Simulation and Visualization of Distributed Artificial Life Scenarios

Michael Syrjakow[*], Jörg Berdux[*], Dietmar Püttmann[*], Helena Szczerbicka[**]

[*]*Institute for Computer Design and Fault Tolerance (Prof. D. Schmid)*
University of Karlsruhe, 76128 Karlsruhe, Germany
[**]*Institute for Computer Science, Fachbereich Mathematik und Informatik*
University of Hanover, 30167 Hanover, Germany

Abstract

The Artificial Life (AL) simulation described in this paper offers the possibility to observe how primitive artificial organisms are forming rather complex communities while evolving and adapting themselves to their environment. For that purpose our simulation provides an intuitive graphical user interface which allows to start/stop/continue simulation runs, vary the animation speed, observe the evolving population as a whole (macro view), observe the life cycle of single individuals (micro view), interactively change environmental parameters (food growth rate, life expectancy of the individuals, etc.) during a simulation run. A specific feature of our application is that it provides the possibility to study the role of cooperative behaviour within populations of autonomous agents. Beside this, it can be easily distributed on several computers allowing to simulate also very complex artificial worlds. In our paper we give a detailed description of the distributed architecture and the functionality of our AL simulation which is realized platform independently as a Java application. Beyond that, some interesting experimental results are presented and an outlook is given on our future work.

Introduction

Artificial Life (AL) is a rapidly growing field of scientific research linking biology, computer science, and engineering. According to the definition of C.G. Langton, who deeply influenced this field, AL is devoted to understanding life by attempting to abstract the fundamental dynamical principles underlying biological phenomena, and recreating these dynamics in other physical media - such as computers - making them accessible to new kinds of experimental manipulation and testing [1]. The fundamental algorithms of AL are learning algorithms [2,3] (typified by Neural Networks [4]), evolutionary algorithms [5] (typified by Genetic Algorithms [6]), and cellular automata [7]. AL also very intensively deals with building adaptive agents living in complex dynamic environments where they have to act autonomously in order to reach certain goals [8]. In this paper such an artificial ecosystem is described. The main characteristics of this AL simulation are the following:

- demanding functionality in the sense that also complex social behaviour is taken into account
- comfortable graphical user interface to ensure an easy usage also by non-experts
- various statistics allowing a comprehensive analysis of performed experiments
- scalability allowing to simulate also very complex AL scenarios
- platform independent realization in Java

We start with a detailed description of the artificial ecosystem and the individuals living in it. Subsequently, we demonstrate some realization details and explain how our AL simulation can be distributed on several computers. In order to show the capabilities of our AL simulation some interesting experimental results are presented. Finally, we summarize and draw some conclusions.

Description of the Artificial Ecosystem

Fig. 1 shows the main window of the graphical user interface of our AL simulation. On the left side the artificial ecosystem is presented, being a rectangular world where four kinds of objects can appear: grass, which is growing with a user-specified grass growth rate, barriers (hills) which are randomly placed within the world during the initialisation phase, and finally two kinds of autonomous agents able to move within the ecosystem. The artificial world can be viewed as a projection of a torus on a rectangle. That means that agents crossing a border of the rectangular world immediately appear on the opposite border. As already mentioned the ecosystem comprises two kinds of agents: primitive and advanced agents. The primitive agents eat grass which is growing in the ecosystem with a certain user-defined growth rate. The advanced agents in turn live on the primitive agents. Thus, a small food chain is established which is however very difficult to keep in balance. In the following the primitive agents are called quarries and the more advanced agents are referred to as hunters.

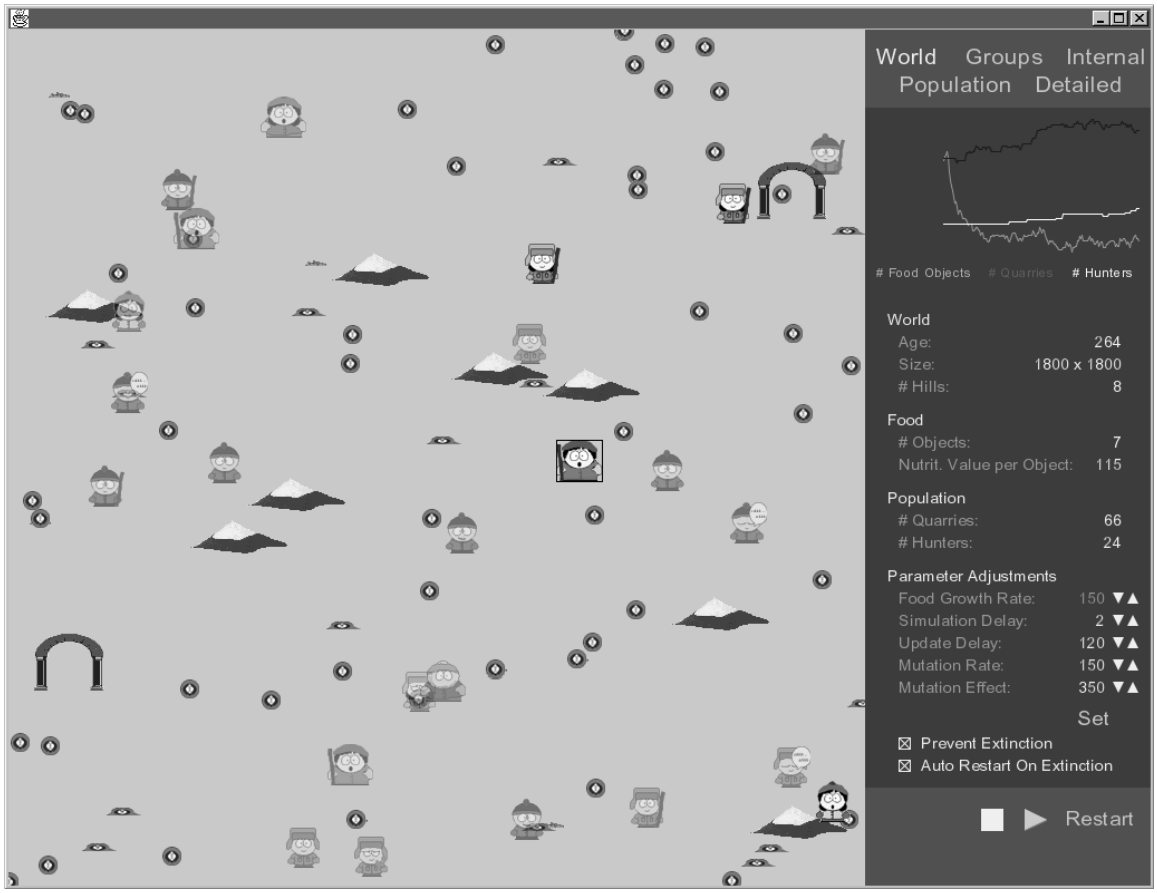

Fig. 1. Graphical user interface of the AL simulation (main window)

The overall goal of the individuals living in the artificial ecosystem is to get as old as possible and beyond that, to reproduce themselves as often as possible. For that purpose agents are first of all looking for food and secondarily for reproduction. In particular, agents can carry out the following basic actions:

- look around in order to generate a local map of their environment,
- move in order to change their position within the ecosystem,
- eat in order to increase their energy level,
- sleep in order to save energy and renew their stamina level (only hunters),
- reproduce in order to pass on genetic information to subsequent generations.

Beyond that, the more advanced hunters have some social behaviour. They are able to build groups in order to increase their chance to successfully hunt for primitive agents. A special hunter with group leader capability leads a group. To recruit new group members a communication mechanism is applied which is based on the agent communication language KQML (Knowledge Query and Manipulation Language) [9]. KQML is a language and protocol for exchanging information and knowledge. It focuses on an extensible set of performatives, which defines the permissible operations that agents may attempt on each other's knowledge and goal stores. The performatives comprise a substrate on which to develop higher-level models of inter-agent interaction such as contract nets and negotiation. Beyond recruitment and resignation of group members the communication mechanism is also used to find sexual partners for reproduction. When two hunters reproduce the resulting child inherits a mixture of the parental capabilities, which are additionally modified by a probabilistic mutation operator.

When the simulation is started a population of hunters and quarries as well as some barriers are generated. Beside this the grass starts growing with a user-defined grass growth rate. All the generated objects are distributed randomly within the ecosystems. After initialisation the state of each object is updated cyclically. When an update cycle is over the world time is increased and the next update cycle begins.

In the following the two kinds of autonomous agents are described more detailed. We start with the quarries, which can be characterized by the following attributes:

1) age
2) maximum age (when a quarry reaches its maximum age it dies)
3) speed
4) range of vision
5) information update rate (rate for updating the environmental information within the range of vision)
6) health level (can be decreased by hunters with attacker capability; when the health level has been reached zero the quarry dies)
7) health regeneration rate (rate for incrementing the health level)
8) reproduction rate (determines the frequency of reproduction)
9) accumulated food
10) energy level (nutritional value of the quarry for the hunters)

Many of the attributes described above are dynamic and may change in time. For example when a quarry eats some grass the accumulated food attribute has to be increased. The next action of a quarry as well as the corresponding attribute updates are computed cyclically by the simulation engine. In each update cycle the following operations are performed on a quarry:

- transform some accumulated food into energy (each action requires some portion of energy)
- when it is time for an information update (determined by the information update rate) update the environmental information by looking around within the range of vision
- when it is time for health regeneration (determined by the health regeneration rate) increase the health level
- if a food target has been determined and this food target is not yet within reach and no hunter is within a certain range then move towards the food target
- if a food target has been determined and this food target is within reach and no hunter is within a certain range then consume a certain part of the food target (in this case the food target remains the same until it is completely consumed)
- if a food target has been determined and a hunter is within a certain range then give up the food target and move on to escape
- if no food target has been determined then use the environmental information to determine one
- if no food target could be found within the range of vision then move on in a randomly chosen direction
- if it is time for reproduction and there are not too many other quarries within the range of vision then create a copy of the quarry, mutate the attributes 2, 3,..., 8 of the copy, set its age to zero, halve accumulated food and energy level of both the original quarry and its copy, and finally determine the next time for reproduction

- if the food level has been reached zero then decrease the health level
- if the health level has been reached zero or the maximum age has been exceeded then the quarry dies
- if the quarry is dead then decrease its remaining energy level
- if the energy level of a dead quarry has been reached zero then remove the quarry completely from the ecosystem

Now the attributes of the more complex hunters who live on the quarries are described. These attributes are the following:

1) age
2) age of retirement (when this age has been reached the attributes 5, 6, 7, and 8 are gradually decreased causing finally the death of the hunter)
3) sex (male, female)
4) special capability (attacker, paralyser, group leader)
5) speed
6) range of vision
7) information update rate (rate for updating the environmental information within the range of vision)
8) accumulated food
9) degree of saturation (only when the accumulated food exceeds the degree of saturation a hunter is willing to reproduce himself)
10) urge for reproduction (determines the frequency of reproduction when a hunter is in the state ready for reproduction)
11) stamina (the lower the stamina is the more the agent has to sleep to regenerate it)

Hunters can be divided into attackers, paralysers, and group leaders. In the following the specific capabilities of these three types of hunters are described more detailed:

- Group leaders
  A group leader cannot actively hunt by himself. In order to get food he has to recruit a group consisting of attackers and paralysers. After recruitment of a group he coordinates the hunting of quarries. In case of a successful chase he is also responsible for the distribution of the captured food among his group.
- Paralysers and attackers
  Paralysers have the special capability to decrease the speed of quarries being located in their neighbourhood. Paralysed quarries in turn can be easily killed by attackers who have the special capability to decrease the health level of quarries. Quarries with a health level of zero die and their remaining energy can be consumed by the hunters.

Because of their complementary capabilities hunters are forced to build groups in order to survive for the longer term. Alone they are not very successful and die when they don't find some food left by other hunters. Altogether hunters have a much more complex behaviour than quarries depending on their current needs. The following list comprises the possible needs a hunter can have and their priorities:

- need to sleep
  This need has highest priority. A hunter needs to sleep when his stamina level has fallen below a certain value. Through sleeping the hunter can recover his stamina level.
- need to eat
  This need has medium priority. A hunter needs to eat when his degree of saturation has not yet been reached. Through consuming the energy of dead quarries a hunter can increase his accumulated food.
- need to recruit/join a group
  This is a consequential need of the previous need. If a hunter without group cannot satisfy the need to eat for some time this need is changed into the need to recruit/join a group. This is done because the most promising way for a hunter to get food (i.e. to satisfy his need to eat) is to hunt cooperatively in a group.

- need to reproduce
  This is the need with lowest priority. A hunter feels an urge for reproduction when his degree of saturation has been reached and he had enough sleep.
- no need
  A hunter has no need when all the other needs described above are satisfied. Then he sleeps in order to save energy and to increase his stamina.

In each simulation update cycle first the age of the hunter is increased and the accumulated food as well as the stamina level are decreased. Then the current need of a hunter is determined. This is done by checking the current values of his attributes (accumulated food, stamina, urge for reproduction, etc.). If a hunter has several needs at the same time the need with the highest priority is chosen. After the current need has been determined a more or less complex sequence of actions is initiated in order to satisfy it. Each performed action may affect the dynamic attributes of the hunter which in turn may cause a new need. If the current need remains the same in the following update cycle the next action of the corresponding action sequence is carried out. When a need with higher priority is determined the hunter changes his plans and starts another sequence of actions in order to satisfy his new need. In the following some exemplary situations (needs and the resulting activities) are described:

1) If the current need of a hunter is to sleep (dependent on his current stamina) the next action is to sleep for some time.
2) If the hunter is an attacker and not attached to a group and his current need is to eat he first looks for quarries in his range of view. If he finds a dead quarry he tries to get some energy of this quarry. If he finds a living quarry he starts to attack this quarry. If his efforts to get some food have not been successful for some time the current need of the hunter changes into the need to join a group.
3) If the hunter is an attacker or paralyser and his current need is to join a group he first looks for a group leader in his environment. If there is no group leader available he makes a random move. Otherwise he sends the group leader an application for joining the group. This application contains some important attributes of the hunter (capability, age, speed, etc.) In case of a successful application the hunter joins the group. Otherwise he tries to find another group.
4) If the hunter is a group leader and his current need is to recruit a group and the preferred number of group members is not exceeded considerably he first determines what kind of capability is needed most in his group. If attackers are required the group leader checks whether there are any attackers in his environment (the hunters within reach are asked about their capabilities). If this is not the case a random move is made. Otherwise the available attacker is asked to join the group. If a group leader is not content with the capabilities of an existing group member he can also replace this member by a hunter with better capabilities.
5) If the hunter is a group leader and his current need is to eat then he initiates a hunt. In order to determine a quarry he first checks his own environmental information. If there is no quarry available he asks his group members to send him their environmental information. If there is still no quarry available every group member makes a random move. Otherwise a quarry is chosen as the target and the paralysers get the command (commands have higher priority than all needs except the need to sleep) to paralyse it. If the quarry has become slow enough the attackers get the command to attack the quarry in order to kill it. When the quarry is dead the group leader distributes the remaining energy among the members of his group.
6) If the hunter is an attacker or paralyser and he is member of a group and he needs to eat but the group leader is not hunting for a longer time he tries to apply to another group.
7) If the hunter is a male hunter and his current need is to reproduce himself then he first sends out a request for reproduction to the female hunters in his environment. Subsequently the positive answers (if any) are checked and one of the corresponding females is chosen. If the number of hunters in the environment of the two mating hunters is below a certain value a child is generated. The age of this child is set to zero, sex and capability are chosen randomly, and for the remaining attributes mutations of the mean values of the parental attributes are computed.

The ability of the advanced agents to build groups is a very distinguishing feature of our AL simulation. In order to allow an observation of the ongoing dynamic group building processes an appropriate visualiza-

tion mechanism is of great importance. For that purpose the graphical user interface of our AL simulation provides the following possibilities:

1. When a hunter is selected (by a mouse click) all his group members (if any) are presented opaquely, whereas the rest of the population is presented transparently. This way it is possible to observe one special group.
2. Additionally the user can activate a transparent polygon indication covering all group members in order to indicate the spatial dimensions of the group.
3. It is also possible to indicate all existing groups by transparent polygons. An example of this possibility is shown in Fig. 2.

In order to distinguish sex and kind of the hunters different types of manikins are used. Fat manikins represent group leaders. Thin manikins may be attackers or paralysers. A certain state (eating, sleeping, hunting, reproducing, communicating, etc.) of a hunter is symbolised by a special figure with the corresponding gesture. A manikin with a stick for example represents a hunter who is currently hunting. A killed quarry is indicated by a flat version of the round figure representing a living quarry.

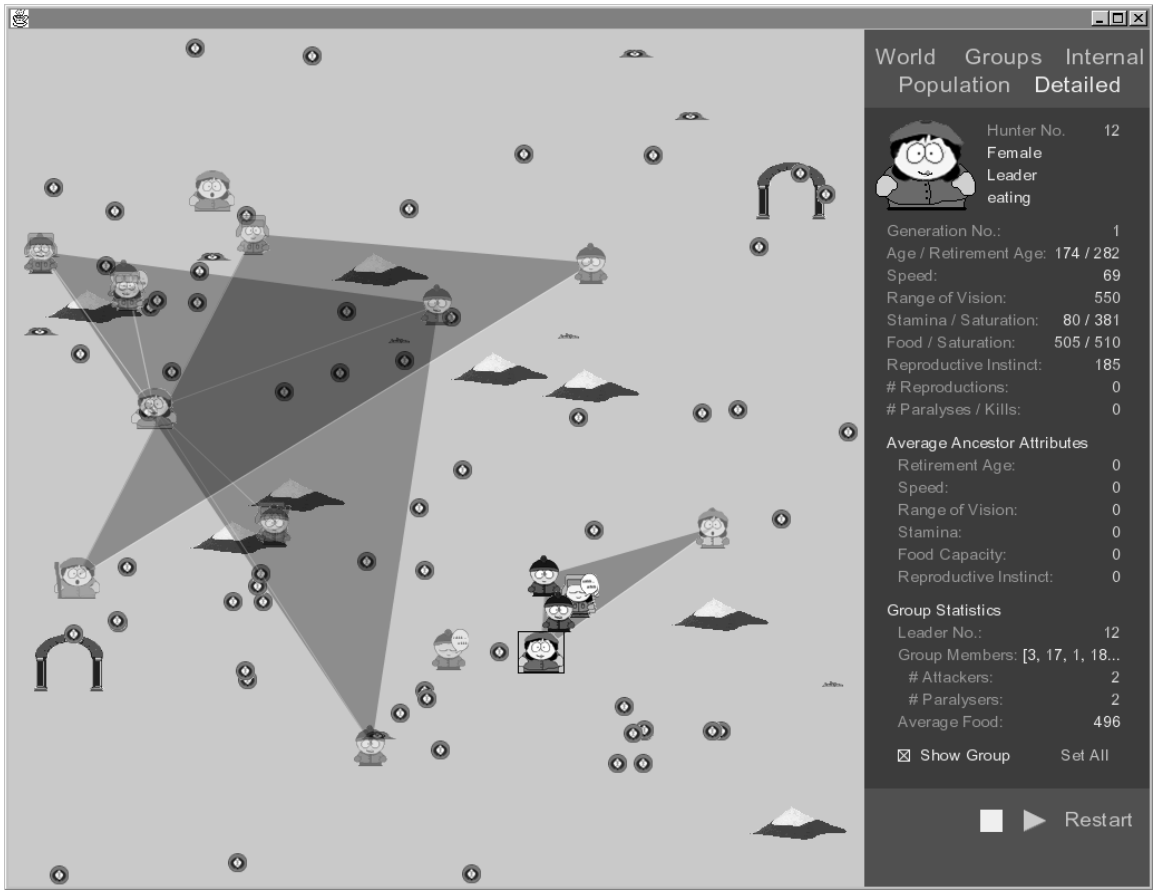

Fig. 2. Visualization of all hunter groups existing in the artificial ecosystem

Realization and Distribution of the AL Simulation

As already mentioned we used Java for implementation of our AL simulation. Decisive for this decision were the following advantageous properties of Java:

- object-orientation and platform independent execution
- extensive API for all kinds of programming (distributed programming, window programming, etc.)

- Applet concept (at the moment our AL simulation is realized as a Java application but we intend to also release an Applet version which allows the user to access the simulation over the Internet [10])

Although Java programs run slower than programs written in compiled languages like C and C++ our application has a quite good execution speed on standard PCs of today (500 MHz processor, 128 MByte RAM). In order to simulate also very complex ecosystems with a high number of agents we have developed a flexible and easy-to-use distribution mechanism for our AL simulation. To realize a distributed artificial world each ecosystem is able to provide portals to other ecosystems running on the same or other computers. When the current ecosystem isn't attractive for an agent any more because he cannot find food or social contact he can use such a portal to change the ecosystem. To keep the communication effort between linked ecosystems low agents cannot contact agents from other ecosystems. Beyond that, an agent is administered and known exclusively by the ecosystem where he is currently living in. When an agent changes into another ecosystem he is completely removed from the current ecosystem.

Fig. 3 shows a possible distributed scenario of our AL simulation. In this example several ecosystems are linked by portals. When a portal is in the range of view of a hunter, the hunter checks whether he is discontented with his current situation. If this should be the case he uses the portal to change to the linked ecosystem.

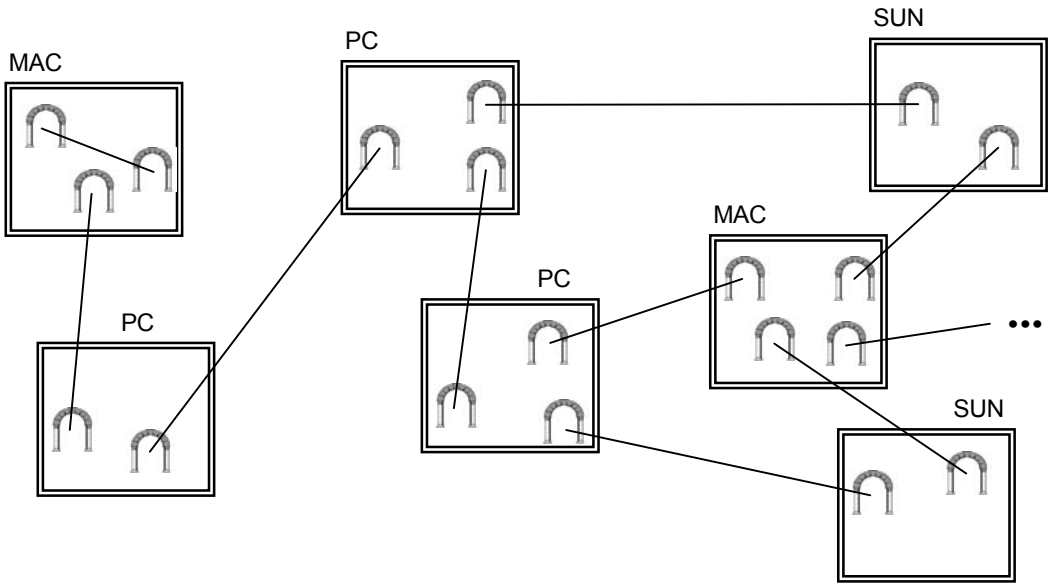

Fig. 3. Example of a distributed artificial world consisting of several linked ecosystems

Experimental Results

To allow an easy handling our AL simulation provides a comfortable and intuitive graphical user interface. In Fig. 1 and 2 important parts of this GUI are presented. The big rectangle on the left provides a macro view of the entire artificial ecosystem which allows to observe the evolving agent populations as a whole. For that purpose all existing static objects (barriers, grass, portals) as well as the existing dynamic objects (quarries and hunters) are shown. The tabbed pane on the right side allows the user to get statistical information about the evolving populations and their individuals. Beyond that, it allows to adjust the control parameters of the simulation which can be roughly divided into

- world parameters ("World" tab)
  - grass growth rate
  - mutation rate
  - mutation effect (allows to vary the implications of a mutation)

- simulation parameters ("World" tab)
  - simulation delay (delay between two simulation update cycles)
  - graphical update delay (allows to vary the time between two graphical updates)
- ranges of internal attribute values of quarries and hunters ("Internal" tab)

Beside the macro view our simulation also provides a detailed micro view allowing to observe the life cycle of single individuals. The micro view can be activated by clicking on the "Detailed" tab. An example is shown in Fig. 2. Here detailed information (sex, capability, current need, age, speed, range of vision, etc.) about the currently selected hunter (in this case a female group leader) is presented. Beside the micro view the tabbed pain also provides detailed statistics about the quarry and hunter populations ("Population" tab) as a whole and statistics about the group building process among the hunters ("Groups" tab).

Our AL simulation supports the following kinds of experiments:

1. Experiments where the initial control parameter setting remains fixed during the whole simulation run. Here the user starts the simulation with an initial control parameter setting and then passively observes what happens. One possible objective of such a static experiment might be to find an initial set of parameter values which causes a balanced ecosystem where no species becomes extinct for a prolonged period of simulation time.
2. Experiments where the control parameter setting is dynamically changed (by the user or a special program) during a simulation run. In this case the user also starts the simulation with an initial control parameter setting but then actively changes this setting in order to observe the consequences. One possible goal of such an interactive experiment might be to intervene always then when the established food chain is threatened to be thrown off balance (the number of individuals of a species falls below a certain value).

In the following the results of two simulation experiments are presented. In the first experiment the initial control parameter setting was never changed during the simulation run. Diagram 1 shows some statistical data obtained from this experiment where about 50 hunter generations were computed.

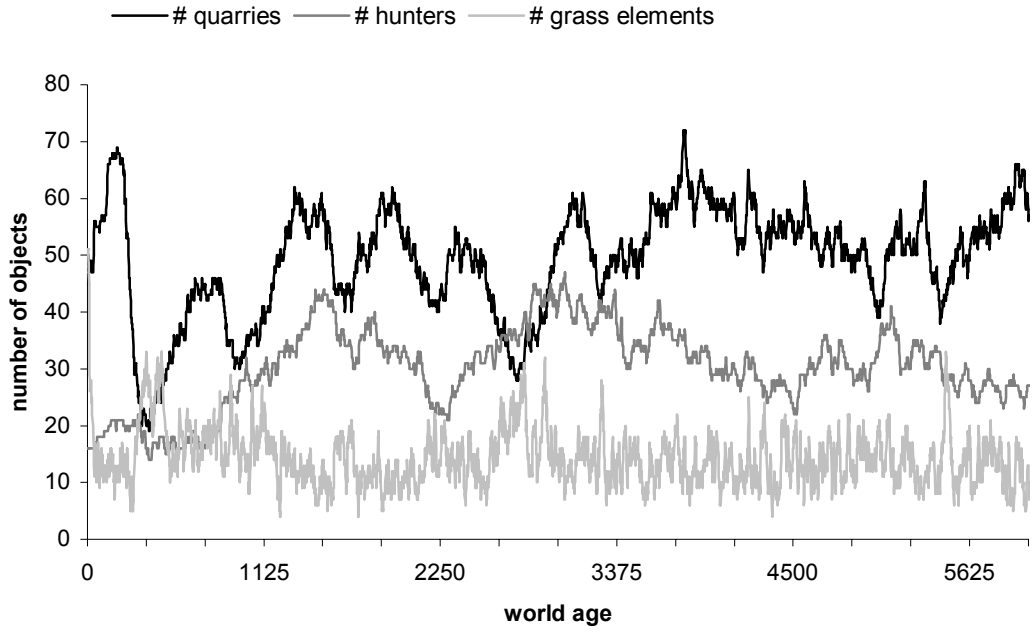

Diagram 1. Results of a simulation experiment without user interactions

The three curves show how the number of grass elements, the number of quarries, and the number of hunters developed over the simulated time. At the beginning of the warm-up phase which covers approximately the first thousand simulated time units a drastic decline of the grass elements and a considerable increase of the quarries can be observed. This is mainly caused by the initial group building processes of the hunters, which have to be finished before the chase after the quarries can really start. After the hunters have been grouped together the number of quarries decreases enormously. This in turn causes an increase of the food elements and also a decreasing number of hunters. Some time later the quarry population recovers again causing in turn an increase of the hunters. Approximately at world age 1000 the warm-up phase has been finished and the three curves start to oscillate within a relative stable range.

The primary goal of this experiment was to show that it is possible to adjust the parameters of the ecosystem that way, that the established food chain keeps quite balanced. This however has been proven to be very difficult. We had to experiment a lot before we could achieve the results shown in Diagram 1 where no agent species dies out for the longer term. We assume that the main reason for this is the limited size of the ecosystem. When several ecosystems are linked together it has been proven to be much easier to get a stable system.

Diagram 2 shows the results of an interactive simulation experiment where about 70 hunter generations were computed.

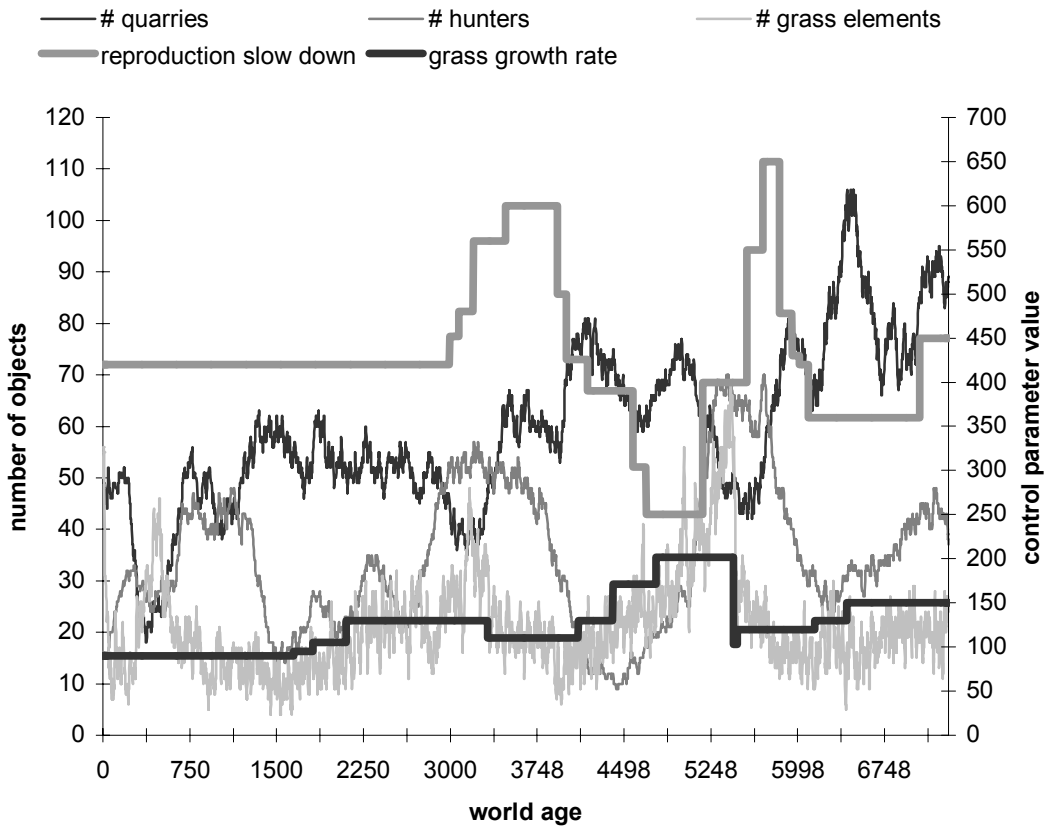

Diagram 2. Results of an interactive simulation experiment

In this experiment the control parameters "reproduction slow down" and "grass growth rate" were changed dynamically during the simulation run in order to keep the number of quarries and hunters within reasonable ranges. Here also a warm-up phase can be observed covering approximately the first 700 simulated time units. After world age 1100 the number of hunters declines drastically. As a countermeasure the grass

growth rate was increased in order to raise the number as well as the energy level of the quarries. As the result the hunters find more quarries with a higher nutritional value and consequently their population quickly recovers. At world age 2900 the number of hunters exceeded the number of quarries and we decided to stop that growth by increasing the parameter "reproduction slow down" which decreases the urge for reproduction of the hunters. This measure results in a drastic reduction of the hunter population which is stopped in the following by a decrease of the reproduction slow down parameter in combination with a further increase of the grass growth rate.

Summing up, it has been proven to be quite difficult to preserve a stable and predictable ecosystem. This is mainly caused by the limited size of the ecosystem but also by the various and sometimes interdependent attributes of the competing agents. Especially an appropriate adjustment of the parameters for controlling the social relations between the hunters requires some experience. The most critical dependencies exist between the reproduction rates of the two species and the global grass growth rate. If the hunter reproduction rate is chosen too high all quarries are killed within a very short time. In order to prevent such a sudden extinction of the quarries the simulation also offers the possibility to add regularly some externally generated quarries into the artificial world.

Also very interesting to observe is how the capabilities of the agents evolve over the generated populations. In the experiments described above speed and range of vision of both quarries and hunters improved considerably over time. However, in order to make more sound statements about this evolutionary development long-term studies with larger ecosystems are required. This will be a topic of our future work which is outlined more detailed in the following section.

Conclusions

In this paper an AL simulation was described allowing to observe how two species of primitive artificial organisms evolve and adapt themselves to a dynamic environment. The most distinguishing characteristics of this simulation are

- modelling of cooperative behaviour
- flexible and easy-to-use distribution concept allowing unrestricted scalability
- comfortable and intuitive graphical user interface providing macro and micro views
- various statistics allowing a comprehensive analysis of the performed experiments
- platform independent realization in Java

Making experiments with our AL simulation has been proven to be very interesting and challenging. Our experimental results achieved so far however show, that it is very difficult to find control parameter values which cause a stable ecosystem. A systematic approach to find a reasonable set of initial parameter values could be the application of an appropriate optimisation method, e.g. Genetic Algorithms or Simulated Annealing. The optimisation task which has to be solved here would be to maximize the time until one of the two species dies out. Another interesting topic of our future work is to develop an automated approach for interactive experiments, i.e. to develop a program or script which permanently observes the ongoing dynamic evolution process and automatically computes the necessary parameter adjustments to prevent that one of the species becomes extinct. Finally, we also intend to further increase the behavioural possibilities of the agents through more sophisticated acting rules and capacity to learn. In this connection we are working on an appropriate encoding that allows an evolutionary development of behavioural patterns.

References

1. C.G. Langton (edt.), 1997. Artificial Life: An Overview. Bradford Books.

2. Tom M. Mitchell and Thomas M. Mitchell, 1997. Machine Learning. McGraw-Hill.

3. Jose C. Principe, Neil R. Euliano, and W. Curt Lefebvre, 1999. Neural and Adaptive Systems: Fundamentals through Simulations. John Wiley & Sons.

4. Laurene V. Fausett, 1994. Fundamentals of Neural Networks. Prentice Hall.

5. Lawrence D. Davis, Michael D. Vose, and Kenneth De Jong (eds.), 1999. Evolutionary Algorithms. IMA Volumes in Mathematics and Its Applications, Vol. 111. Springer Verlag.

6. David E. Goldberg, 1989. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley.

7. Richard J. Gaylord and Kazume Nishidate. 1996. Modeling Nature: Cellular Automata Simulations with Mathematica. Springer Verlag.

8. C. Adami, 1998. Introduction to Artificial Life. Springer Verlag.

9. T. Finin, Y. Labrou, and J. Mayfield. 1997. KQML as an Agent Communication Language, in "Software Agents", J. Bradshaw (edt.). The MIT Press.

10. M. Syrjakow, J. Berdux, and H. Szczerbicka, 2000. Interactive Web-based Animations for Teaching and Learning. *Proceedings of the Winter Simulation Conference* (WSC'00), Orlando, Florida, USA, December 10-13, 2000, pp. 1651-1659.