( Agents: 109
Simulation stopped.

Elapsed time: 30( Agents: 157
Simulation stopped.

Elapsed time: 40( Agents: 217
Simulation stopped.

Elapsed time: 50( Agents: 227
Simulation stopped.
```

Figure 44. Experiment 6 Screen Captures (continued)

5.6.4 Discussion

In many respects, this experiment is the most interesting of all the XeriScape simulation runs; it is certainly the most complex in terms of data gathered. In examining the large amount of data generated in this simulation run, we can observe a number of interesting trends.

One example of an unexpected trend is shown in Figure 45 below. In previous experiments (see section 5.1.4) the population dropped off precipitously and then stabilized at a level much lower than the initial population. In Experiment 6, the population (that is, the carrying capacity) of the XeriScape with gendered agents actually increases with time.

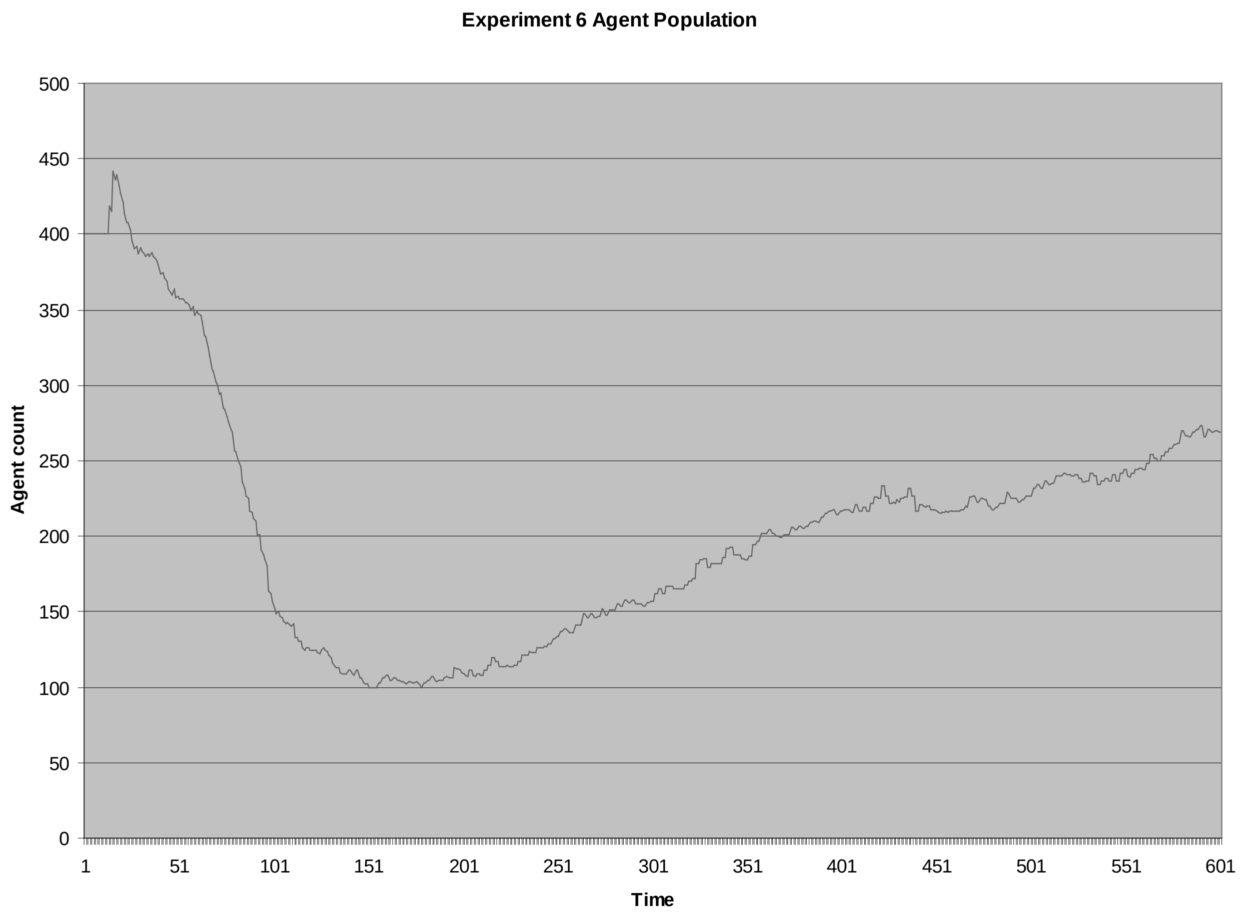


Figure 45. Experiment 6 Agent Population

Figure 46 and Figure 47 below show the effects of natural selection in action. Mean agent vision increases over time and mean agent metabolism decreases. This is due to the genetic selection occurring with each generation selecting for increased vision and decreased metabolism. Agents with these characteristics are more likely to breed, and will breed more often since they generally will have greater wealth.

This effect also explains the increase in carrying capacity; as mean metabolism decreases and mean vision increases, agents become more efficient gatherers and use fewer resources, so the XeriScape can support more of them.

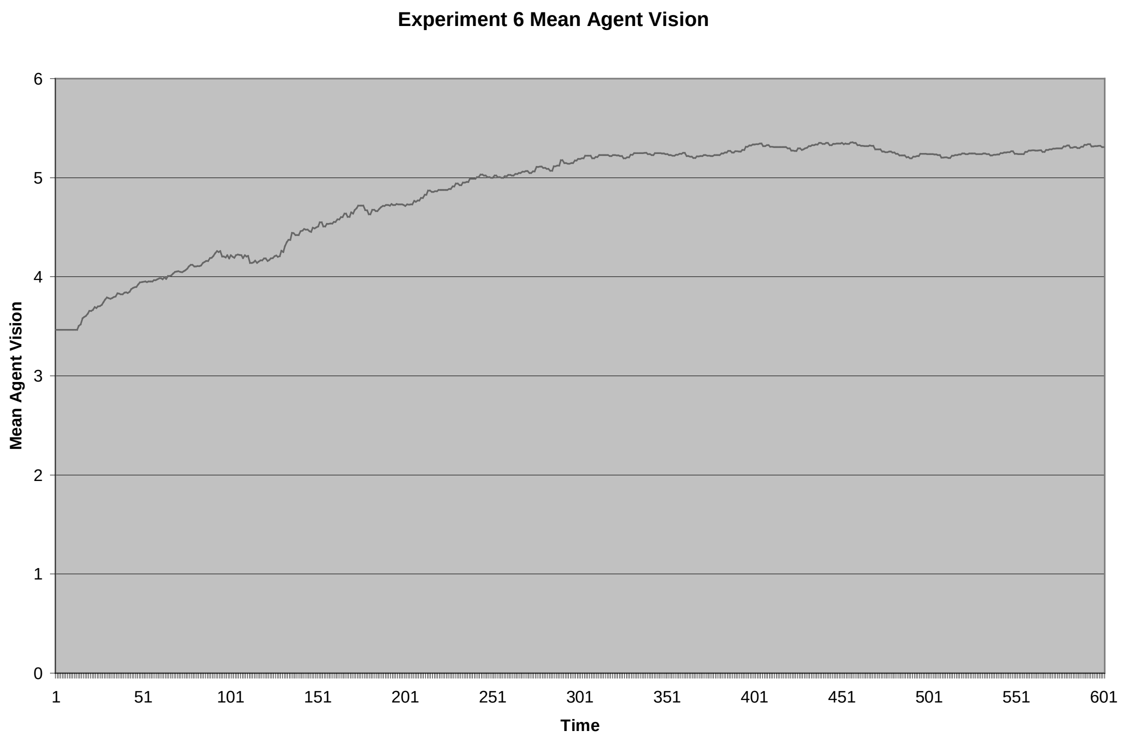


Figure 46. Experiment 6 Mean Agent Vision

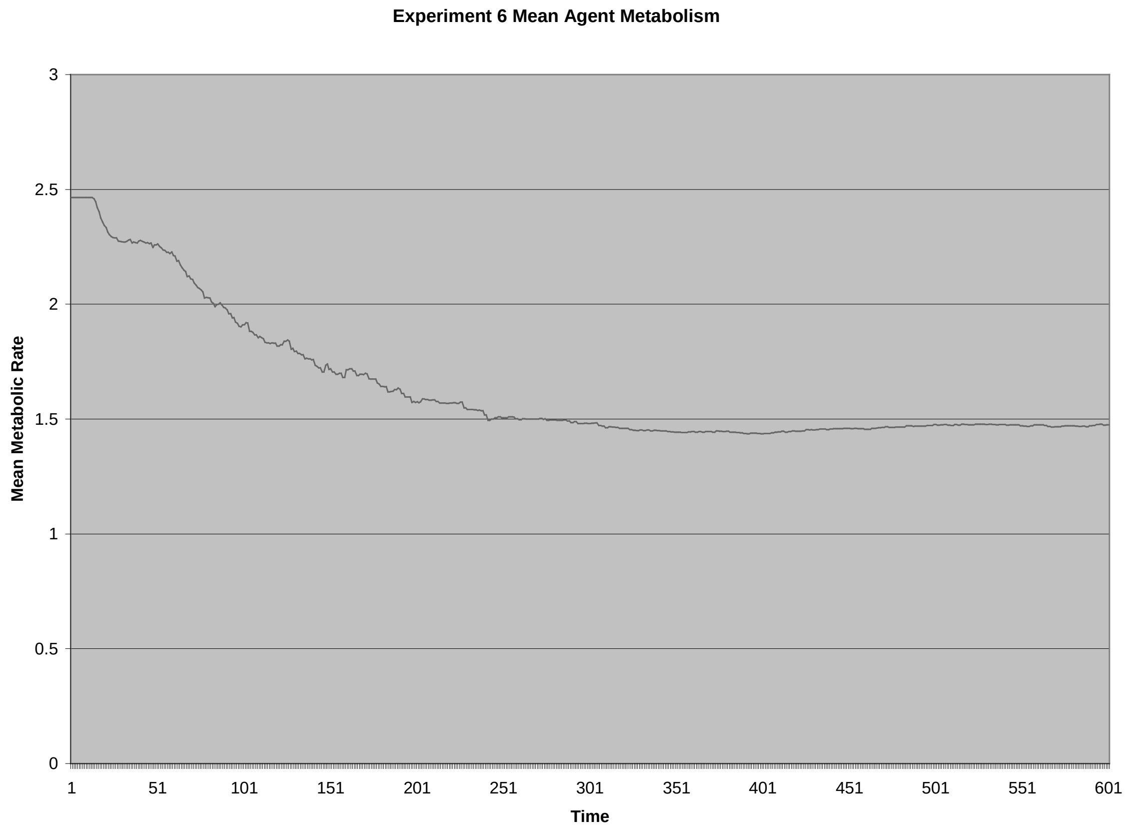
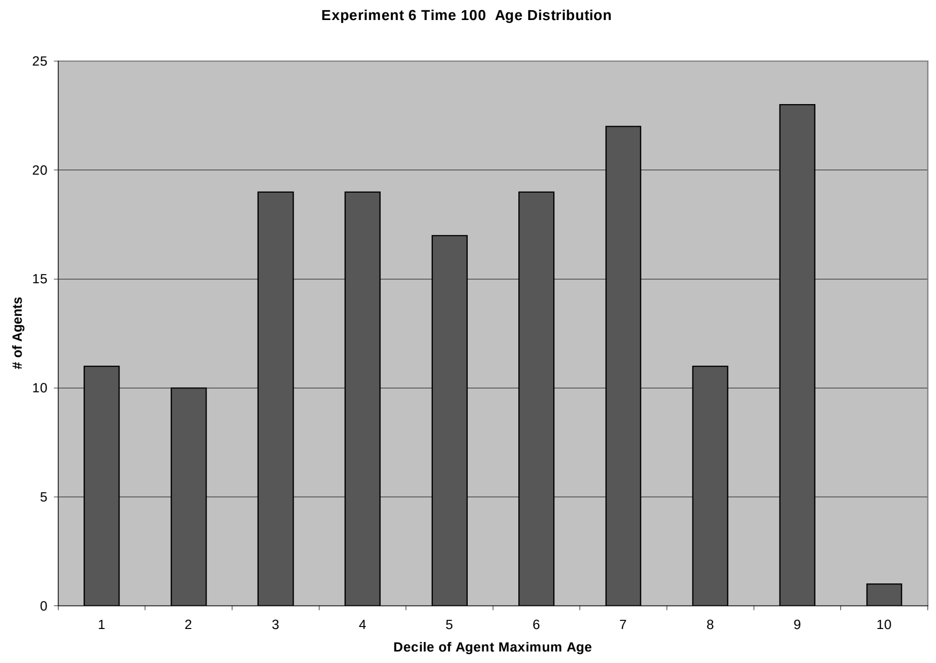


Figure 47. Experiment 6 Mean Agent Metabolism

Figure 48, Figure 49, and Figure 50 below show an age histogram for Experiment 6 at times 100, 200, 300, 400, 500 and 600. The horizontal axis of the histogram represents the population age broken out by decile of the maximum age. The vertical bars represent the number of agents in each decile. When looking at these histograms, it is important to bear in mind the agent population (carrying capacity) fluctuations seen in Figure 45. In general, we would expect that once the population level stabilizes the age distribution should be relatively equal. It would seem, however, that some fluctuations are still seen even at times 500 and 600.



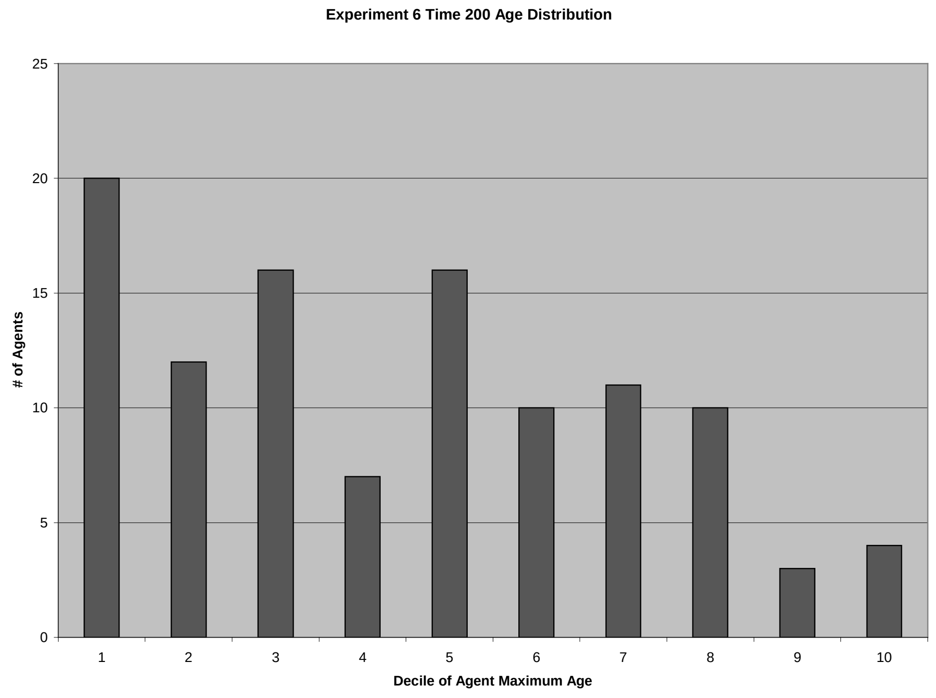
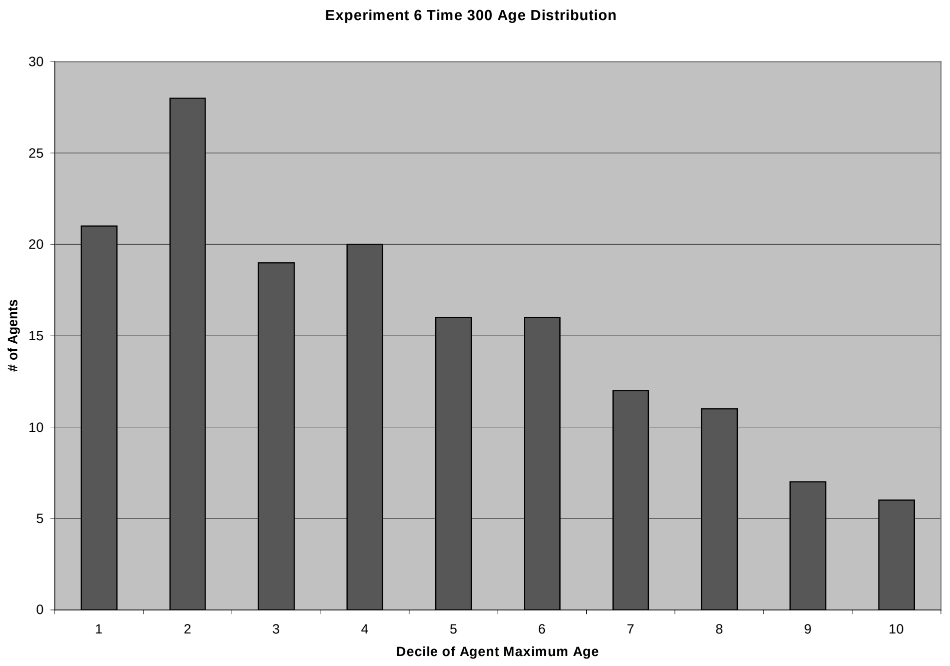


Figure 48. Experiment 6 Age Distribution



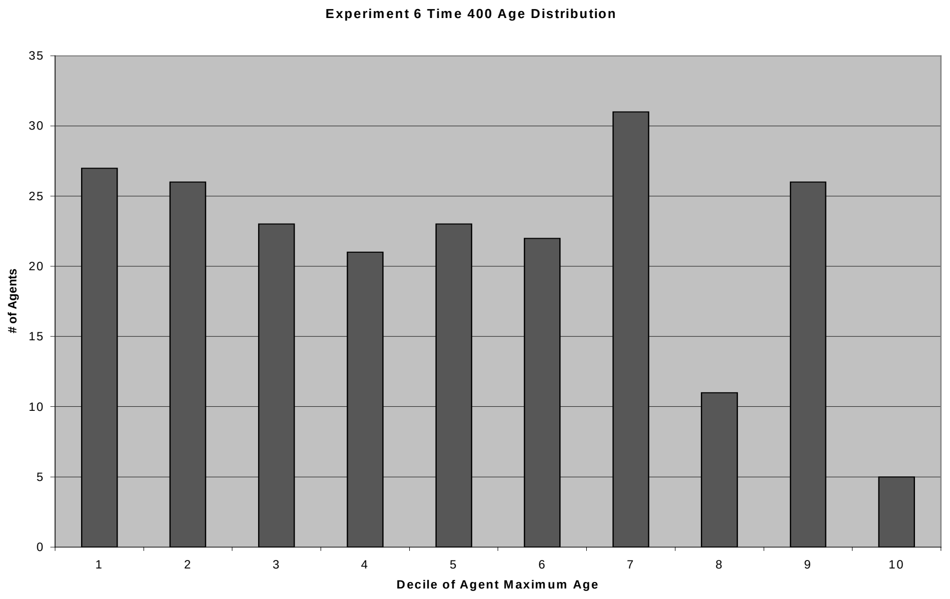
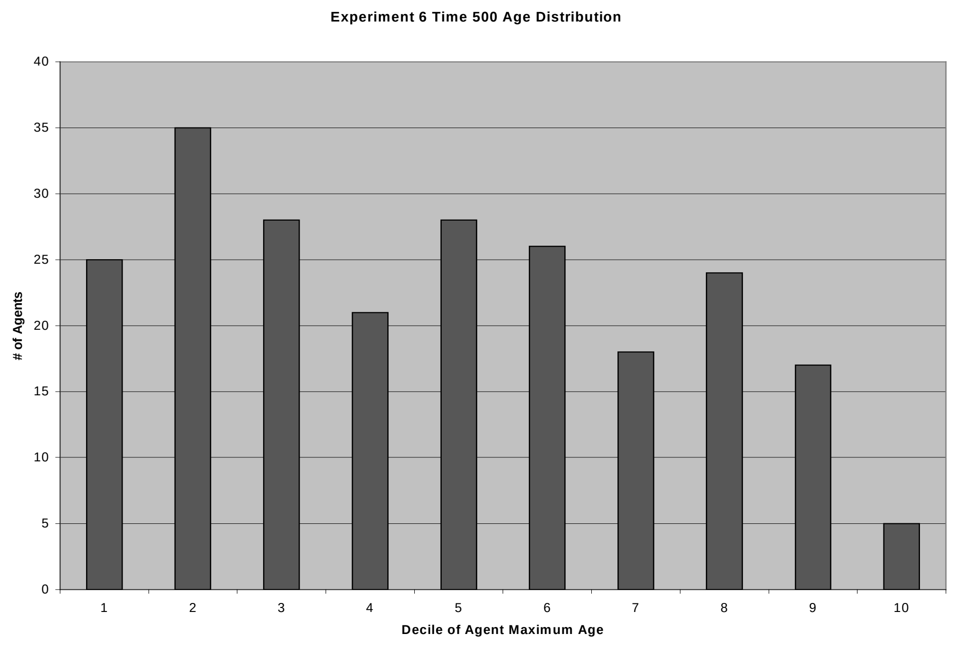


Figure 49. Experiment 6 Age Distribution (continued)



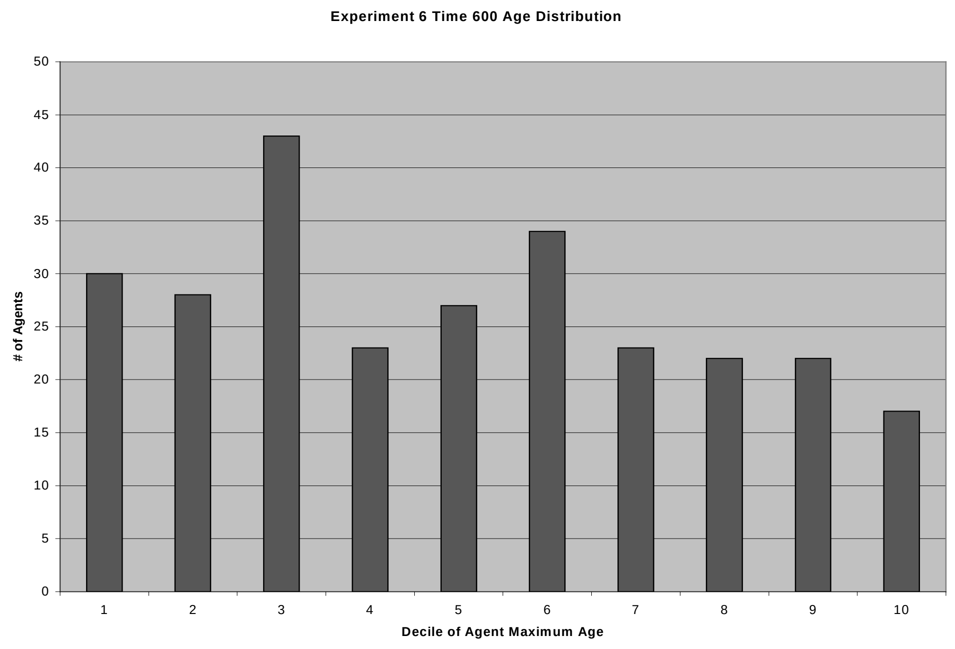
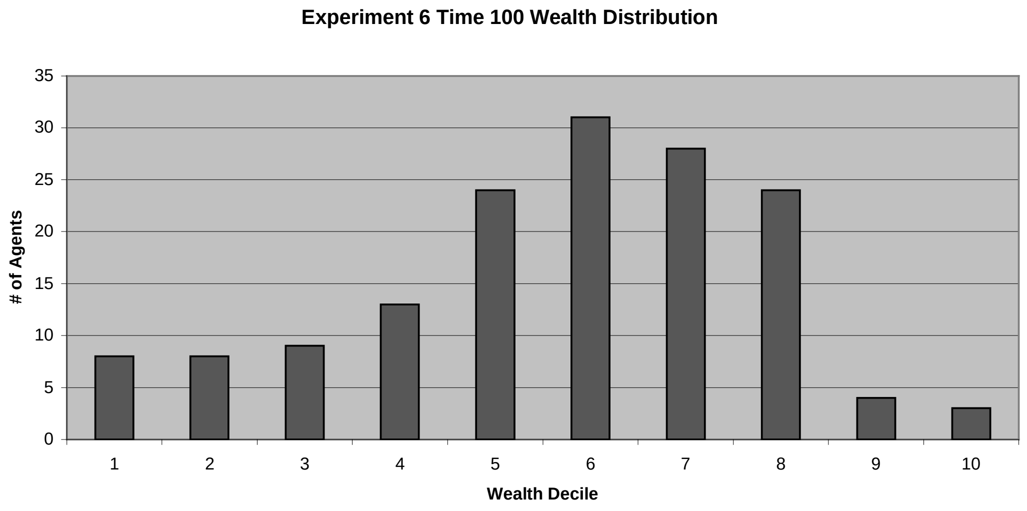


Figure 50. Experiment 6 Age Distribution (continued)

Figure 51, Figure 52, and Figure 53 show the wealth distribution histograms for Experiment 6. In these histograms, the vertical bars represent the number of agents for each decile of the total wealth of the simulation.



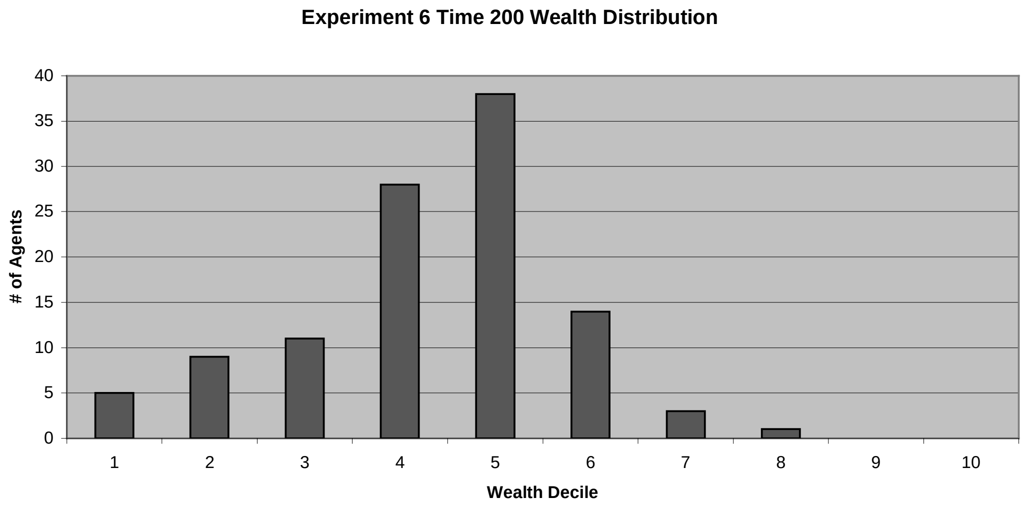
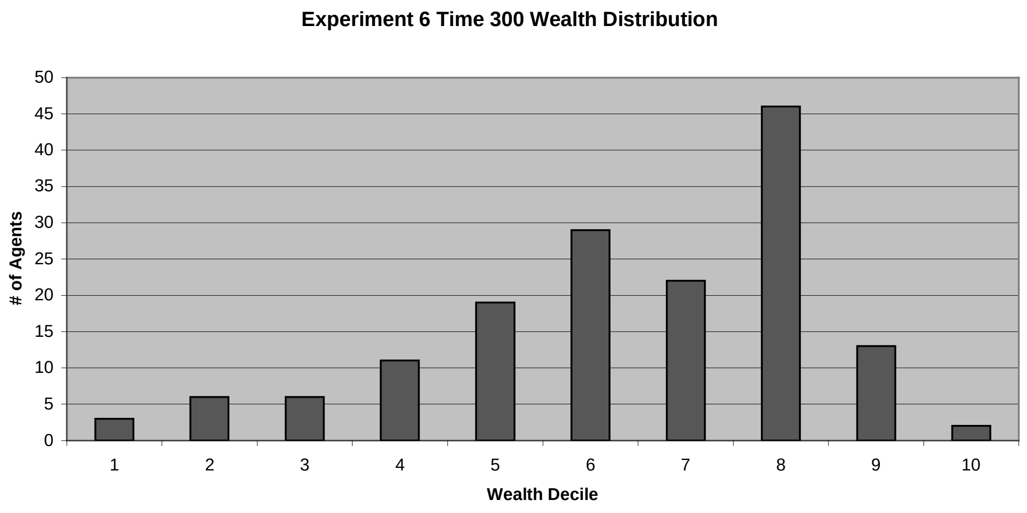


Figure 51. Experiment 6 Wealth Distribution



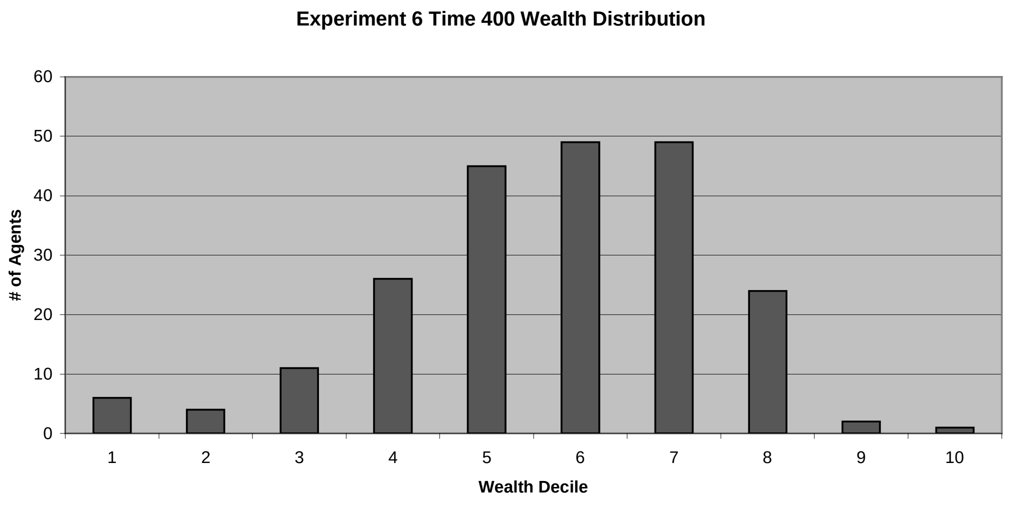
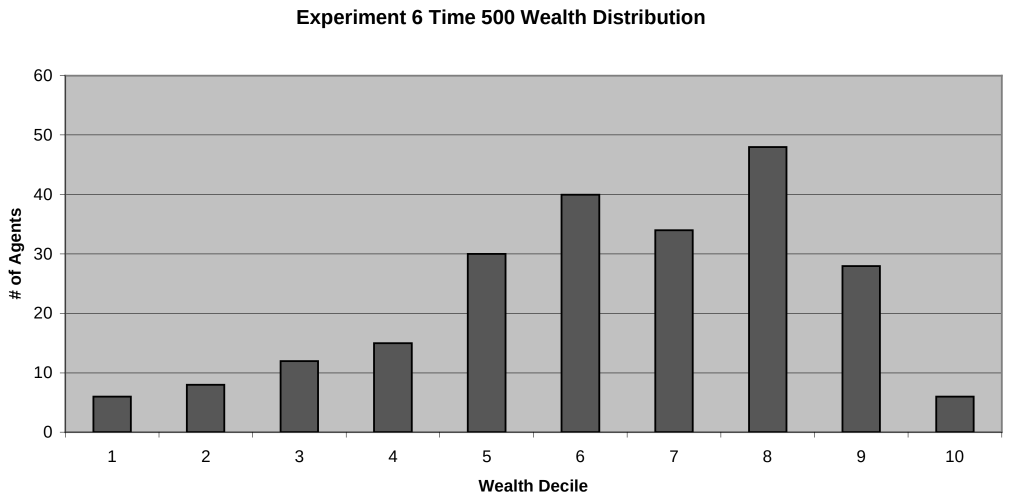


Figure 52. Experiment 6 Wealth Distribution (continued)



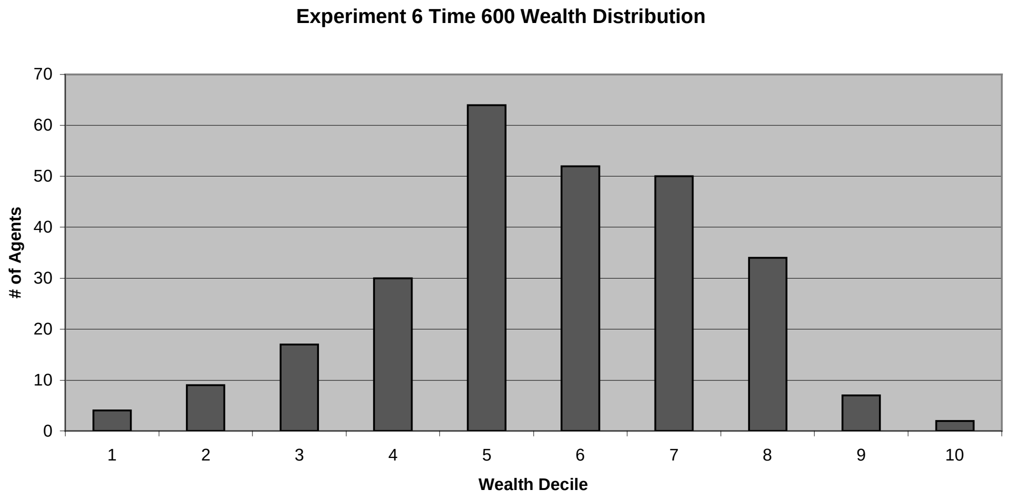


Figure 53. Experiment 6 Wealth Distribution (continued)

Figure 54, Figure 55, Figure 56, and Figure 57 show Lorenz curves for Experiment 6; the corresponding Gini coefficients are shown in Table 1. Figure 58 shows perhaps more intelligibly the trend: after an initial increase, the distribution of wealth stabilizes somewhat to be relatively more equitable than was seen in Experiment 3.

Gini Coefficient	**Time**
0.2216689	100
0.20001036	200
0.17675966	300
0.17704827	400
0.19694364	500
0.18891907	600

Table 1. Experiment 6 Selected Gini Coefficients

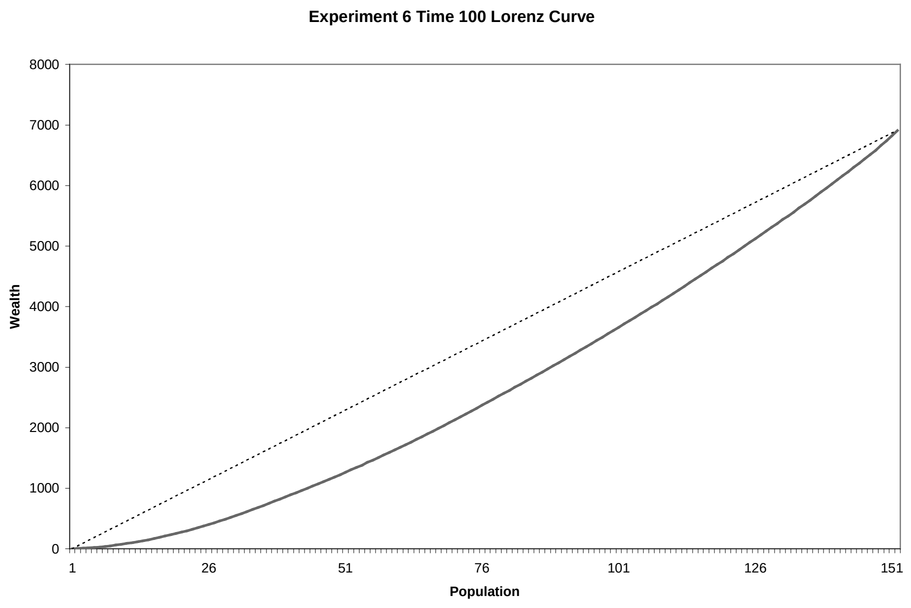
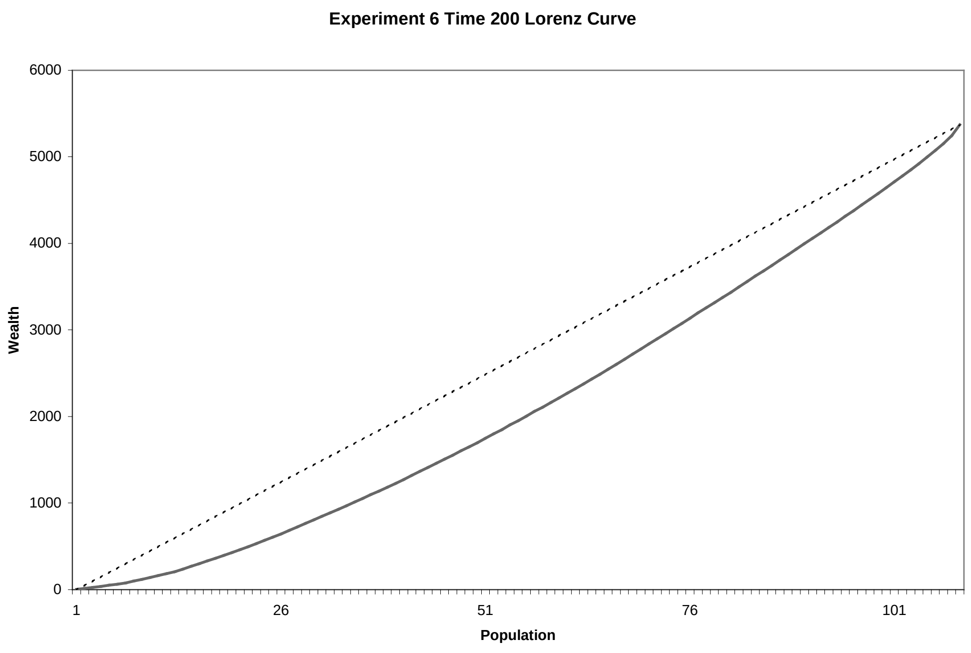


Figure 54. Experiment 6 Lorenz Curves



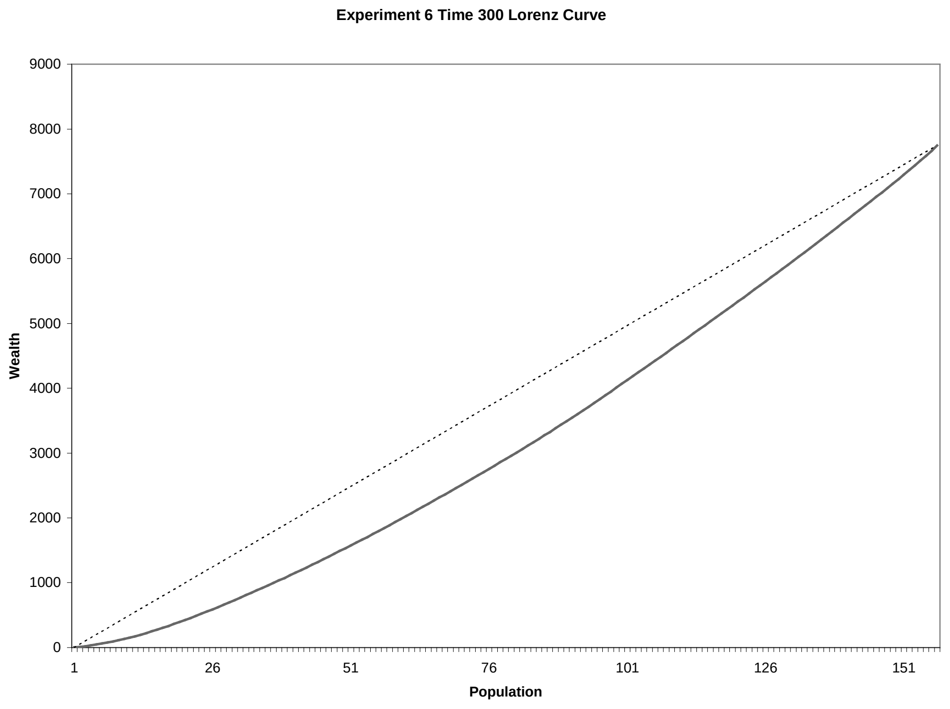
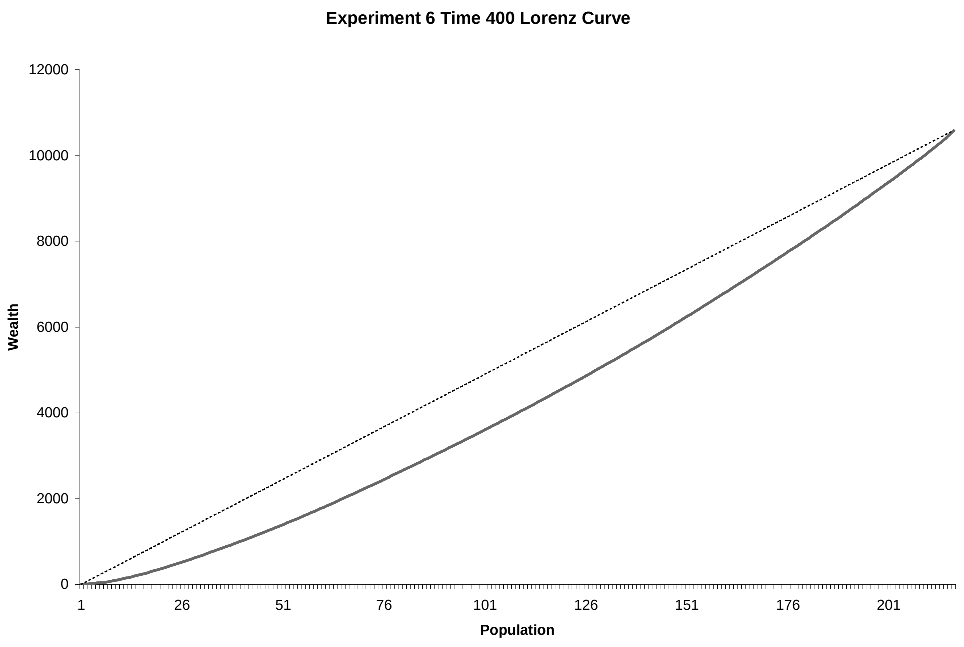


Figure 55. Experiment 6 Lorenz Curves (continued)



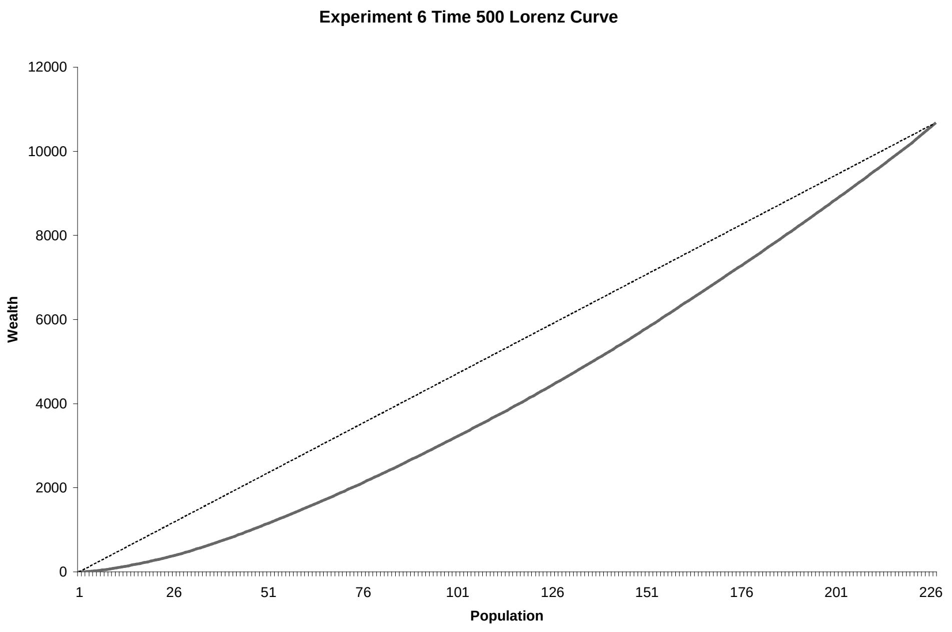


Figure 56. Experiment 6 Lorenz Curves (continued)

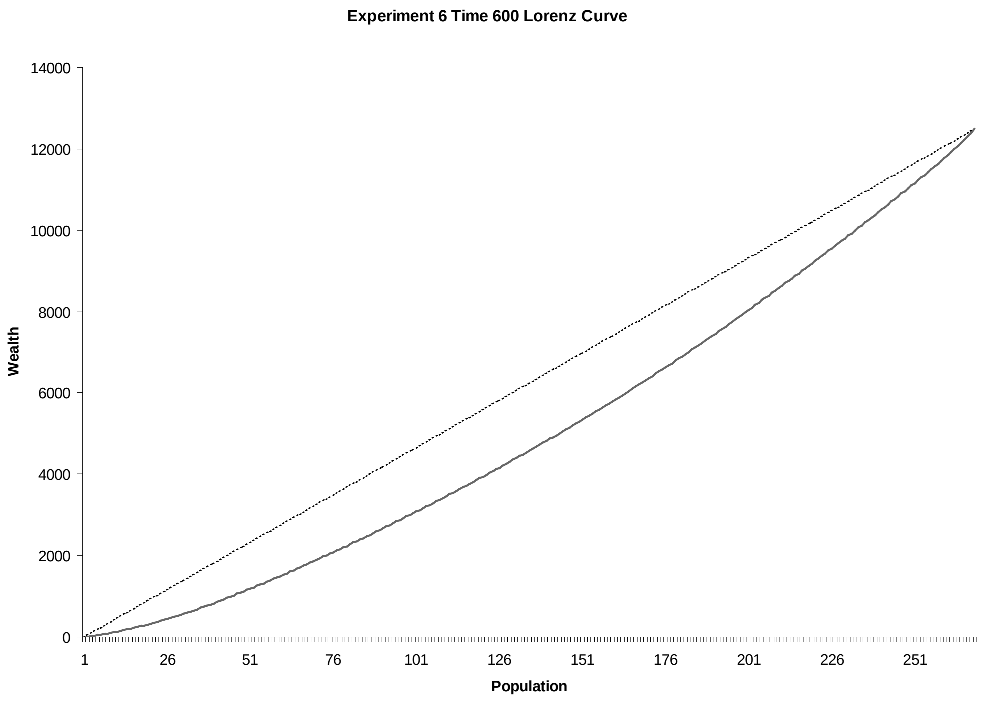


Figure 57. Experiment 6 Lorenz Curves (continued)

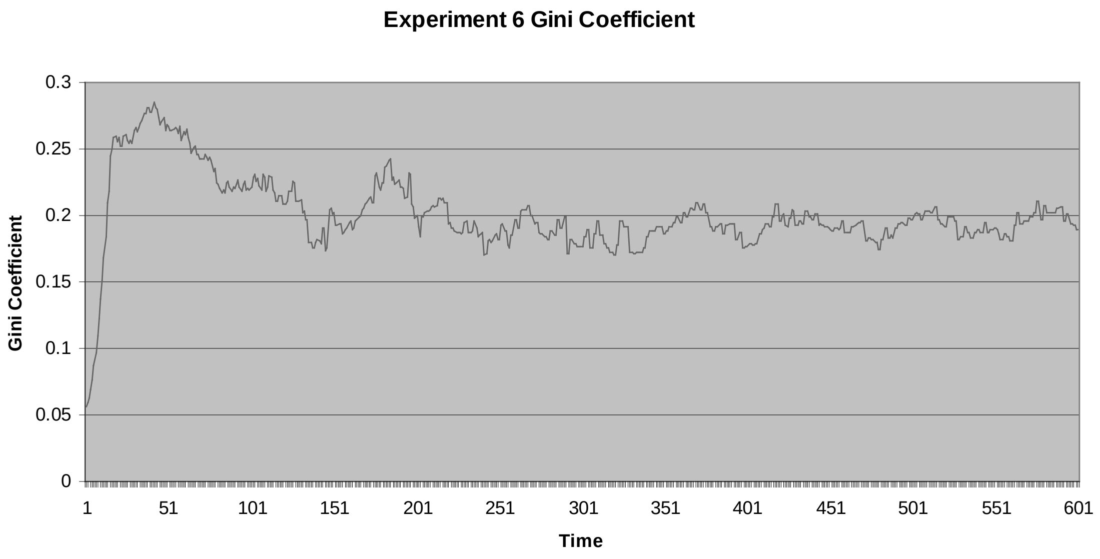


Figure 58. Experiment 6 Gini Coefficient

One may well ask why the society of Experiment 6 would be more equitable than that of Experiment 3. One explanation would lie in the effect of the large families the wealthy tend to have in this society. Wealth in Experiment 6 tends not to accumulate, but instead enables the agent to have more children. Since there is no inheritance mechanism the resource wealth in the society is passed on to children at their birth. The result is that the wealth is spread throughout the society and not concentrated as it might otherwise be.

5.7 Experiment 7 — Mixed Finite and Polluting Agents

5.7.1 Setup

Experiment 7 illustrates the enhanced capabilities made possible by DEVS-Java. Sugarscape is not capable of mixing different classes of agents in one simulation, but DEVS-Java allows XeriScape to do this readily. In this experiment, finite (non-polluting) and polluting agents (see the class hierarchy in Figure 13) are mixed in the simulation and coexist side by side. Recall that the polluting agents produce and are sensitive to pollution; finite agents do not produce pollution and are not sensitive to it. The Java reflection mechanism allows the simulation to determine the type of each agent and process it appropriately. Java's inheritance and polymorphism allow agent classes to be written as easily and efficiently as possible, taking advantage of common traits.

The simulation starts with a mix (50% finite, 50% polluting) of 400 agents randomly dispersed throughout the environment.

5.7.2 Experimental Frame

The experimental frame for Experiment 7 is shown in Figure 59 below.

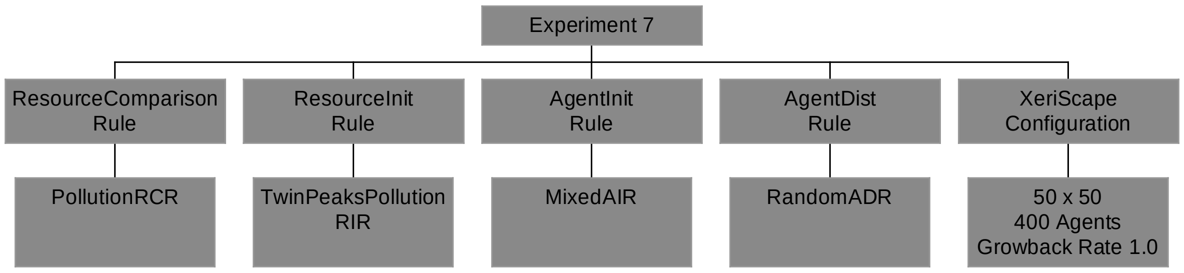
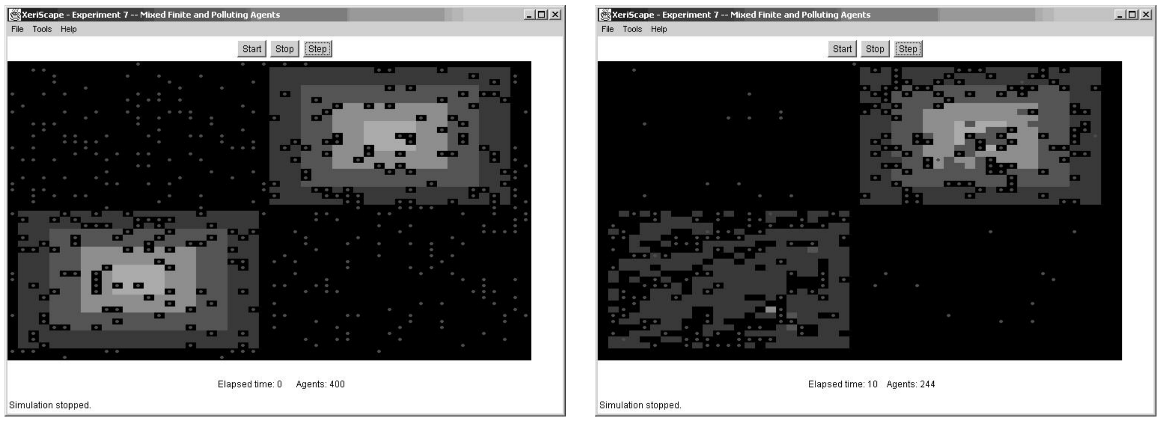


Figure 59. Experiment 7 Frame

5.7.3 Screen Captures

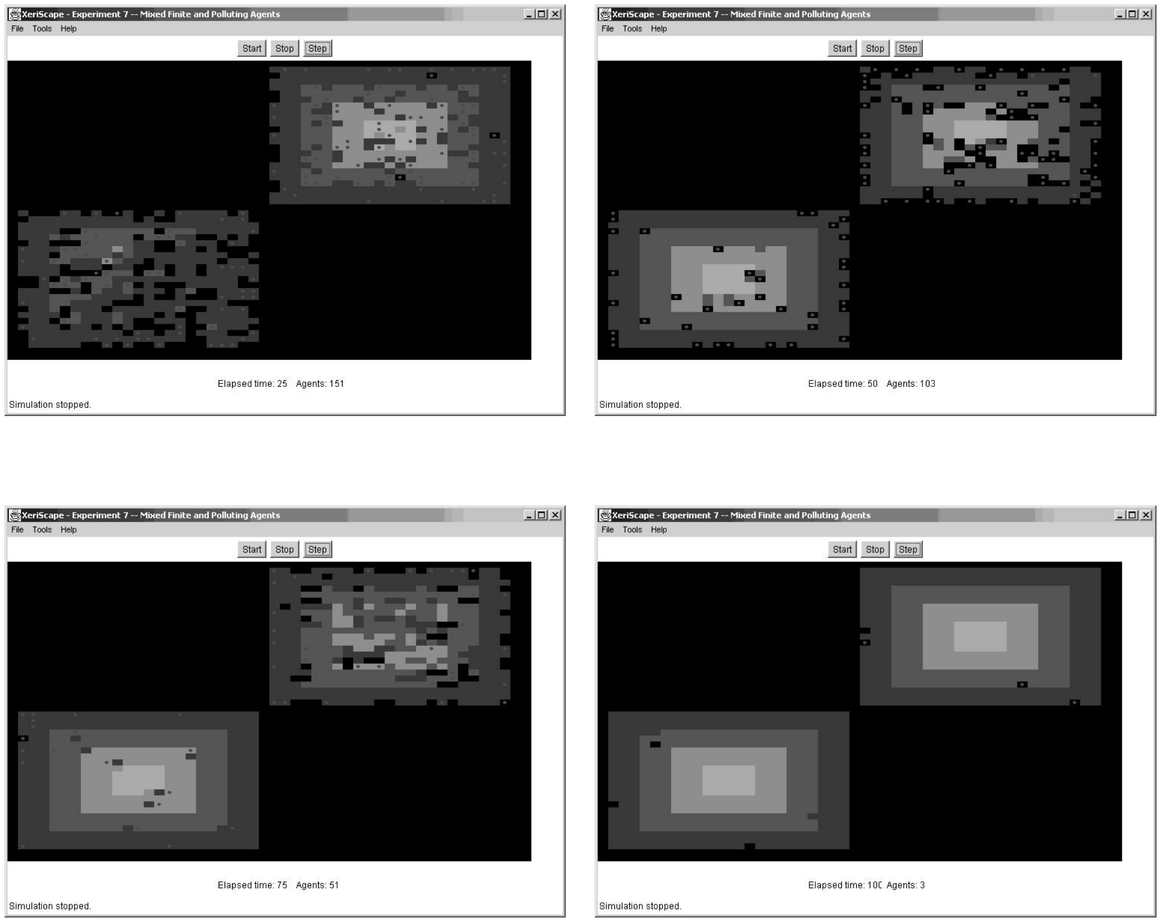
The Experiment 7 screen captures are shown in Figure 60 and Figure 61 below.


XeriScape - Experiment 7 -- Mixed Finite and Polluting Agents
File
Tools
Help
Start
Stop
Step
XeriScape - Experiment 7 -- Mixed Finite and Polluting Agents
File
Tools
Help
Start
Stop
Step

Elapsed time: 0 Agents: 400
Simulation stopped.

Elapsed time: 10 Agents: 244
Simulation stopped.

Figure 60. Experiment 7 Screen Captures


XeriScape - Experiment 7 -- Mixed Finite and Polluting Agents
File Tools Help
Start Stop Step

Elapsed time: 25 Agents: 151
Simulation stopped.

Elapsed time: 50 Agents: 103
Simulation stopped.

Elapsed time: 75 Agents: 51
Simulation stopped.

Elapsed time: 10(Agents: 3
Simulation stopped.

Figure 61. Experiment 7 Screen Captures (continued)

5.7.4 Discussion

Experiment 7 shows the enhanced capabilities of XeriScape, its flexibility and extensibility. Figure 60 and Figure 61 show behavior similar to that seen in Experiment 5, except that since the finite agents are not sensitive to the pollution being generated by the polluting agents, they continue to occupy the areas of the XeriScape that have the most water resources even though those areas are heavily polluted. The polluting agents move off the resource peaks due to the pollution present there.

As in Experiment 5, since these agents have limited lifetimes and replacement mode is off, by the time of the last screen capture many of the agents have died.

5.8 Experiment 8 — Variable Time Steps

5.8.1 Setup

Experiment 8 illustrates another capability made possible by DEVS-Java. Experiment 8 is based on Experiment 7's mixing of finite and polluting agents. In addition, each agent has its own time step, that is, each agent has its own level of activity, which determines the rate at which its life cycle (see Figure 4) is executed. The time step varies from agent to agent, but it is fixed for the life of each agent. The time step values are randomly distributed from 5 to 15 time units. Since agents will not be waking up as often, they are given a larger initial resource value to prevent them from dying before they wake up.

This variable time step translates the discrete time simulation approach of Sugarscape into a true discrete event simulation. It can be viewed as a measure or indicator of the agent's vitality or energy level, or perhaps of its will to survive or alertness. Since the agents do not cycle in lock step the simulation is more like a real society, where individuals move at their own pace.

As in Experiment 7, Experiment 8 starts with 400 mixed finite (non-polluting) and polluting agents distributed randomly throughout the environment.

5.8.2 Experimental Frame

The experimental frame for Experiment 8 is shown in Figure 62 below.

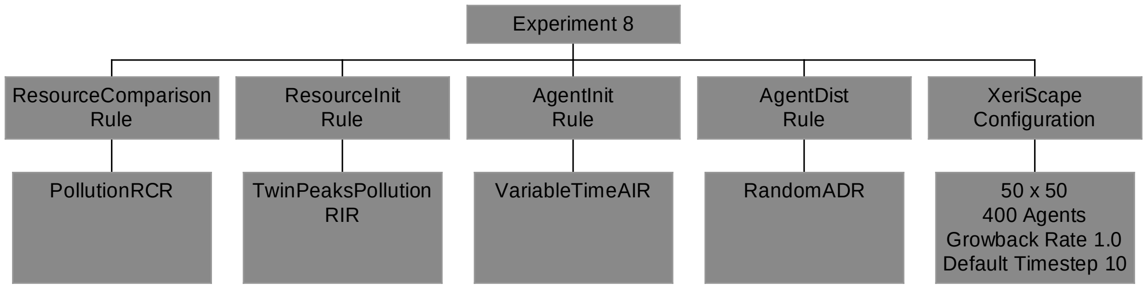
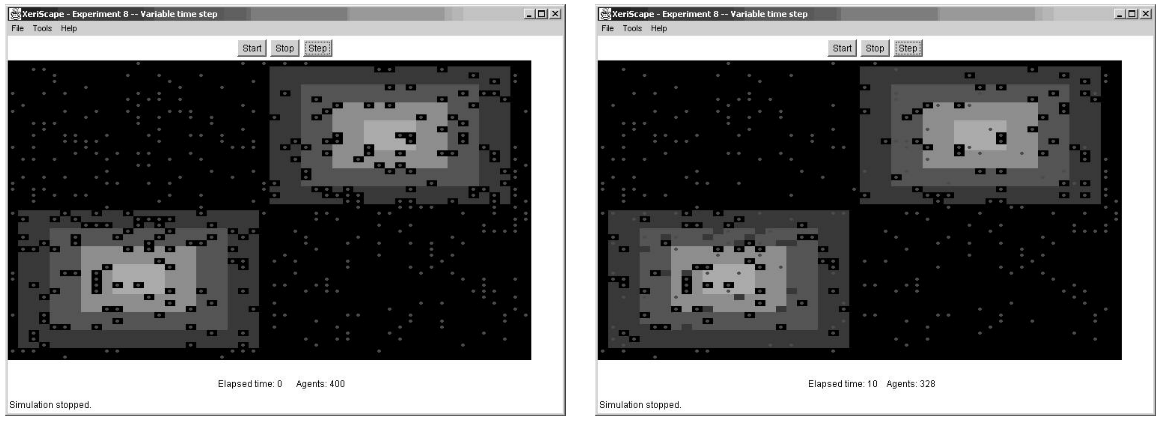


Figure 62. Experiment 8 Frame

5.8.3 Screen Captures

Screen captures for Experiment 8 are shown in Figure 63 and Figure 64 below.


XeriScape - Experiment 8 -- Variable time step
File Tools Help
Start Stop Step
XeriScape - Experiment 8 -- Variable time step
File Tools Help
Start Stop Step

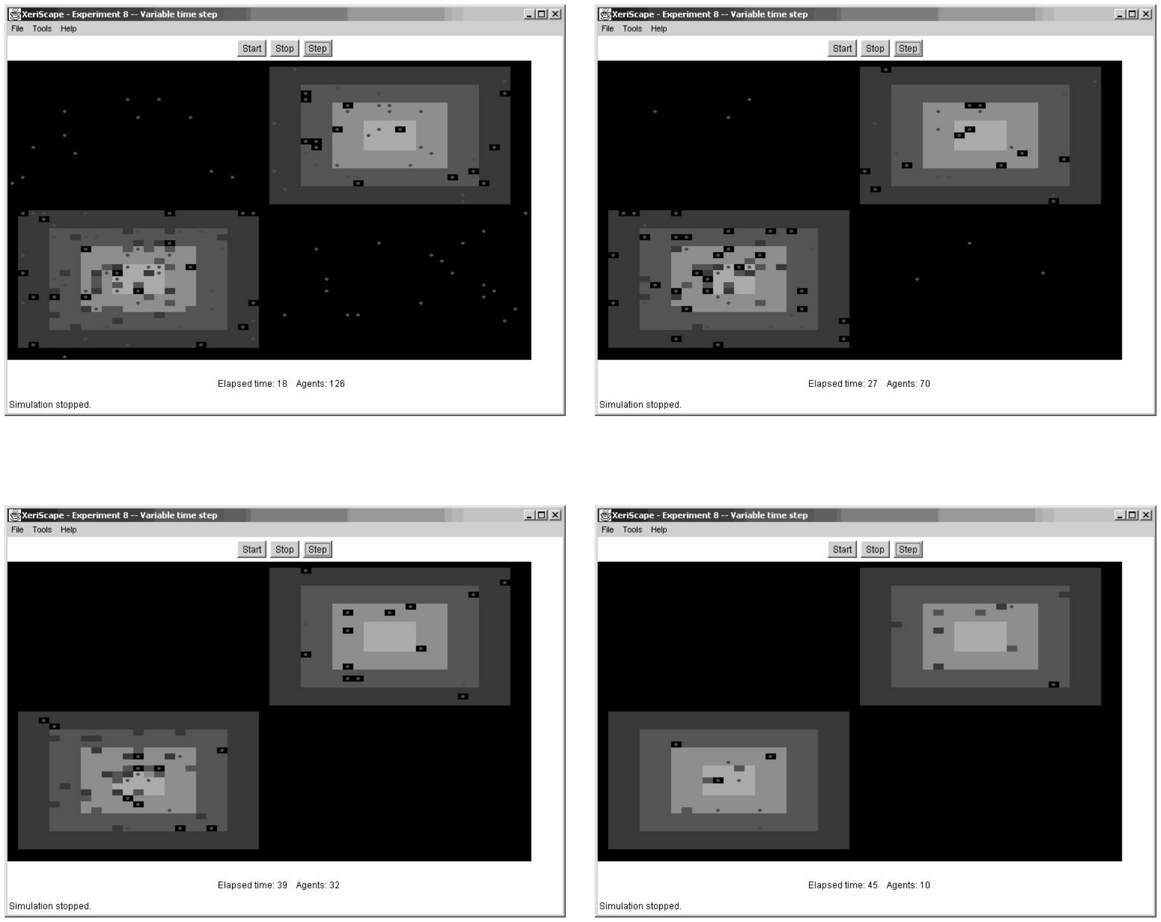
Elapsed time: 0 Agents: 400

Simulation stopped.

Elapsed time: 10 Agents: 328

Simulation stopped.

Figure 63. Experiment 8 Screen Captures



Elapsed time: 18 Agents: 126

Simulation stopped.

Elapsed time: 27 Agents: 70

Simulation stopped.

Elapsed time: 39 Agents: 32

Simulation stopped.

Elapsed time: 45 Agents: 10

Simulation stopped.

Figure 64. Experiment 8 Screen Captures (continued)

5.8.4 Discussion

This experiment shows, in a particularly striking fashion, the enhanced capabilities DEVS-Java makes possible. While the results in general resemble those of Experiment 7, the ability of the simulation to respond to each individual agent means that each time step is reached only if one or more agents wake up at that time step. Thus, during a simulation run, the clock may step from 5, to 7, to 8, to 10, and so forth. This means the simulation is as efficient as possible since no computations are done for the time steps that are stepped over. For example, executing the Experiment 8 simulation for the first 50 time units required only 33 computation cycles instead of the 50 that Sugarscape would have required.

6 FUTURE DIRECTIONS AND CONCLUSIONS

6.1 Research Questions

Can DEVS be applied to artificial society simulations? Clearly, the answer is a resounding “yes.” The XeriScape system is a straightforward implementation of a Sugarscape-style artificial society.

Can DEVS do artificial society simulations efficiently and quickly? Again, the answer is “yes.” DEVS-Java’s object-orientation and discrete event formalism mean that only the minimum computation is performed. Figure 65 below is a graph of the cumulative active cells in XeriScape vs. a discrete time simulation of equivalent size, that is, the cumulative number of cell computation cycles performed by each vs. time. It shows quite well that the computational burden of XeriScape is much less than that of a discrete time simulation such as Sugarscape. This is in spite of the fact that in Experiment 6 each cell that has agents is cycling every time step. The efficiency gain is entirely due to the fact that dormant cells don’t require any computational effort. Experiment 8 shows an even greater performance increase due to its variable time step. In the first 30 time units of simulation for Experiment 8, only 3,332 cumulative active cells existed. In the corresponding discrete time simulation the number would be 30 time units $\times$ 2,500 cells, or 75,000. The DEVS-Java implementation is over 22 times faster! Furthermore, as the calculation above shows, the larger the environment the greater the performance increase is likely to be. DEVS-Java is a high-performance simulation environment.

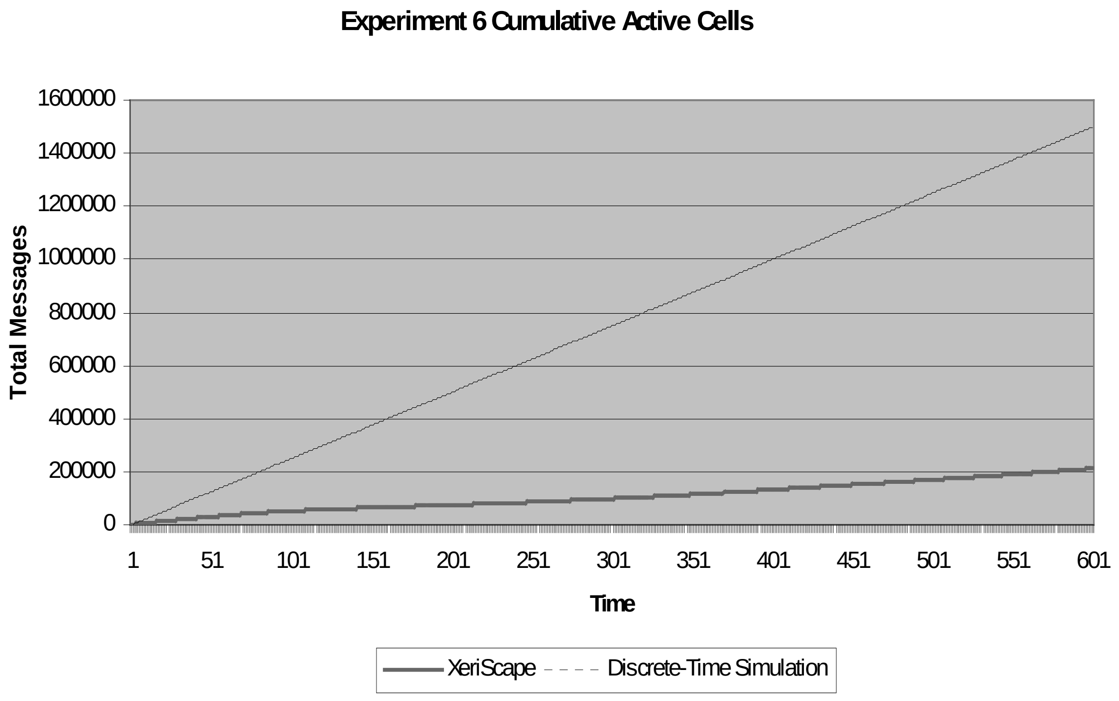


Figure 65. Experiment 6 Cumulative Active Cells

Can a DEVS implementation answer other social science questions quickly and easily? While the question is not directly answered by the research, in looking at the results the answer is affirmative. DEVS is a broad-based tool that can easily be applied to other social science questions besides artificial societies. The general concepts explored in XeriScape can be readily generalized to other kinds of problems. For example, the cellular automaton developed as the environment for XeriScape could itself be the basis for experimentation in environmental studies such as pollution generation and diffusion. The agents developed for XeriScape could be the basis for a stand-alone multiple agent simulation. Since much of social science research deals with interactions among individuals (i.e., events), any bottom-up approach to social science simulation can benefit from discrete event simulation.

Can a DEVS artificial society simulation provide a more general and extensible foundation for further research? The object-oriented nature of Java and the general design of XeriScape make it easy to develop new rules and new types of agents and insert them into the simulation environment. The inheritance property of Java means that new agents and new rules can inherit most of their code from previous work, only writing the new code that differentiates them from their predecessors. An examination of the agent code for XeriScape, for example, shows that a great deal of code for the more sophisticated agents (GenderedAgent, for example) is inherited from the base Agent class.

Can a DEVS artificial society provide a feedback mechanism that allows agents to grow and learn, and to pass on what they have learned to their offspring? This has been demonstrated in XeriScape. While it is true that there is no explicit learning mechanism as such incorporated in XeriScape, the genetic selection seen in Experiment 6 shows that agents can pass their properties on to their offspring in such a way that they improve on their ancestors. Over the course of time the society as a whole learns to be more efficient.

6.2 General Conclusions

In the course of this work, a number of general conclusions about social science simulations and DEVS-Java presented themselves.

Probably the most obvious conclusion is that the Sugarscape approach is fundamentally a discrete time oriented simulation. In adapting the concept to a discrete event orientation, some awkwardness arose in translating between the two. Thus, Experiments 1 through 7 are not as efficient as they might be if they were approached more as a discrete event system. Experiment 8 shows that the discrete event simulation can work for an artificial society.

A second conclusion is that there are many ways to design a simulation, both architecturally (its structure) and philosophically. In the course of development, several observers suggested that it would have seemed more "natural" to make everything in the simulation a DEVS model, especially the agents themselves. In this approach, a variable structure DEVS would have been required in order for the agents to move between cells. While this approach would be worth exploring, it seemed to lack a certain transparency and simplicity of design. In order to accommodate the static structure it became necessary to implement the agent life cycle in the cell and not in the agent itself. This proved both a help and a hindrance in certain respects: it made the cells flexible and able to deal with all the different agent types, but it also made the cell more "aware" of the agent life cycle than seems prudent.

Finally, the XeriScape project showed that the worlds of social science and simulation science are not well connected, and there is little communication between the two. Technology used by social scientists has not benefited from the extensive research in modeling and simulation of the last few years, in particular efforts at massively parallel, collaborative, and distributed simulation. In order for social science to take advantage of all that is available in the best of simulation science, more collaboration is needed. Social scientists are not simulation experts. Similarly, simulation scientists are not social scientists, and their favorite problems do not usually intrude into the social science problem space. There is a large, fruitful area to be explored in creating more common ground between the two groups. Such efforts would undoubtedly prove helpful to both and to science in general.

6.3 Future Directions

There are two main directions for future research using the XeriScape system. The first is extension of the model via new components: new agents, rules, etc. The second direction involves enhancements and extensions of the capabilities of the system itself.

6.3.1 New XeriScape Components

Epstein and Axtell’s work provides a rich environment for additional XeriScape components. Some possibilities worth exploring in future versions include:

- Merchant agents that trade resources on the XeriScape
- Tribal allegiances or nation-states that compete for resources and space and wage war to get them
- Diseases that infect agents and spread throughout the XeriScape
- Agents that use cloning for reproduction
- Agents that limit their reproduction

- Agents that develop a tolerance for pollution

- Agents that mate for life

- Rules that select for offspring of a particular sex (e.g., female infanticide)

6.3.2 Enhanced System Capabilities

During the course of software development and experimentation, many possibilities for new capabilities for XeriScape presented themselves. Some of them include:

- Distributed processing capability – the ability to use multiple processors to run larger simulations, and to run simulations more quickly

- Enhanced graphical interface – the ability to click on a cell and inspect its contents, reset parameters, drag-and-drop agents from cell to cell, etc.

- Enhanced statistics processing – the ability to display statistics (mean agent vision, mean metabolic rate, Gini coefficients, etc.) as the simulation runs

- Flexible neighborhoods – the ability to use Moore neighborhoods in addition to von Neumann neighborhoods

- Agent clustering – allowing multiple agents to inhabit the same cell

- Variable density – allowing the number of agents per cell to vary, either statically or dynamically

APPENDIX A XERISCAPE SOFTWARE

The XeriScape software and accompanying documentation may be obtained from http://www.zaft.org/gordon/XeriScape/index.htm. Note that the software described in this thesis requires a license for DEVS-Java, which can be obtained from the Arizona Center for Integrative Modeling and Simulation (ACIMS) at http://www.acims.arizona.edu.

APPENDIX B XERISCAPE CONFIGURATION FILE

The XeriScape configuration files store the setup for a specific simulation run. The XeriScape configuration file is a Java properties file stored in text format. The configuration file for Experiment 3 is shown below as an example. Configuration files can be loaded from the XeriScape GUI or specified on the command line when the simulation is run.

```
# XeriScape Experiment 3
# Copyright (c) 2001 Gordon C. Zaft
# $Id: Experiment3.cfg,v 1.5 2001/07/22 04:13:49 zaft Exp $
rows=50
cols=50
name=Experiment 3 - Finite Agent Lifetimes with Replacement
replace=true
growbackrate=1.0
rescomprule=SimpleResourceComparisonRule
resinitrule=TwinPeaksResourceInitRule
agentinitrule=FiniteAgentInitRule
agentdistrule=RandomAgentDistRule
numagents=125
adrseed=17171717
airseed=77777777
```

Configuration file lines that begin with “#” are comments and are ignored by XeriScape. Blank lines are also ignored. Non-comment lines are key-value pairs; the keys and their values are described in Table 2 below. Configuration files may be created and edited manually, or they can be created by using the File, New command from the XeriScape GUI and saved using File, Save.

Key	Value
rows	Number of rows in the simulation (integer).
cols	Number of columns in the simulation (integer).
name	Title of the simulation as it should appear in the title bar.
replace	“true” turns replacement mode on (see section 4.13). If the parameter is “false,” replacement is off. If replace is omitted, the default is for replacement off.
growbackrate	Number of units to grow back per unit time period (floating point value). If growbackrate is not specified, the default is an infinite growback rate.
rescomprule	The name of the resource comparison rule.
resinitrule	The name of the resource initialization rule.
agentinitrule	The name of the agent initialization rule.
agentdistrule	The name of the agent distribution rule.
numagents	The number of agents for the simulation (integer).
adrseed	The random number generator seed value for the agent distribution rule. If adrseed is not specified, the random number generator is seeded from the system clock.
airseed	The random number generator seed value for the agent distribution rule. If airseed is not specified, the random number generator is seeded from the system clock.

Table 2. Configuration File Entries

XeriScape assumes that all rules are in the package XeriScape.Rules. XeriScape uses Java’s reflection capability to create instances of the rules specified in the configuration file so that they may be mixed and matched without recompilation, that is, at runtime. XeriScape allows specification of the random number generator seeds for the agent distribution rule and agent initialization rule so that the same sequence of random numbers can be generated from run to run, resulting in consistently repeatable results.

APPENDIX C COMPARISON OF XERISCAPE AND SUGARSCAPE

While XeriScape is intended to be an implementation and adaptation of the Sugarscape concept to the DEVS formalism and DEVS-Java technology, it is not a perfect representation of Sugarscape. This appendix details some of the differences.

C.1 Von Neumann vs. Moore Neighborhoods

While Epstein and Axtell report on Sugarscape only as a von Neumann neighborhood system, in passing they claim that Sugarscape can run as either a von Neumann or Moore neighborhood system.

XeriScape runs only as a von Neumann neighborhood system.

C.2 XeriScape Topology

While the XeriScape topology is designed to be an approximation of the Sugarscape topology, a close examination of Epstein and Axtell shows resources dispersed more diffusely than in the XeriScape “Twin Peaks” layout.

C.3 Agent Reproductive Frequency

In the Sugarscape implementation, agents can mate multiple times in a single cycle if they have the resources and opportunity. The XeriScape implementation allows agents to mate only once per cycle. Also, there is currently no rule for selecting the best possible mate; the first available mate is chosen.

C.4 Pollution Generation/Diffusion Immediate

In Sugarscape, there is a delay factor before pollution is generated or diffused. XeriScape does not insert a delay, but immediately begins generating and diffusing pollution.

C.5 Multiple Agents in a Cell

In Sugarscape each cell could have only one agent at a time. XeriScape allows an arbitrary number of agents in a cell. The number of agents per cell (i.e. number of slots, see section 4.8) is determined at the time the cell is created. It would be possible to create another rule that would vary the number of slots in each cell. It is also possible to vary the number of slots while the simulation is running.

REFERENCES

[1] Zeigler, B.P., *Theory of Modeling and Simulation.* New York: John Wiley, 1976.

[2] Epstein, J.M., and R. Axtell. *Growing Artificial Societies: Social Science from the Bottom Up*. Washington, D.C.: Brookings Institution Press/Cambridge, Mass.: MIT Press, 1996.

[3] Pournelle, G., "Chaos Manor: Of Supercomputers, Sound Files and Sugarscape," *Byte*, June 1997.

[4] Peterson, I., "The Gods of Sugarscape," *Science News,* November 23, 1996.

[5] Gardner, M., "The Fantastic Combinations of John Conway's New Solitaire Game 'Life'," *Scientific American* 23(4), October 1970, pp. 120-123.

[6] Werner, G. M., and M. G. Dyer, "Bioland: A Massively Parallel Simulation Environment for Evolving Distributed Forms of Intelligent Behavior," *Massively Parallel Artificial Intelligence*, edited by H. Kitano and J. A. Handler. Menlo Park, Calif.: AAAI Press/MIT Press, 1994.

[7] Nowak, A. and B. Latané, "Simulating the emergence of social order from individual behavior," *Simulating Societies*, edited by N. Gilbert and J. Doran. Guildford, England: UCL Press, 1994.

[8] Drogoul, A. and J. Ferber, "Multi-agent simulation as a tool for studying emergent processes in societies," *Simulating Societies*, edited by N. Gilbert and J. Doran. Guildford, England: UCL Press, 1994.

[9] Doran, J., "Questions in the Methodology of Artificial Societies," *Tools and Techniques for Social Science Simulation*, edited by R. Suleiman, K. Troitzsch and N. Gilbert. Heidelberg: Physica-Verlag, 2000.

[10] Gilbert, N. "Models, Processes and Algorithms: Towards a Simulation Toolkit," *Tools and Techniques for Social Science Simulation*, edited by R. Suleiman, K. Troitzsch and N. Gilbert. Heidelberg: Physica-Verlag, 2000.

[11] Hiebler, D. "The Swarm Simulation System and Individual-based Modeling," *Decision Support 2001: Advanced Technology for Natural Resource Management*. Toronto, Sept. 1994.

[12] Minar, N. and R. Burkhart, C. Langton, M. Askenazi. "The Swarm Simulation System: A Toolkit for Building Multi-Agent Simulations," http://www.swarm.org, 1996.

[13] Balzer, W., "SMASS: A Sequential Multi-Agent System for Social Simulation," *Tools and Techniques for Social Science Simulation*, edited by R. Suleiman, K. Troitzsch and N. Gilbert. Heidelberg: Physica-Verlag, 2000.

[14] Urban, C., "PECS: A Reference Model for the Simulation of Multi-Agent Systems," *Tools and Techniques for Social Science Simulation*, edited by R. Suleiman, K. Troitzsch and N. Gilbert. Heidelberg: Physica-Verlag, 2000.

[15] Rockloff, M., "Bedrock: Building Multi-Agent Simulation Applications," *Tools and Techniques for Social Science Simulation*, edited by R. Suleiman, K. Troitzsch and N. Gilbert. Heidelberg: Physica-Verlag, 2000.

[16] von Neumann, John, *Theory of Self-Reproducing Automata*. Edited by A. W. Burks. Urbana, IL: University of Illinois Press, 1966.

[17] Wolfram, S., *Cellular Automata and Complexity: Collected Papers*. Addison-Wesley, 1994.

[18] Zeigler, B.P., H. Praehofer and T.G. Kim, *Theory of Modeling and Simulation*, 2nd edition. Academic Press, 2000.

[19] Barros, F.J., "Dynamic Structure Discrete Event System Specification: A New Formalism for Dynamic Structure Modeling and Simulation," *Proceedings of the 1995 Winter Simulation Conference*. Arlington, VA: 1995.

[20] Barros, F. J., "Representation of Dynamic Structure Discrete Event Models: A Systems Theory Approach," *Discrete Event Modeling and Simulation Technologies: A Tapestry of Systems and AI-Based Theories and Methodologies*, edited by H. Sarjoughian and F. Cellier. New York: Springer-Verlag, 2001.

[21] Wainer, G. and N. Giambiasi, “Timed Cell-DEVS: Modeling and Simulation of Cell Spaces,” *Discrete Event Modeling and Simulation Technologies: A Tapestry of Systems and AI-Based Theories and Methodologies*, edited by H. Sarjoughian and F. Cellier. New York: Springer-Verlag, 2001.

[22] Kakwani, N., “Lorenz Curve,” *The New Palgrave Dictionary of Economics*, edited by J. Eatwell, M. Milgate, and P. Newman. New York: Stockton Press, 1990.

[23] Dagum, C., “Gini Ratio,” *The New Palgrave Dictionary of Economics*, edited by J. Eatwell, M. Milgate, and P. Newman. New York: Stockton Press, 1990.