Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

INTERACTIVE WEB-BASED ANIMATIONS FOR TEACHING AND LEARNING

Michael Syrjakow
Joerg Berdux

Institute for Computer Design and
Fault Tolerance (Prof. D. Schmid)
University of Karlsruhe
76128 Karlsruhe, GERMANY

Helena Szczerbicka

Institute for Computer Science
Fachbereich Mathematik und Informatik
University of Hanover
30167 Hanover, GERMANY

ABSTRACT

Web-based study resources can be viewed as a basic requirement in order to remain a competitive player on a more and more globalised educational market. For that reason it is getting increasingly important for universities to supplement offered lectures with additional Web-based learning material. In this paper we focus on interactive multimedia elements like computer animations and simulations, which can be used by students for individual experimentation. Such supplementary material represents a motivating but also a very effective chance to deepen and to increase the knowledge acquired in the lecture. This paper gives some general guidelines for building interactive Web-based animations. Beyond that, two of our developed animations are presented in detail. The first animation visualizes the search processes of some common direct global and local optimization strategies. In the second animation an artificial ecosystem is simulated, where several autonomous agents have to perform a number of different actions in order to survive. Our animations are realized as Java-applets, which have the advantage that they can be executed within Web browsers anywhere in the World at any time and without having to install anything.

1 INTRODUCTION

Simulation is a subject of education but also a very powerful tool for education. In this paper we address the second aspect by describing some simulators, which we have developed, especially for teaching natural and social science subjects. Previous experiences gathered in the field of computer aided learning and instruction have shown that the realisation of illustrative and didactically valuable computer-based learning material usually is very expensive (Diaz and Fernandez 1996). For that reason an implementation of such material only makes sense if it is frequently used over a long period of time. Today the Internet has established a powerful model for providing information and services to people all over the world. Therefore it is an obvious decision to use Web technologies like HTML, XML, and especially Java, the programming language of the Internet, for implementation of Web-based study materials. Beside platform independence Java provides the following advantages (Flanagan 1996)

- Java is familiar and simple,
- Java is object-oriented,
- Java is multi-threaded,
- Java provides a powerful set of class libraries.

In this paper we report about our experiences regarding the design, implementation, and use of interactive Web-based animations. Section 2 describes how Web-based animations can be effectively used in teaching, (online-) learning, and also in research. Subsequently, in Section 3 some general guidelines for the design of Web-based animations are presented. In Section 4 two of our developed animations are described exemplarily. Finally, in Section 5 we summarize and draw some conclusions.

2 USE OF WEB-BASED ANIMATIONS

Interactive Web-based animations can be used for many purposes. One important application field is teaching and learning. Here animations can be utilized as follows

- to liven up lectures
  Lecturers can use interactive animations within their lectures to better demonstrate and explain the behavior of complex dynamic systems.
- to improve Web pages of lectures
  Interactive animations presented in a lecture should be also offered on the Web. That way the students get the chance to make experiments by themselves, which is a motivating but also a very effective way to deepen and to increase acquired knowledge.

- to expand Computer Based Training applications
  CBT applications are mainly intended to check the personal learning success. For that purpose they usually offer different kinds of didactically approved multiple-choice tests. The quality of such CBT applications can be considerably increased by interactive animations, which have to be examined deeply by the learner to be able to answer the questions.

Beside teaching and learning interactive Web-based animations can be also excellently used to support research. Here they can be applied

- to acquire fundamental knowledge
  Sophisticated animations as presented in Section 4 give deep and illustrative insights into the behavior of complex dynamic systems. Especially animations with an appropriate presentation module may show new and undiscovered aspects of a dynamic system.
- to demonstrate research results
  Web-based animations are an excellent means to demonstrate research results to interested people all over the world in a way that is also understandable for non-experts.

More details about the usage of interactive Web-based animations especially within CBT applications can be found in (Syrjakow et al. 1999).

3 SOME GENERAL GUIDELINES FOR THE DESIGN OF WEB-BASED ANIMATIONS

Today the most common and suitable way to realize a Web-based animation is to implement it as a Java-applet, which can be executed platform-independently within Web browsers all over the world. The general architecture of such an animation applet is shown in Figure 1. Core of the animation applet is an animation engine, which has the task to compute state transitions of the animated dynamic system. Principally, there exist two realization alternatives for an animation engine

1.) animation engine, which is able to compute all possible sequences of state transitions of the animated system. Such a "complete" animation engine requires a full implementation of the state machine (algorithm), which drives the animated system.
2.) animation engine, which is able to compute only a (limited) subset of all possible sequences of state transitions of the animated system. Such a "reduced" animation engine has the advantage of being much cheaper to realize as a complete one. The great disadvantage however is, that the resulting animation usually is not very powerful, expressive, and interactive.

In the following, we will focus on animations with complete animation engines. In case of very complex animations, the problem may arise, that the animation applet cannot be executed on the client any more, because it would require too much computational resources. This problem can be solved by executing the animation engine not within the Java-applet on the client, but on a more powerful machine on server side. However, when we expect a PC of today on client side, server side execution is not required in most cases, because the computational power of a modern PC is usually sufficient to compute also demanding Java animations. The rather complex animations presented in Section 4 for example can be executed on client side without any problems.

The second important building block of the animation applet shown in Figure 1 is the presentation module. Outgoing from the data generated by the animation engine the presentation module has the task to visualize the state transitions of the animated process on the screen. The realization of the presentation module strongly depends on the kind of the animated system. Sometimes it may be useful to realize several presentation modules in order to allow the user different views on the animated system. In some other cases it is possible to realize a presentation module, which is generally applicable to a whole class of similar systems. An example of such a general presentation module is presented in Section 4.1.

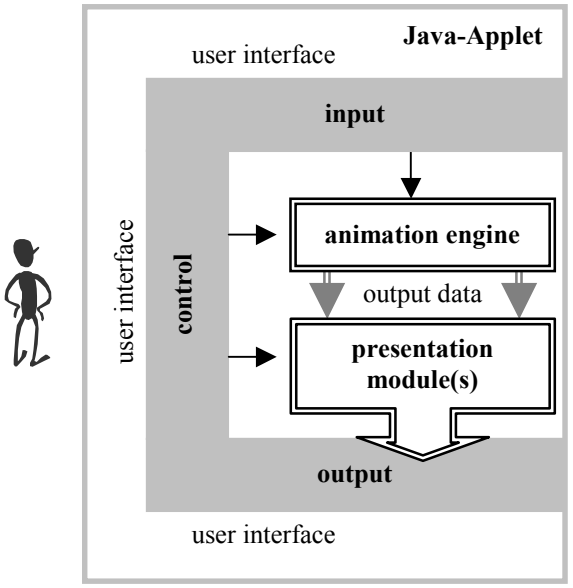

Figure 1: General Architecture of an Interactive Web-Based Animation

The third important component of an animation applet is its graphical user interface (GUI). Here an intuitive and

user-friendly design is of great importance. As shown in Figure 1 the graphical user interface consists of the following three parts:

- input part to define and to parameterize an animation experiment;
- control part to interactively control (start, stop, switch into a step mode, etc.) the running animation process;
- output part to observe the state changes of the animated system but also to examine statistical data about the animation after it has finished. As shown in Figure 1, the quality of the output part mainly depends on the realization of the presentation module.

Based on the general design guidelines described above we have developed several Web-based animations. In the following Section two of these animations are described in detail.

4 EXAMPLES

This Section describes two of our developed interactive Web-based animations. In the first animation the search processes of some common direct optimization algorithms are visualized. The second animation simulates an artificial ecosystem, where adaptive autonomous agents perform a number of different actions in order to survive. Both animations are realized as Java-applets, which can be accessed on the World Wide Web at the following URL: <goethe.ira.uka.de/people/syrjakow/animations.html>.

4.1 Animation of Direct Search Algorithms

During the last three decades direct search methods have gained great importance in optimization. Compared to traditional mathematical optimization techniques direct search strategies have the advantage that derivatives or other auxiliary knowledge about the optimized goal function is not required. For orientation direct search methods use nothing but goal function values. This property makes them general applicable and predestinate to black box optimization. In our research work direct optimization algorithms have been applied very successfully to parameter optimization of simulation models (Syrjakow and Szczerbicka 1995; Syrjakow and Szczerbicka 1997).

Some well-known direct optimization methods for global search are Genetic Algorithms GA (Goldberg 1989; Michalewicz 1992) and Simulated Annealing SA (Aarts and Korst 1990). These methods extensively apply probabilistic search operators, which are based on principles of nature. A well-known representative for direct local optimization is the Pattern Search algorithm of Hooke and Jeeves (1961). This very efficient Hill-Climber is solely based on deterministic search operators.

One big problem of the powerful optimization heuristics described above is that a lot of experience and expertise is needed to successfully apply them to a given optimization problem, i.e. to find appropriate control parameter settings for their sophisticated search operators. For getting the right feeling for such algorithms a visualization of the search process has been proven to be very helpful. In the following we describe a Java-applet for animation of the complex search processes performed by the direct optimization methods mentioned above. We call this applet an animation environment because its presentation module is general applicable to all kinds of direct search algorithms. The main objectives of this animation environment are

- to gain a better insight into the sophisticated working mechanisms of direct optimization methods;
- to offer inexperienced users the possibility to properly deal with these methods (to get a feeling for a good parameterization of their search operators);
- to acquire fundamental knowledge which enables to further enhance the performance of direct search methods.

Figure 5 and Figure 6 give an overview of the graphical user interface of the animation environment. It consists of the following three parts:

- welcome part (Figure 5)
  Here the user can choose an optimization strategy as well as a goal function to which the optimization strategy is applied. At the moment the user can choose between Genetic Algorithms, Simulated Annealing, and Pattern Search, which are offered in the right choice menu. After selection of one of the 13 goal functions offered in the left choice menu a 3D- and a 2D-representation of this function is shown below.
- animation part (Figure 6)
  This part enables the user to observe and to control the running animation. More information about the animation part is available below.
- help part
  The help part comprises user instructions for the animation environment, detailed information about the animated optimization algorithms and links to other applets and Web pages about Genetic Algorithms, Simulated Annealing, and Pattern Search.

The left frame allows the user to switch between the different parts of the animation environment.

In the following the animation part is described more detailed. First of all, it should be mentioned that the underlying animation engine consists of complete implementations of the offered direct optimization algorithms, which enables the user to exhaustively examine their behavior. The main task of the animation part is to make the trajectory (sequence of state changes) of a running direct optimization process visible on the screen. For that purpose we have realized a flexible presentation module, which is generally applicable to all kinds of iteratively working direct optimization methods (point-to-point as well as population-based strategies). Population-based methods like GA generate a set (population) of search points in each iteration step, whereas point-to-point strategies like SA or Hill Climbing compute exactly one point. In order to keep the complexity of the presentation module low we restricted our considerations to 2-dimensional real parameter optimization problems with rectangular search spaces. In that special case the surface of the goal function can be easily represented by a 2-dimensional density plot. In the welcome part shown in Figure 5 the chosen goal function (function 3) is presented as a 3D-plot (on the left) as well as a density plot (on the right).

For the graphical visualization of the running optimization process the generated search points are plotted directly upon the density plot, which represents the playing ground for the optimization algorithm (see Figure 3, 4, and 6). When the presentation module is applied to point-to-point methods the generated search points are plotted one after another without deletion of the former ones. This way it is possible to observe the development of the whole optimization trajectory (all generated search points). In case of population-based search however, where the quickly growing optimization trajectory soon becomes very difficult to survey it makes more sense to plot the generated populations separately.

Besides the graphical visualization of the ongoing optimization process the animation environment also provides the user with a textual output, which is printed in the text area to the left of the density plot. Here detailed information about the single states of the optimization process is presented.

The animation can be interactively controlled by the user through the following buttons:

- "Start" button
  The search process of the selected optimization algorithm can be started by pressing the "Start" button. The "Start" button is also used to induce the presentation and/or comparison of computed optimization trajectories (for more details see "Comparison" button).
- "Stop" button
  The pressing of the "Stop" button stops a running search process. That way the user can comprehensively analyze selected states of an animation for an arbitrary amount of time.
- "Continue" button
  This button allows to continue a stopped search process.
- "Reset" button
  The "Reset" button causes the total suspension of a started search process.
- "Adjustment" button
  This button gives the user the opportunity to manipulate the control parameters of the selected optimization algorithm.
- "Comparison" button
  This button enables the user to display and/or to compare optimization trajectories computed by the implemented optimization algorithms. When the "Comparison" button has been pressed, a window appears that shows, whether optimization data regarding the currently chosen goal function is available or not. If a new goal function is selected all optimization data which has previously been stored is deleted. The comparison can be started by pressing the "Start" button.

The optimized goal function and the animated optimization algorithm can be changed using the two choice menus in the middle of the animation part. With the scrolling lists on the right it is possible (with a double click) to vary the shape and color of the search points, which are painted onto the density plot.

Now the capabilities of our animation environment are demonstrated by means of a detailed animation example. In this example the probabilistic search process of a population-based Genetic Algorithm is visualized. The task of the Genetic Algorithm is to maximize the 2-dimensional goal function shown in Figure 5.

In the following three different states of a typical animation run are presented. For parameterization of the GA we use the settings shown in Figure 2, which are based on recommendations from literature (Schaffer *et al.* 1989).

Figure 3 shows the optimization process just after computation of the initial population. It is easy to see that a random number generator was used to distribute the 20 individuals of this population randomly all over the search space.

Figure 4 shows the distribution of the individuals of the sixth population. Here we can already clearly recognize convergence towards the region of the global optimum point which is located at (4,4).

Figure 6 shows the situation after 10 computed generations. Here more than half of the population is gathered around the global optimum point. The best search point of the tenth population, which is distinguished from the other ones by a little cross already represents a very accurate approximation of the global optimum point.

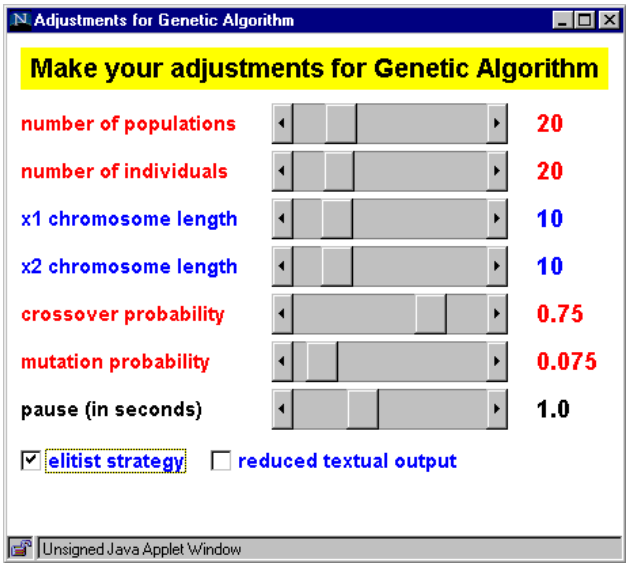

Figure 2: Control Parameter Settings of the GA

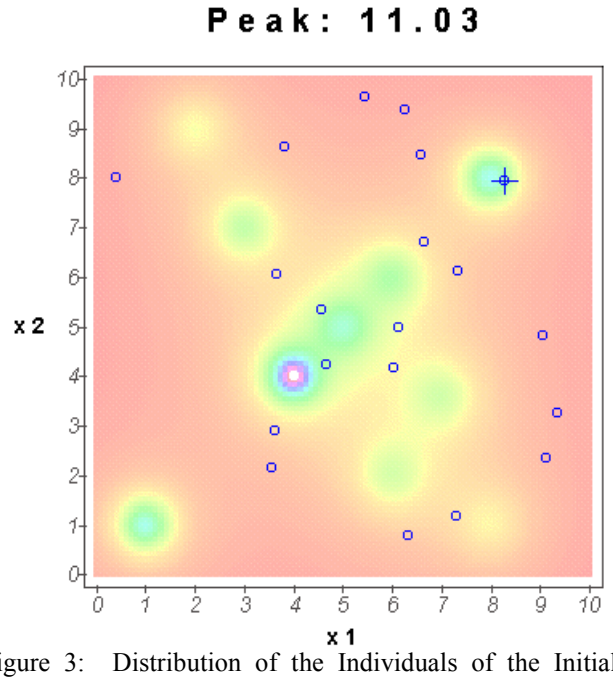

Figure 3: Distribution of the Individuals of the Initial Population

Summing up, the animation environment described above gives a detailed insight into the complex search processes of direct optimization algorithms. It enables the user

- to thoroughly analyze their behavior (also at extreme parameterizations);

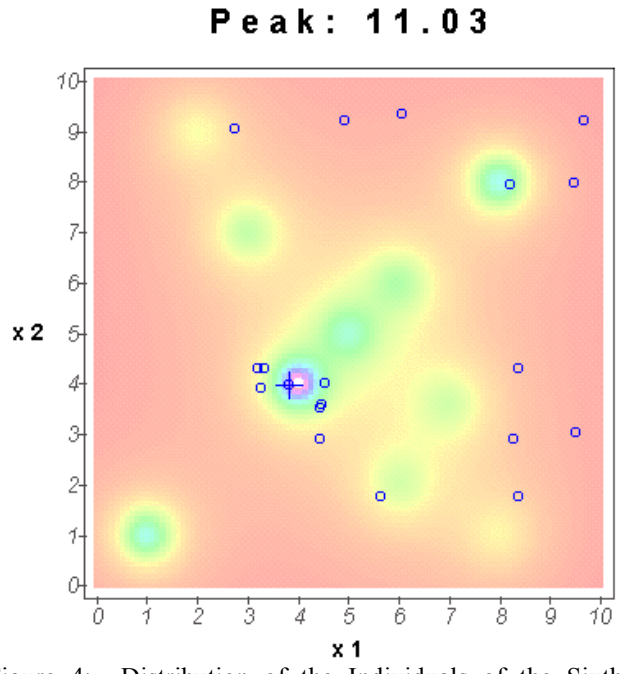

Figure 4: Distribution of the Individuals of the Sixth Population

- to evaluate design alternatives of their search operators;
- to adapt them optimally to specific optimization problems.

The animation environment is available as a Java-applet on the World Wide Web at: <goethe.ira.uka.de/people/syrjakow/anim_env3/start_environment.html>.

In our future work we intend to add further optimization algorithms to our animation environment as well as to enlarge the set of goal functions. Beside this, we want to equip the graphical user interface with more functionality. In this context a worthwhile feature would be a possibility for visualization of progressing convergence measures.

4.2 Animation of Artificial Life

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 (Langton 1997). The fundamental algorithms of AL are learning algorithms (typified by Neural Networks), evolutionary algorithms (typified by Genetic Algorithms),

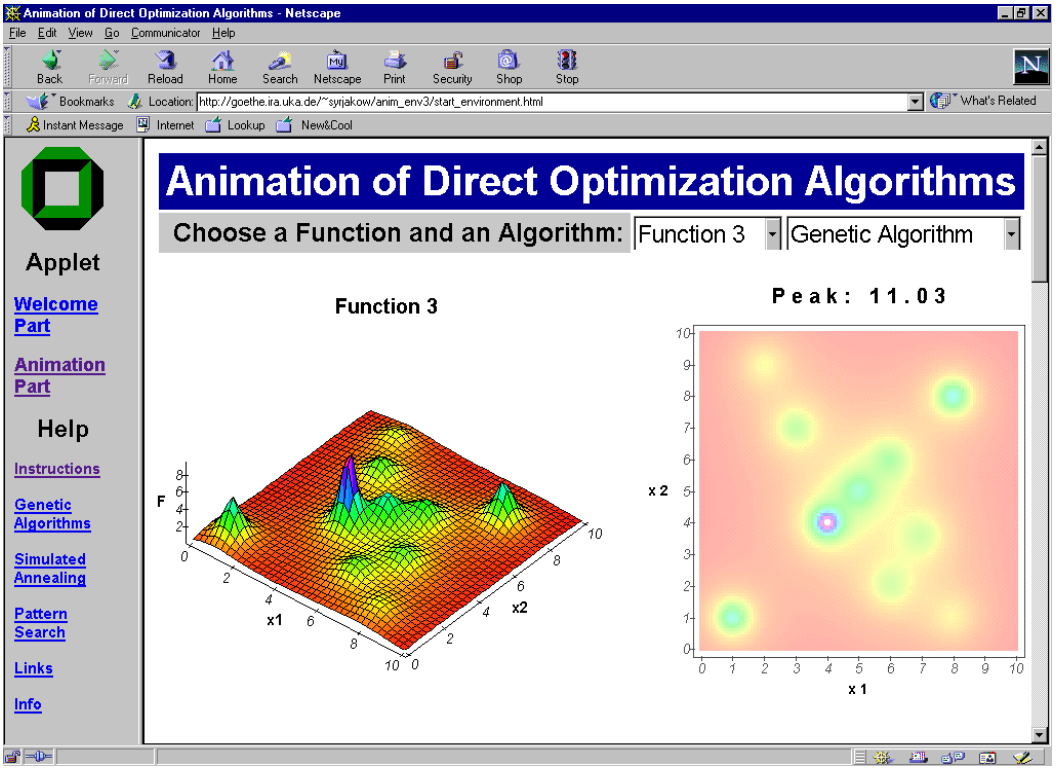

Figure 5: GUI of the Animation Environment for Direct Search Algorithms (Welcome Part)

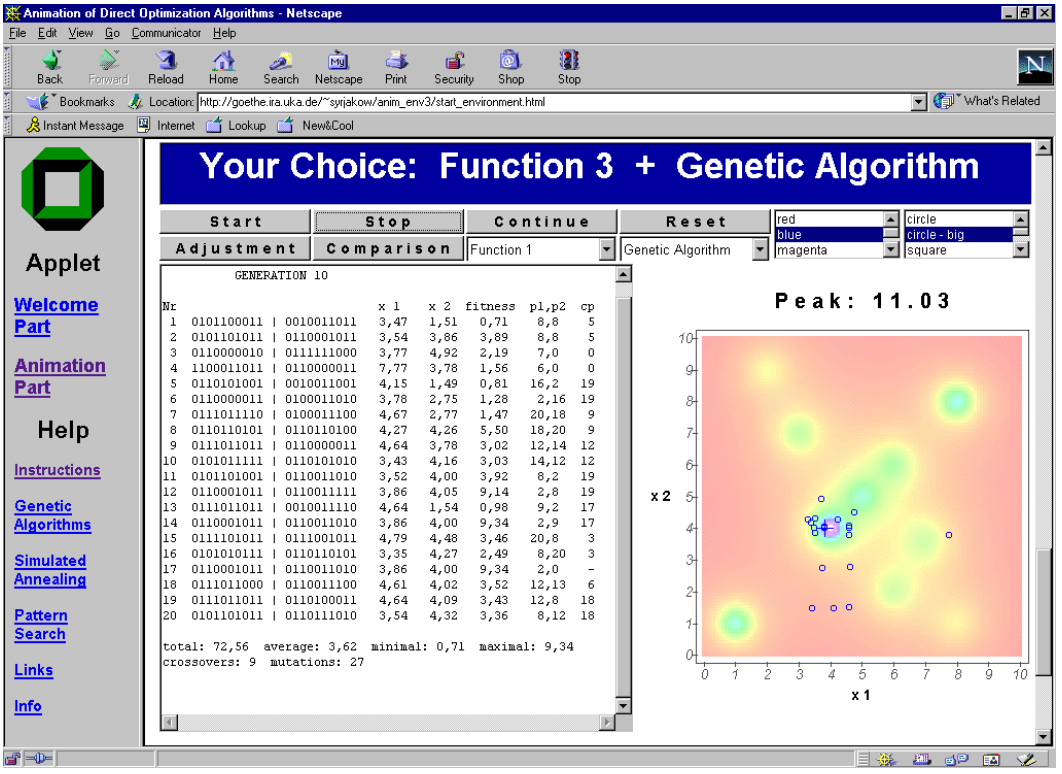

Figure 6: GUI of the Animation Environment for Direct Search Algorithms (Animation Part)

and cellular automata. 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 (Adami 1998).

Undoubtedly computer models play an entirely significant role in the interdisciplinary field of Artificial Life. On the one hand they are required by researchers to achieve research results. On the other hand they are an important tool for passing on the results to non-experts. The AL animation which is presented in the following mainly addresses the second aspect. It simulates an artificial ecosystem, where several autonomous agents perform a number of different actions in order to survive. The goal of each agent is to get as old as possible and beyond that, to reproduce itself as often as possible. For that purpose agents are first of all looking for food and secondarily for sexual partners. In particular, agents can carry out the following basic actions:

1.) look around in order to generate a local map of the environment,
2.) move in order to change the position within the ecosystem,
3.) eat in order to increase the energy level,
4.) sleep in order to save energy,
5.) reproduce in order to pass on the own genetic information to subsequent generations.

The environment in which the agents live consists of rectangular regions, which can be of the following three kinds: empty regions (desert), regions overgrown with food, which can be eat by the agents in order to increase their energy level (grass), and finally regions, which cannot be walked on by the agents (barriers).

Figure 7 shows the graphical user interface of the AL animation. The rectangle on the left represents the artificial ecosystem being occupied by a population of agents evolving in it. Our AL animation is realized as a Java-applet. It can be accessed on the World Wide Web at the following URL: `<goethe.ira.uka.de/people/syrjakow/agents/EcoSystemApplet.html>`. The applet allows the user

- to initialise and to edit the ecosystem (add/delete agents, desert, grass or barrier blocks),
- to start/stop/continue animation runs,
- to vary the animation speed,
- to observe the evolving population as a whole (macro view),
- to observe the life cycle of single individuals (micro view),
- to interactively change environmental parameters (food growth rate, life expectancy of the agents, etc.) during a simulation run.

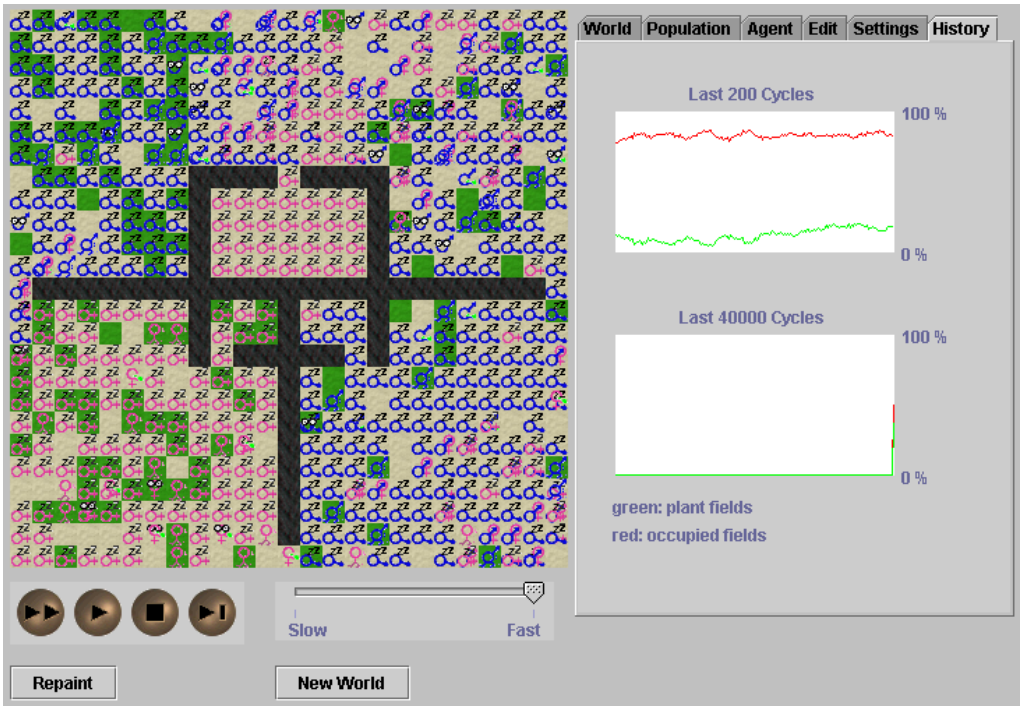

Figure 7: GUI of the Artificial Ecosystem Animation

Our AL applet represents a good means for getting a first impression of how artificial life implementations may look like. It allows to observe how primitive artificial organisms are forming rather complex communities while evolving and adapting themselves to their environment. In Figure 7 a snapshot of such an evolving community of autonomous agents is presented.

Our applet can be used within lectures but also as supplementary interactive Web-based study material allowing students to make experiments by themselves. At the moment we are developing a distributed and a more complex AL animation in order to obtain more realistic artificial life scenarios, where the agents have more behavioural possibilities like capacity to learn, more sophisticated acting rules, etc.).

5 CONCLUSIONS

Today the creation of modern and demanding study material becomes more and more important. This material must be interactive, user-friendly, and available on the World Wide Web. In this paper some general guidelines for the design and development of interactive Web-based animations were presented. We focused on animations with complete animation engines being able to compute all possible sequences of state transitions of the animated system. In order to show how well designed interactive animations may look like two examples were presented. The first animation visualizes the search processes of some common direct optimization algorithms. The second animation simulates an artificial ecosystem, where adaptive autonomous agents perform a number of different actions in order to survive. These animations, which are realized as Java-applets can be used by lecturers to liven up their lectures but also by students for consolidation and rework of lecture contents. Another important application field of these animations is research. Here they can be used to acquire fundamental knowledge but also for an illustrative demonstration of research results.

Finally, one last advice for all those who now intend to build their own Web-based animations: Before you start you should thoroughly search the Web because the animation you want to develop may already exist. Such tiresome searches as well as unnecessary developments of already existing animations could be easily avoided through the establishment of an open online library/repository, where Web-based learning material is gathered and administered.

ACKNOWLEDGMENTS

We want to thank Prof. D. Schmid for his encouragement and support of our work. We also thank our students, especially C. Bentz, Matthias Liefländer, and Dietmar Püttmann for their engagement and contributions.

REFERENCES

Aarts, E., and J. Korst. 1990. *Simulated annealing and Boltzmann machines*. Wiley.

Adami, C. 1998. *Introduction to artificial life*. Springer Verlag.

Diaz de Ilarraza Sanchez, A., and I. Fernandez de Castro (eds.). 1996. *Proceedings of the Third International Conference on Computer Aided Learning and Instruction in Science and Engineering*, CALISCE'96, San Sebastian, Spain, July 29-31.

Flanagan, D. 1996. *Java in a nutshell*. A Nutshell Handbook. O'Reilly.

Goldberg, D.E. 1989. *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley.

Hooke, R.A., and T.A. Jeeves. 1961. Direct search solution for numerical and statistical problems. *Journal ACM*, 8: 212-221.

Langton, C.G. (edt.). 1997. *Artificial life: an overview*. Bradford Books.

Michalewicz, Z. 1992. *Genetic algorithms + data structures = evolution programs*. Springer Verlag.

Schaffer, J.D., R.A. Caruana, L.J. Eshelman, and R. Das. 1989. A study of control parameters affecting online performance of genetic algorithms for function optimization. In *Proceedings of the Third International Conference on Genetic Algorithms*, June 4-7, George Mason University, pp. 51-60.

Syrjakow, M., and H. Szczerbicka. 1995. Simulation and optimization of complex technical systems. In *Proceedings of the 1995 Summer Computer Simulation Conference (SCSC'95)*, Ottawa, Ontario, Canada, July 24-26, pp. 86-95.

Syrjakow, M., and H. Szczerbicka. 1997. Efficient methods for parameter optimization of simulation models. In *Proceedings of the 1st World Congress on Systems Simulation (WCSS'97)*, Singapore, Republic of Singapore, September 1-3, pp. 54-59.

Syrjakow, M., and H. Szczerbicka. 1999. Java-based animation of probabilistic search algorithms. In *Proceedings of the 1999 International Conference on Web-based Modeling and Simulation*, part of the Western MultiConference (WMC'99), San Francisco, USA, January 17-20, pp. 182-187.

Syrjakow, M., J. Berdux, H. Szczerbicka, B. Zimmermann, and A. Otto. 1999. A flexible Java-based authoring system for building multimedia-enriched CBT applications. In *Proceedings of the 13th European Simulation Multiconference (ESM'99)*, Warsaw, Poland, June 1-4, Volume I, pp. 324-328.

AUTHOR BIOGRAPHIES

MICHAEL SYRJAKOW was born in 1964 in the Federal Republic of Germany. He received the Dipl.-Inform.

degree from the University of Karlsruhe, Germany in 1991. Since then he has been with the professional group Performance Modelling at the Institute for Computer Design and Fault Tolerance at the University of Karlsruhe. In February 1997 he received the Ph.D. in Computer Science from the University of Karlsruhe. His email and Web addresses are <syrjakow@ira.uka.de> and <goe the.ira.uka.de/people/syrjakow/>.

JOERG BERDUX was born in 1966 in the Federal Republic of Germany. He received the Dipl.-Inform. degree from the University of Karlsruhe, Germany in 1995. In 1996 he worked as a freelance in the field of Multimedia/Internet. Since then he has been a research assistant at the Institute for Computer Design and Fault Tolerance. His email and Web addresses are <berdux@ira.uka.de> and <goethe.ira.uka.de/people/berdux/>.

HELENA SZCZERBICKA was born in Poland. She received the M.Sc. in applied Mathematics and the Ph.D. in Computer Science from the Technical University of Warsaw, Poland, in 1974 and 1982, respectively. In July 1985 she joined the Faculty of Computer Science at the University of Karlsruhe, Germany. In May 1994 she became a professor in computer science at the University of Bremen, Germany. Since May 2000 she has been a professor at the University of Hanover, Germany. Her email address is <hsz@informatik.uni-hannover.de>.