PERGAMON

Computers & Graphics 25 (2001) 955–964

COMPUTERS & GRAPHICS

www.elsevier.com/locate/cag

# An artificial life approach for the animation of cognitive characters

Fábio Roberto Miranda[a,*], João Eduardo Kögler Jr[a], Emílio Del Moral Hernandez[b], Márcio Lobo Netto[b]

[a] *SENAC College of Computer Science & Technology, Rua Galvão Bueno, 430 São Paulo, SP, Brazil*
[b] *Polytechnic School, University of São Paulo, Av. Prof. Luciano Gualberto, trav. 3, 158 São Paulo, SP, Brazil*

**Abstract**

This paper addresses the problem of cognitive character animation. We propose the use of finite state machines for the behavioral control of characters. Our approach rests on the idea that the cognitive character arises from the evolutionary computation embedded in the artificial life simulation, which in our case is implemented by the finite state machine. We present some of the results of the WOXBOT/ARENA research project. This project to build virtual worlds is aimed at the graphic simulation of an arena, where small mobile robots can perform requested tasks while behaving according to their own motivation and reasoning. Each robot is an intelligent agent that perceives the virtual environment through a simulated vision system and reacts by moving away from or approaching the object it sees. The conception and specification of the robots and environment are being done very carefully to create an open distributed object architecture that could serve as a test-bed freely available and ready to use for testing theories in some computational areas such as evolutionary computation, artificial life, pattern recognition, artificial intelligence, cognitive neurosciences and distributed objects architectures. Furthermore, it is a first step towards building a cognitive animated character. © 2001 Published by Elsevier Science Ltd.

*Keywords:* Artificial life; Computer cognitive animation; Evolutionary computation; Genetic algorithms; Neural networks

## 1. Introduction

Artificial life is a term originally intended to mean the simulation of macroscopic behavioral aspects of living beings using microscopically simple components [1]. However, the term spanned to designate a wide variety of simulated living creatures, including virtual characters whose behavior emerges from hierarchical and functionally specialized complex structures like an animal's body [2]. Both kinds of simulations are very interesting from the computational point of view: they offer very attractive means to computationally model complex behavior, a subject that has gained a special relevance under both the theoretical and the applied standpoints.

Artificial life worlds are computational simulations like virtual places where animated characters interact with the environment and with other virtual beings of the same or distinct categories. Different levels of sophistication can be found in these virtual creatures, from unicellular life with minimalist models to complex animals with detailed biomechanical models. However, all of them display behavior that emerges from the dynamics of a complex system (for instance, systems with learning, reasoning, ontologies, cognitive processes, etc.)

In the history of computer animation, characters had been first animated using geometrical and physical rules and constraints. New tendencies point to behavioral animation, where characters have some degree of autonomy to decide their actions. What we are aiming

*Corresponding author.
*E-mail addresses:* fabio.rmiranda@cei.sp.senac.br (F.R. Miranda), kogler@cei.sp.senac.br (J.E. Kögler Jr), emilio@lsi.usp.br (E. Del Moral Hernandez), lobonett@lsi.usp.br (M. Lobo Netto).

at is one of the most recent proposals to animation, cognitive animation. In this approach, artificial intelligence techniques are mixed with evolutionary computation and computer graphics to generate a computer animated character capable of acting in response to a storyboard or screenplay and displaying naturalism when performing other tasks not explicitly specified. Following this track to cognitive animation, we proposed a project to exploit the capability of several methodologies and techniques that employs adaptive algorithms, evolutionary programming and artificial intelligence techniques.

The present paper reports results obtained in a project set to build an artificial environment, the ARENA (Fig. 1), for animated virtual creatures. Although it was conceived to allow more than one robot or virtual creature to be easily introduced to co-exist in the same environment, its first version presented here is a single-robot virtual world.

Our goal is to obtain virtual creatures capable of performing specified tasks in their environment by exploring certain strategies and adapting those whenever necessary. This scenario can be useful for many purposes: for behavior modeling, as a laboratory of learning algorithms, for research on societies of virtual characters, in the study of collective dynamics of populations, etc. The ARENA robot—wide open extensible robot (*WOXBOT*) (Fig. 2)—has a vision system consisting of a simulated camera and a neural network that classify the visual patterns to provide input to a motor system controlled by a deterministic finite state machine (FSM). This FSM is an automaton obtained from an optimization procedure implemented with a genetic algorithm, and applied through generations of robots.

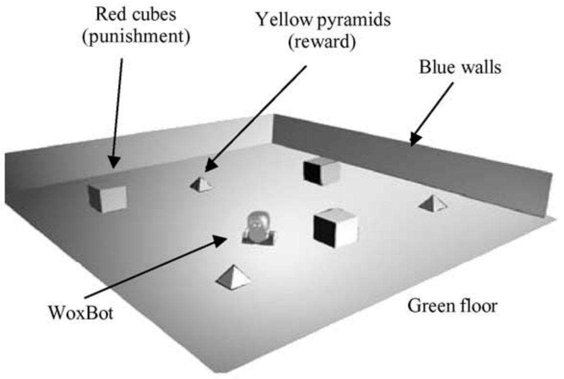


Fig. 1. Artificial environment for animats (ARENA)—a virtual world for the development of artificial life projects. The character at the center is a robot that avoids the cubes and seeks the pyramids. A contact with a cube reduces the robot life, while the pyramids extend it. The robot goal is to prolong life as long as it can. An evolutionary computing scheme selects the better performing robots. The environment can support several robots.

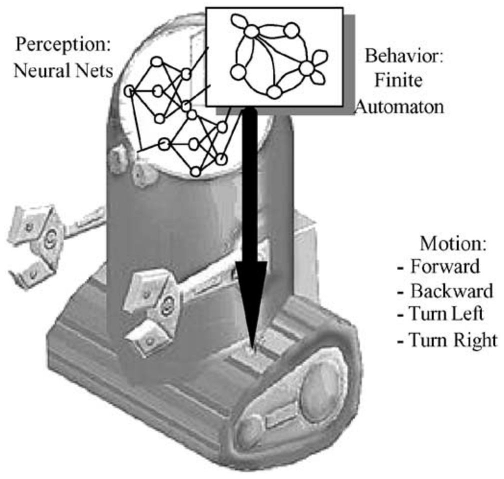


Fig. 2. Wide open extensible robot *WOXBOT*—a virtual character that gets input from vision and reacts by moving according to the decision taken by the finite state automaton.

As an example of application, a given research project can be targeting visual recognition methods, so it will be using *WOXBOT* as a subject employing the methods under test, and will take ARENA as a laboratory for evaluation experiments. Another research project instead, can target autonomous robot traveling strategies. In this case, the ARENA floor will be the field of traveling, and *WOXBOTS* will be simulating the autonomous vehicles. Yet the *WOXBOT* could be re-shaped to have the aspect of a given car or truck model and the ARENA could be configured to resemble a street, in a project intended to analyze vehicle traffic conditions.

The ARENA is implemented as a distributed object environment and can run on a single processor platform as well as can take advantage of parallelism or multithreading in high-performance computer architectures. In the latter case it could be used to handle highly sophisticated and complex applications, such as the simulation of the street with many cars. It could also be used to evaluate multi-agent distributed computational models using the *WOXBOT* as the prototypical agents.

The organization of the paper is as follows. Sections 2 and 3 review several conceptual aspects involved in the various project components and give guidelines to be used in the implementation. Section 4 gives an overview of the project, pointing out its requirements and choices for solutions. Sections 5 and 6 discuss particular implementation issues, like the environment and character models, and the choice of JAVA 3D as the programming language. Section 7 presents the results obtained in some simulations. Finally, Section 8 foresees the further steps to be taken with the project.

## 2. Artificial life

Artificial life has been associated to computer science by different ways, from cellular automata theory [3–5] to computer graphics animation [2–8]. Some researchers have been involved with this field looking for models that would describe how real life began and evolved. In fact they were looking for a universal life concept, which should be independent of the media on which they exist [3]. Other scientists were looking for physical models to give natural appearance to their characters. Very interesting results have been achieved here [2]. Although very different in nature, these works have been related to evolution and natural selection concepts. In many of them, evolutionary computing schemes such as genetic algorithms have been an important tool to assist the conduction of transformations in the genotype of virtual creatures, allowing them to change from one generation to another. Mutation and combination (by reproduction) provide efficient ways to modify characteristics of a creature, as in real life. Selection plays a role, choosing those that, by some criteria, are the best suited, and therefore allowed to survive and reproduce themselves.

Genetic algorithms are computational procedures to find the optimal solution in particular hard problems [9–12]. Each point of the optimization domain is represented by a binary string that is seen as a "genetic information", in the sense that this string can be mutated, by flipping bits, and two such strings can be mixed by a crossover operation, much likely as in the actual biological reproduction. These strategies provide an efficient way to obtain global optimization in cases where it is very difficult or not practical to formalize the optimization problem on an analytical framework.

In this work the evolutionary computation is employed to generate the control mechanism of the computer animated character. Each *WOXBOT* (Fig. 2) is controlled by a finite state automaton, which evolves through generations, based on genetic mutations and crossover. Descendant robots can arise with some innovative features that may be better or worse than those from previous generations. A character being better or worse is a relative concept, but it should reflect how well suited are the new robots to their environment. Those that by some criteria achieve higher grades (for instance, keeping greater energy, living more time, performing tasks faster, etc.) have higher chances to be selected for reproduction, and therefore to transfer the genotypes, and consequently, their features.

The use of knowledge in the artificial life environment can be done under two aspects: (i) it can be present in the environment and in the conception of the creatures and (ii) it can be used by the creatures themselves when performing their actions. In both cases, the effective and rational use of the knowledge (about the world, the actions, the tasks) leads to what is called intelligent behavior. Intelligence can also be considered a property emergent from evolutionary systems [5]. Thus, the evolving characters can be modeled as intelligent agents performing tasks in the virtual environment.

## 3. Intelligent agents

Intelligent agents may be understood as computational entities that behave with autonomy in order to manipulate the information associated with its knowledge. This autonomy must regard the agent design, the environment, the goals and motivations [12]. They have the capability of reasoning about what is going on in the environment where they are running, or to respond to some query conducted by external agents or persons to themselves. Many features of agents have been identified, such as their capacity to communicate with other agents, their mobility (migrating from one computer to another through networks), their intelligence (how efficiently they can perform or use reasoning), and so forth.

Our robots also exhibit "life" features that allow their classification as intelligent agents. In order to be autonomous and to present intelligent behavior these agents need to be able to sense the world, by getting information about the environment and creating their own representation of this world. Furthermore, they should be able to analyze this information, to use it to increase their knowledge and to reason about actions to be taken. Finally, they must be able to express these decisions through actions such as movement, communication, etc.

### *3.1. Sensing and perception*

Sensing is achieved by simulating mechanisms like vision and audition. Virtual scene images can be generated from the agent point of view, and represent its own and particular view of the scene. This image analysis is performed by a perception mechanism, implemented, for example, using a neural network for classification purposes.

Since we are interested in simulating natural phenomena, one could expect to see the natural evolution of all mechanisms used in our model. For instance, the neural network classifier should also result from the evolution of a primitive one. Our model, however, focuses on the usage of evolution only for the behavior aspects. Therefore, a pre-defined three level neural network has been chosen, and consistently trained to recognize the objects in our scene—the pyramids and the cubes. Herewith we are able to improve the quality of the sensing (rendering better images) and perception (adjusting the neural network size and training).

### *3.2. Behavior: reasoning and acting by learning and evolution*

The behavior is one form to express the capabilities of agents and their relation with the environment and with other agents. Agents may acquire and evolve reasoning capabilities using two main approaches: learning and evolution through generations. Learning is the skill of an agent that allows it to improve its knowledge about the world and to use this knowledge to perform a more elaborated cognition. It is a property of each agent and is continuously performed throughout its life. Evolution is a concept applied from one generation to another. Herewith the species can be improved, by combining the good features of some individuals with other good features from other individuals. Natural selection approaches complete the process, providing better survival chances for better-suited individuals after some criteria.

We propose a cognition system for our agents implementing reasoning in an automaton, which may evolve through generations. This reasoning controls the actions undertaken by this agent.

## 4. Project overview

Our research line in artificial life is aimed at the development of complex virtual worlds with realistic creatures able to exhibit sophisticated behaviors like reasoning, learning and cognition. To achieve it, the following aspects should be attended to:

- account for the wide variety of mathematical and computational models usually required to represent the complex aspects of living behavior;
- provide an efficient environment and platform with enough computational power to appreciate the details of the simulated creatures behavior in real time, including the ability to support live interactivity with persons;
- specify and build a system that embodies constant evolution and improvement in its constructs at the same time that enables or, moreover, promotes the re-use of well-succeeded results and implementations.

Our choice was to fulfill these requirements building an application development environment on top of an architecture of distributed communicating objects mapped on a microcomputer cluster [13]. However, the ARENA implementation also exploits parallelism through multithreading, so the program can run both on multi- and single-processor platforms.

The virtual space of the ARENA (Fig. 1) has a floor and walls encompassing the limits of the virtual world. Inside this scenario one can place objects of several kinds and functionalities like obstacles, barriers, traps, shelters, energy or food sources, etc., serving as pitfalls or resorts for the creatures. One or many characters can be introduced in this environment to perform certain tasks. In the first stage of the project the sole character will be the *WOXBOT* (Fig. 2), whose task is to keep itself alive as long as it can.

Our desire is to provide a very general environment able to serve as a tool for different research applications. Since the main focus is on artificial life and cognitive character animation, certain conceptual ingredients play specific roles in composing the artificial life scenario and thus should be present in the implementation: sensing and perception, knowledge use and evolution. In the following sections we examine these aspects.

## 5. Current implementation model

In this first implementation, while there is only one *WOXBOT* in the ARENA, it does not have any mechanism for providing communication with other robots, just sensing and mobility skills. Besides, *WOXBOT* has to decide which action it should take at every moment. A finite state machine has been evolved from scratch for this purpose. The robot should look for energy sources, while avoiding traps that make it weaker. As explained ahead, based on visual information, the robot plans its way to get closer to the energy sources, and a finite state machine that evolved through generations in fact conducts this planning.

The next versions of our robot will be able to communicate in an environment with other robots and they will also implement sociability skills, which are important features to promote a better adaptability from a creature to its environment.

### *5.1. Sensing and perception (pattern recognition)*

One of the tasks that *WOXBOT* has to perform in this first project is to be aware of nutrients (yellow pyramids) and hurting entities (red cubes) that are present in ARENA. The ARENA scenario and the *WOXBOT* vision are simulated by means of JAVA3D. The visual information is gathered from the 3D surrounding scene by projecting it to a view port with 3 color channels, namely R (red), G (green) and B (blue). Fig. 3 depicts a typical input, which at the present state is taken in low resolution without any loss of accuracy, since there are no textures yet used in our 3D virtual world simulation. To differentiate and spatially locate these entities, two specialized neural networks interpret the visual information received by the robot, one of them targeting nutrients and the other targeting hurting entities. These networks are named here ANN-I and ANN-II.

The network ANN-I is trained for the identification of yellow pyramids as well as for a general evaluation

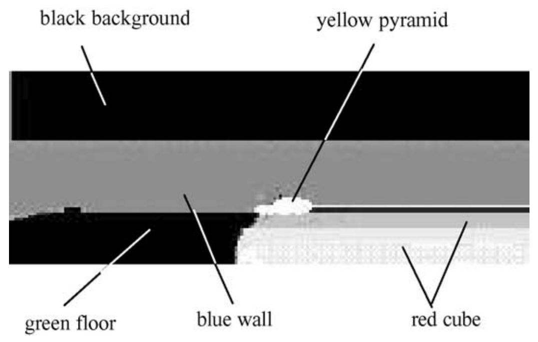


Fig. 3. A typical image of the ARENA environment as seen by the *WOXBOT* vision. The actual image has the depicted colors with shading resulting of the illumination. This image is the input of the neural nets that perform perception. It is also possible to add textures to the 3D scene and take yet more complex.

Table 1
ANN-I and ANN-II observers

| Code | Scene situation |
|---|---|
| 0 | No object ahead |
| 1 | Target object at left |
| 2 | Target object at center |
| 3 | Target object at right |

(classification) of the position occupied by the principal yellow pyramid in the robot's visual field. To help in this task, a simple filter for the yellow color is implemented as a pre-processing stage. This is done by combining the primary sensory channels RGB so as to obtain a gray scale where yellow is mapped into high values of gray and the remaining colors are mapped into low values of gray (Fig. 3).

The monochromatic image obtained in this way is then used as the input to the ANN-I, the one that targets nutrients. After training, this network is able to activate one of four outputs (see Table 1). In this way, ANN-I makes the robot aware of the presence and location of nutrients.

On the same lines as ANN-I, the second neural network, ANN-II, performs a similar function for the identification and position evaluation of the red cubes representing the hurting entities. For this second network, however, the gray images that are sensed by the neural network inputs are obtained by using a different filter, which combines the channels RGB so as to favor the red color.

The architecture chosen for ANN-I and ANN-II is the standard multi-layer perceptron with one single hidden layer of moderate size and learning through the error back propagation algorithm. Only eight hidden nodes were enough for this recognition task. Notice that this low number is consistent with the fact that we are dealing with a low number of output classes, i.e., only four.

Among the observations that we made during the training of the neural networks we want to mention that the detection and position classification of visual targets was facilitated by the use of samples (pyramids and cubes) of similar sizes. In other words, ANN-I and ANN-II do not have good abilities to implement scale invariance. That was in fact expected, since the complexity of the network is relatively limited, and its architecture does not have any specialized organization proper to implement scale invariance. This indicates that some kind of size invariance mechanism could be added in the pre-processing stage. Another possibility is to use a larger and more complex neural architecture so as to make possible the interpretation of targets positioned at broader range of distances.

An important issue to consider in the context of the neural networks of *WOXBOT* is adaptability. In these initial experiments with the robot, the training of ANN-I and ANN-II was done in an isolated phase, which was performed previously to the exploration of *ARENA*. In other words, only after *WOXBOT* was able to perform the differentiation between nutrients and hurting entities, it started its exploration in ARENA, and then, no further adaptation of the visual recognition system was allowed. In future phases of this project, we want the recognition system to be able to deal with unpredicted changing conditions such as change of illumination, and we will include the change of the features of the nutrients targeted by *WOXBOT* (shape and color for example). Of course, this depends strongly on the ability of the visual recognition system to adapt, since the target recognition prototypes change with new experiences and new needs of the robot, as its life and exploration of ARENA evolve. At that point of the project, the intrinsic adaptive nature of neural networks will be crucial, and it will be fully explored.

## 5.2. *Action and behavior: learning and reasoning*

The *WOXBOT* character is an intelligent agent with a simulated visual sensor to pick images of the environment from its point of observation. The two neural networks inside the agent analyze these images classifying the visual patterns. These networks outputs are tokens fed into the agent control system, which is an FSM. The inputs to the FSM generated by the combination of the ANN-I and ANN-II outputs are shown in Table 2.

Based on the above codes, the FSM chooses an action from its repertory. Table 3 shows the action encoding used to build the FSM representation as a genetic code.

Table 2
Input codes to the FSM and their meanings[a]

| FSM input code | Semantics | FSM input code | Semantics |
|---|---|---|---|
| 0000 | YP: No RC: No | 1000 | YP: Center RC: No |
| 0001 | YP: No RC: Left | 1001 | YP: Center RC: Left |
| 0010 | YP: No RC: Center | 1010 | YP: Center RC: Center |
| 0011 | YP: No RC: Right | 1011 | YP: Center RC: Right |
| 0100 | YP: Left RC: No | 1100 | YP: Right RC: No |
| 0101 | YP: Left RC: Left | 1101 | YP: Right RC: Left |
| 0110 | YP: Left RC: Center | 1110 | YP: Right RC: Center |
| 0111 | YP: Left RC: Right | 1111 | YP: Right RC: Right |

[a] YP = yellow pyramid and RC = red cube.

Table 3
Movement encoding

| Code | Action |
|---|---|
| 00 | Turn left |
| 01 | Go straight ahead |
| 10 | Turn right |
| 11 | Go backwards |

The motion control is performed by the FSM automaton, which is designed without specifying how the interstate transitions should occur. It is set initially with a random structure that is improved based on evolutionary computation concepts. Only the maximum number of allowed FSM states is specified. The FSM is represented by a string of bits coding its states, inputs and actions. This string is named the *WOXBOT* chromosome. For each FSM state there is a chromosome section. All these sections have the same structure: for each of the 16 possible inputs shown in Table 2, there is an entry on the chromosome state section, composed by two fields: the first one is the number of the next state after the transition caused by the corresponding input; the second field is the code of the action, following Table 3, taken upon the given input. Fig. 4 gives an example of an FSM and Fig. 5 is the corresponding chromosome.

At the start of the evolutionary process, a population of *WOXBOTS* is generated by sampling chromosomes at random from a uniform distribution. This population constitutes the first generation. Each member of the current generation is put into the ARENA, and it is assigned a performance value proportional to the duration of its life. A contact with red cubes shortens the life of the *WOXBOT*, while contact with yellow pyramids extends it. After each generation of robots in the ARENA, the fittest ones are selected to be the parents of the next generation and their chromosomes are grouped in pairs. These chromosomes thus undergo a mutation followed by crossover of each pair. This constitutes part of the next generation; the remnants are

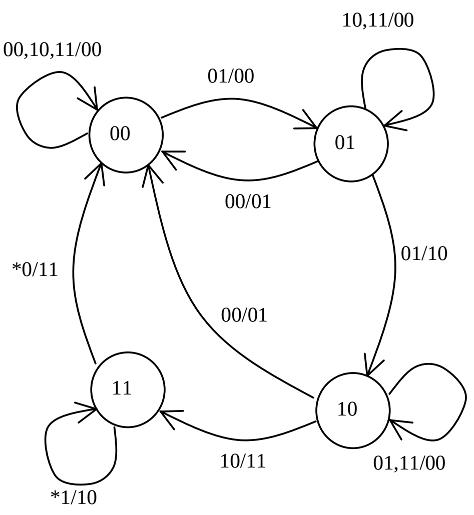
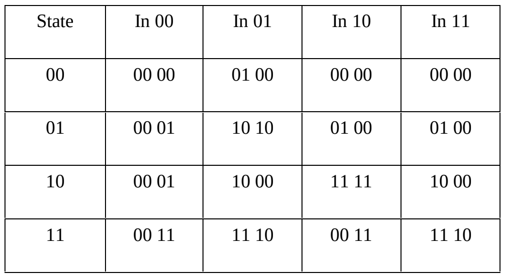


Fig. 4. A sample Finite State Machine (FSM) used by *WOXBOT* to control its reactions to the visual stimuli.



| State | In 00 | In 01 | In 10 | In 11 |
|---|---|---|---|---|
| 00 | 00 00 | 01 00 | 00 00 | 00 00 |
| 01 | 00 01 | 10 10 | 01 00 | 01 00 |
| 10 | 00 01 | 10 00 | 11 11 | 10 00 |
| 11 | 00 11 | 11 10 | 00 11 | 11 10 |

Fig. 5. Chromosome for the FSM of Fig. 4. There are 4 states. Each state has 4 inputs. Each Input has 2 fields, the first is the next state after the transition and the second is the action code.

completed, in a heuristic method of adding variability, by drawing them from a uniform probability distribution, in the same fashion as was done for the first generation. This procedure follows, giving thus more

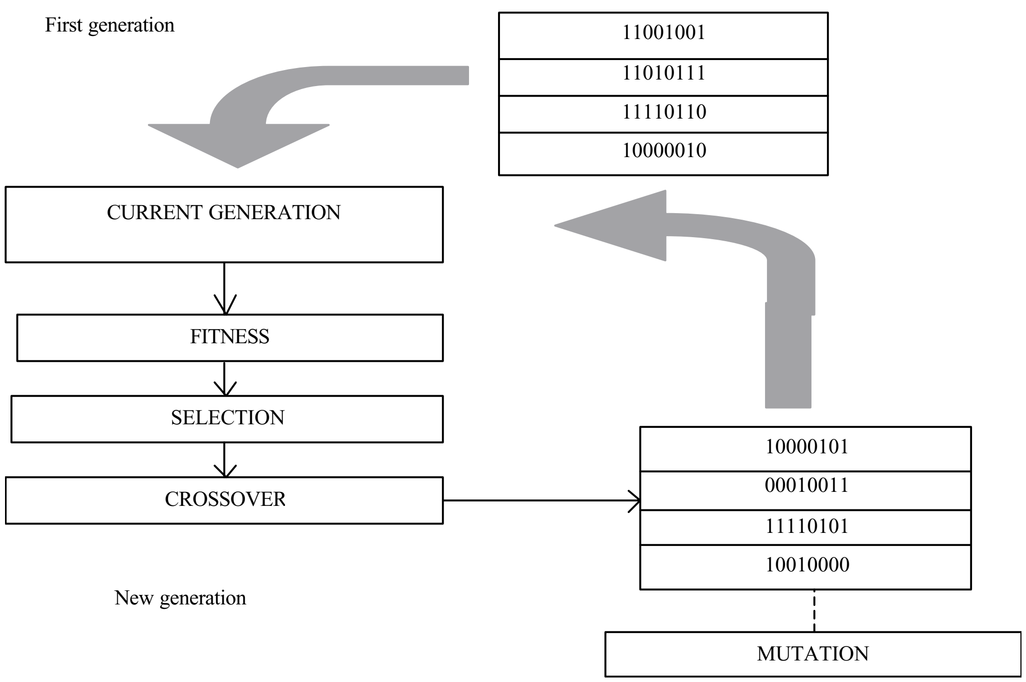


Fig. 6. The evolutionary process.

generations, until there is a stabilization of the fitness value of the population. Fig. 6 shows schematically the whole evolutionary process.

## 6. Implementation issues: character and environment representation

Before building an application intended for virtual world simulation, one key decision to be made is whether it will be a real-time application or not. Many 3D animation packages today, like Maya$^{TM}$ or Softimage$^{TM}$, have integrated application programming interfaces (API), allowing algorithmic animation of characters. This approach, yet useful for some kinds of simulations, has the drawback of few or none capabilities for user interaction.

Having decided for real-time capability, the next issue is the choice of the API. Some requirements we decide to fulfill with ARENA included: portability through many platforms, share ability and open architecture, and easy of use for others. Given these issues, our choice was made for the JAVA programming language for development and JAVA3D as 3D graphics package. JAVA is a worldwide spread and freely available language strongly object-oriented. Its three-dimensional extension, JAVA3D, is available on a variety of platforms (Windows, Linux, Irix and others), is also freely available and can be run through web browsers with the proper plug-in. The most powerful feature of JAVA3D is its scene-graph oriented approach that shifts the focus of 3D programming from vertexes and the rendering process to hierarchies of scene objects and events, shortening development time.

Additional advantages include loaders for content authored in 3D modeling and animation packages, use of underlying OpenGL or DirectX accelerated hardware, automatic use of threads for executions, taking advantage of parallel hardware.

### 6.1. The environment: ARENA

The ARENA environment is presently composed of a green rectangular floor-plan, surrounded by four blue walls and contains two types of objects: red cubic boxes that exhaust energy from the creatures inside the ARENA and therefore should be avoided by them, and yellow pyramids that furnish energy to the creatures, which in turn should be sought by them. Therefore, these creatures should spend their lives looking for pyramids, while avoiding cubes to keep their energy at levels that could sustain their lives. A threshold set to the energy levels will determine whether the creature is alive or dead.

We call a "robot life" the history from the first time the robot is introduced into the ARENA up to the moment that it dies because its energy has reached the minimum threshold level.

### 6.2. The character: WOXBOT

The *WOXBOT* character is an intelligent agent with a simulated visual sensor to pick images of the environment from its point of observation. Two neural networks inside the agent analyze these images classifying the visual patterns of environment configuration: one neural network is specialized in identifying the red cube positions relative to the robot (at the left, the right or the center) and the other does the same for the yellow pyramids. These networks outputs are tokens of an FSM embedded in the agent and responsible for taking decisions. The robot motion control is performed by this FSM. For instance, it can decide to either go ahead, turn right or left in order to reach a pyramid or avoid a cube.

This automaton was designed without specifying how the inter-state transitions should occur. It is set initially with a random structure that is improved by changing based on evolutionary computation concepts.

The state machine is coded into a string of bits representing its states, inputs and actions. This string is named the robot's chromosome. At the start of the robot life, the contents of this string are set at random, resulting in an arbitrary state machine. The optimization procedure using genetic programming begins with a set of robots initially created with arbitrary state machines. The optimization criteria are defined so as to accomplish the goal of maximizing the duration of the robot's life. Then, each of these initial robots is put to live individually in the ARENA, and their lives durations are recorded. The best-suited, i.e. the ones who lived more, are then preserved. They can generate descendants by reproduction making the crossover of the chromosomes producing a new breed. The newer generations are then tested in the same manner, until no expressive changes in the robot life duration could occur. Then, a random mutation is produced in some of the chromosomes, to induce variety and to proportionate a new strategy.

Based on this concept, we expect to be able to see after many generations the survival of the most adapted creatures.

## 7. Simulation and results

The genetic algorithm was used to evolve a population of 16 individuals along 30 generations. The first generation was randomly created, every subsequent generation had half of its members' genotype randomly generated and the other half created by a crossover of genotypes of the two most fitted individuals of the previous generation, followed by a mutation. The crossover occurred at a random point at the parents' chromosomes, exchanging complementary segments, so that resulting and original chromosomes have the same length. The mutation rate used in this simulation was 0.06, and it was a bit mutation rate.

A state machine with 4 states was used, which corresponds to a genotype of 258 bits (4 times 64 bits, the length of the description of each state transition/output relations, plus additional 2 bits to indicate the initial state of the machine).

The members of the generations are tested at the simulation environment one after another. In the simulation discussed in this paper each individual was given a time of 60 s that could be extended or decreased on the basis of the individual's behavior. Each time an individual collided against a yellow pyramid it received an extra 4 s and that pyramid was then considered "eaten", and a new pyramid would appear at its place only after 4.5 s. This distribution of values was made to avoid behaviors where individuals have the tendency to stay at the same place indefinitely just waiting for the energy. Therefore they will end up dying once waiting for energy spends more energy that is gained when the a waited energy arrives. The collision against a red cube caused the individual to lose 5 s. Cubes are also considered "eaten" after the individual collision, and at the simulation discussed here those cubes took 3 s to reappear. By varying parameters such as values of the rewards and punishments related to passing over to the objects, and the time it takes the objects to reappear, we expect to see a range of strategies from daring to conservative evolving.

Fig. 7 depicts the fitness curve along the generations. Some oscillation of the tendency is due to the low number of members in the populations, but the overall

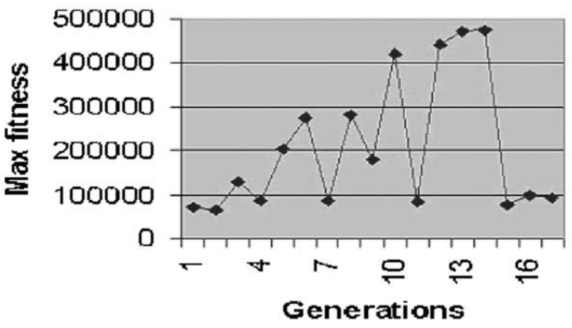


Fig. 7. Fitness of the *WOXBOTS* along the evolutionary process of a population of 16.

Initial state:3

3/2 3/0 0/3 1/0 0/3 3/3 2/1 3/3 2/3 3/0 3/2 3/1 2/0 3/1 3/1 0/2
2/2 0/3 2/3 3/0 0/3 0/1 3/0 0/3 1/0 2/2 1/0 1/1 0/2 3/2 1/0 1/3
1/0 2/3 2/2 0/3 3/1 1/0 2/0 1/0 1/3 1/0 0/0 1/1 3/2 0/0 3/2 0/0
0/1 2/0 2/3 3/3 3/2 3/1 3/3 2/1 0/0 1/0 1/0 0/0 0/2 1/3 1/1 3/2

Fig. 8. Chromosome of the best fitted individual of the experiments run with the specifications provided in the text.

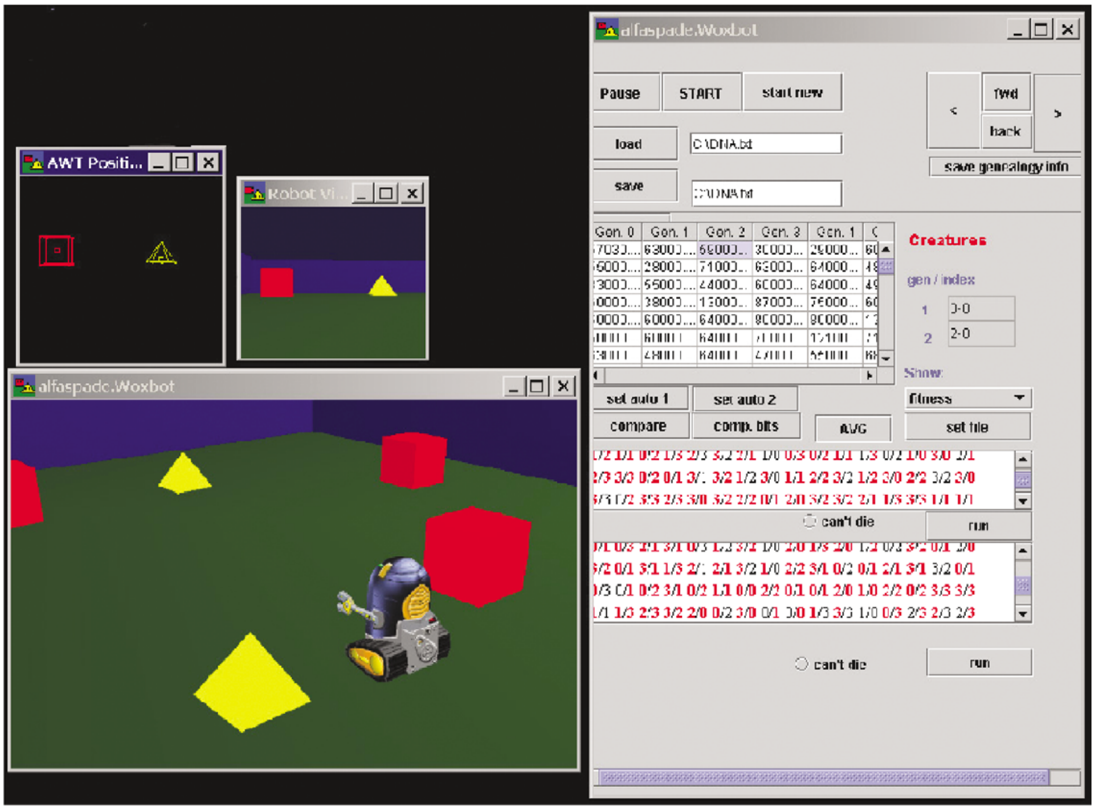


Fig. 9. The development desktop environment built with JAVA/JAVA3D. It allows one to follow the evolutionary process live action, data logging for the population parameters, chromossomes, etc. It enables one to load a previously evolved individual and watch its performance again.

tendency to increasing and stabilization can be appreciated. After a long term running with bigger populations, the fitness will probably stabilize. Fig. 8 shows the chromosome of the best fitted individual in the above experimental run.

At the present stage, the *WOXBOT* still does not memorize the previous actions, so it actually cannot take advantage of learning. The number of states of the FSM is also low—a maximum of only 4 states is allowed. These limitations were adopted just to provide a reasonable chance for an understanding of the behavior of the machine by examining its structure. Fig. 9 presents a snapshot of the environment built in JAVA 3D to develop the ARENA/*WOXBOT* project.

## 8. Conclusion and future prospects

The first stage of the project was to build the ARENA with a single *WOXBOT* to test the main idea and to orient the specification of the further steps. There are two main directions to follow simultaneously from now. One is to proceed with the ARENA introducing more complexity in the environment, in the tasks and using two or more *WOXBOT* simultaneously. The other is to improve the *WOXBOT* to build new and more sophisticated characters. Jack [6] can be an inspirational experience for us, where different levels of research development and application and incremental development were successfully explored.

Concerning the actual stage of the simulations, the *WOXBOT* will have some improvements, namely to allow many more states and to provide a memory for a finite number of previous actions. Also, the ARENA will undergo some improvements concerning its size, the distribution of objects, the presence of textures and allowing simultaneous *WOXBOTS*. Also, concerning the *WOXBOTS*, given the use of textures, improvements on its visual system will also have to be provided.

## References

[1] Bonabeau EW, Therulaz G. Why do we need artificial life? In: Langton CG, editor. Artificial life an overview. Cambridge, MA: MIT Press, 1995.

[2] Terzopoulos D. (org.). Artificial life for graphics in animation, multimedia and virtual reality. In: ACM/SIGGRAPH 98 Course Notes 22, August 1998.
[3] Adami C. Introduction to artificial life. Berlin: Springer, 1998.
[4] Bentley PJ, editor. Evolutionary design by computers. Loss Altos, CA: Morgan Kaufmann, 1999.
[5] Fogel DB. Evolutionary computation—toward a new philosophy of machine intelligence. New York: IEEE Press, 1995.
[6] Phillips C, Badler NI. Jack: a toolkit for manipulating articulated figures. In: ACM/SIGGRAPH Symposium on User Interface Software, Banff, Canada, October 1988.
[7] Badler NI, et al. Simulating humans: computer graphics, animation and control. Oxford: Oxford University Press, 1992.
[8] Miranda FR, et al. ARENA and WOXBOT: some steps towards the animation of cognitive characters acting in a virtual world, 10. Encontro Português de Computação Gráfica, 2001.
[9] Holland JH. Adaptation in natural and artificial systems. Cambridge MA: MIT Press, 1992.
[10] Koza J. Genetic programming. Cambridge, MA: MIT Press, 1992.
[11] Mitchell M. An introduction to genetic algorithms. Cambridge, MA: MIT Press, 1997.
[12] Beer RD, et al. A biological perspective on autonomous agent design. In: Maes P, editor. Designing autonomous agents. Cambridge, MA: MIT Press, 1990.
[13] Bernal V, et al. PAD cluster: an open modular, and low cost high performance computing system. In: Proceedings of the 11th Symposium on Computer Architecture and High Performance Computing, Natal, Brazil, September 1999. p. 215–22.