A Virtual Reality Framework for Artificial Life Simulations

Rogério Neves[1], Marcio Netto[1]

[1]Laboratory of Integrated Systems
Polytechnic School of the University of São Paulo
{rponeves, lobonett}@lsi.usp.br

Abstract. *This paper describes the project, codename ALIVE, whose first objective is the creation of a development framework for Artificial Life experiments in a three-dimensional virtual space. Special attention was given for visualization in Virtual Reality facilities, or using VR devices on personal computers. The framework provides an application and programming interfaces (API) for multi-agent and environment development, providing tools for manipulation, communication and visualization of agents in the experiment. The application supports a wide range of visualization technologies, from 3D-boards to high-end VR-devices, and makes use of parallelization in multi-processed computers or distributed systems.*

1. Introduction

This paper describes the ALIVE project, an Artificial Life experimenting framework developed at the Laboratory of Integrated Systems of the Polytechnic School, University of São Paulo, Brazil.

ALIVE is the acronym for Artificial Life in Virtual Environments, once the project uses Virtual Reality for the visualization of Artificial Life (ALIFE) experiments. The project comprehends the creation of a programming framework for agent and environment development, providing tools for agent manipulation, communication and visualization. The goal is to make the implementation process as easy as possible, requiring minimal language knowledge from the user to develop ALIFE experiments.

Some characteristics of the evolved framework include: user interactivity, friendly interface, concurrent execution for multi-processed systems, rendering optimizations for distributed systems (clusters), cross-platform capability, multi-agent architecture, immersive 3D graphics and vector based operations in the three-dimensional space.

The project first intended to make use of the five-side CAVE (virtual reality facility, see Cruz-Neira, 1992); also www.lsi.usp.br/~rv) available on the laboratory, but supports a wide variety of VR devices, from ordinary 3D Game Boards, monitors with optional Shutter Glasses, widely available for personal computers, to stereo monitors, Head Mounted Displays, and VR-Walls and Tables on professional systems.

The environment and agents executes in a thread-based multi-agent context, meaning that each agent and the environment are small pieces of programs running independently (Bentley, 1999; Lejter, 1996), but communicating with each-other,

allowing performance optimizations for parallel execution on multi-processed computers.

To implement the simulation framework, A.L.I.V.E. makes use of the cross-platform and multi-thread capabilities of the Java language and the visualization tools available trough the Java3D API (java.sun.org). An Applet launcher allows the web visualization of experiments on any browser with Java and Java3D support.

The agent's visual representation is based on Actors (a denomination similar to Avatar, adopted in the Java3D API specification). A wide variety of 3D objects can be imported for the agent representation, such VRML, LightWave and WaveFront Objects. The render engine can be alternatively disabled to improve the simulation performance, employing full computational power for behavioral processing. The framework features and utilization are described in detail in section 3.

The experiments developed with use of the framework are bundled with the project (www.lsi.usp.br/rponeves) and aim to demonstrate the capabilities for simulating micro-organism behaviors and environmental physics. The experiments actually being researched by the group explore emergence of complexity and the arise of cognitive characteristics from the bottom-up. These are synthesized in section 4.

The final conclusions and discussion for future work is presented in section 5.

2. Related work

Much have passed since the born of first Artificial Life programs, where experiments where most a bunch of printed pages filled with black characters, or a green screen with flashing squares like in Conway's Game of Life (Langton, 1995). Such experiments made use of epoch's available technologies and were of a vital importance to the conception of Artificial Life as discipline.

Nowadays, perhaps, the great increase of computational power, followed by the fast advance on Computer Graphics technologies, makes possible new experimental approaches for Artificial Life (ALIFE) simulations. The ALIFE programs are increasingly making use of 3D Graphics to represent more realistically the data generated in the simulation.

The first, and maybe the most famous, researcher to cross the Artificial Life and the Computer Graphics universes was Karl Sims, in his popular work on evolving virtual creatures. Sims experiments inspired a generation of researchers to apply 3D Graphics to the visualization of ALIFE experiments, as well as computer graphics professionals to apply ALIFE concepts to obtain more realistic animations.

Innumerous related researches are presently under development all over the world, in a scale that makes it impossible to summarize. Actually, in this same lab (LSI), two other projects are being developed over the subject: One related to Artificial Humans (Cavalhieri, Site) and the other under project name WOXBOT (Netto, 2000). The presented framework focuses on the Virtual-Reality and open-source capability, allowing one with sufficient programming skills to implement experiments, and profit on the advanced visualization engine and parallel, multi-agent architecture it provides.

In the computer graphics scenery, much on actual research can be found at www.biota.org, a site directed to Artificial Biology and Computer Graphics cross-

worlds. The book *"Artificial Life and Virtual Reality"* (Thalmann, 1994) brings some well-known applications and open problems on the topic.

3. Framework features

The framework is recommended to the simulation of nature-like systems in a multi-agent context. The use of a three-dimensional space is appropriate for the visualization in Virtual Reality, but bi-dimensional experiments can also be developed within the framework. The project was developed under an Object Oriented Paradigm (OOP), applying multi-agent concepts.

Three concepts are fundamental in the framework, they are: Environment, Agents and Actors, or Avatars. An Environment is where action takes place. It hierarchically contains the Agents and other objects of the simulation, possesses individual variables, physics and rules relative to the simulated virtual world. The Agent is the entity that exists in the Environment. It is a small piece of code that operates the variables and states on itself, the environment and other Agents. The Actors exists in the virtual scene context, representing the Agents and other objects present in the simulated environment by shapes specifically assigned for visualization purposes. Actors can make use of form, color, texture and transparency to identify components in the simulation. **Figure 1** illustrates the connection between the concepts.

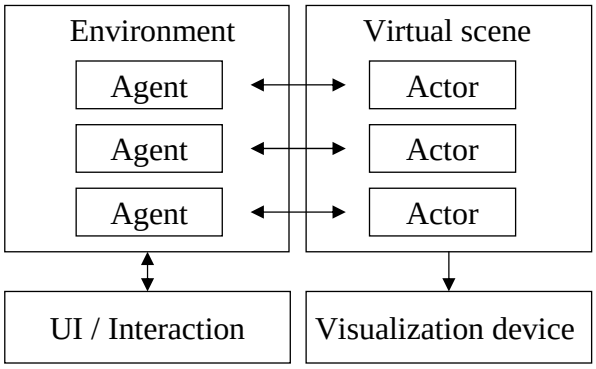

Figure 1. The simulation consists of Environment and executing Agents, while Actors are related to the visualization engine.

The main program is composed of seven classes. Among the classes, three are the Application Programming Interfaces (API), which must be extended by user experiments for the creation of new special purpose Environments, Agents and Actors. The project classes are presented in the **Figure 2**.

The Environment API and Agent API hold the common physical variables (vectors and flags) that represent the environment and agent current states. They also provide methods to create, communicate, orient and move agents in the environment. An experiment developed within the framework context shall use these super classes to coordinate physics and agent behaviors. **Figure 3** shows an experiment diagram developed in the described context. The communication between Agents is made trough the environment, that contains pointers to all living agents in the simulation in a vector. This vector is consulted every time an agent asks for some information. The ViewWindow class is responsible for the visualization, it directs the output to the

selected device and controls camera and light positioning. Cameras can be dynamically changed, by user interaction in real-time or linked to one agent's vision.

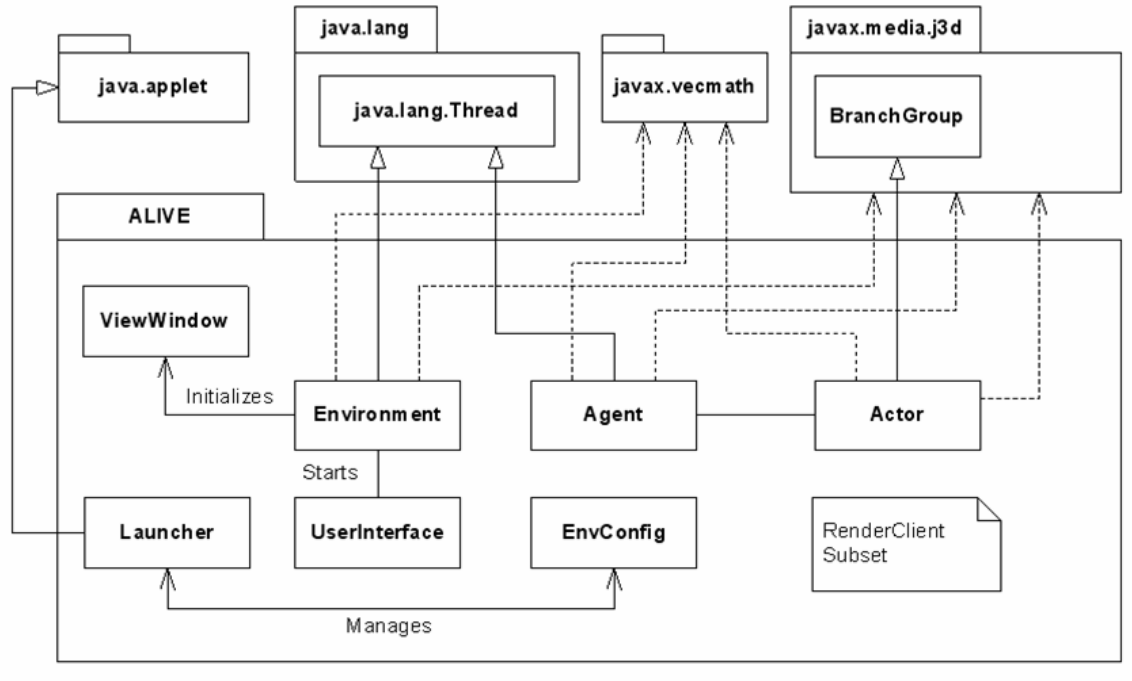

Figure 2. The UML diagram shows the dependencies for the project classes. The ALIVE package contains seven main classes and the RenderClient engine, for rendering over an IP network.

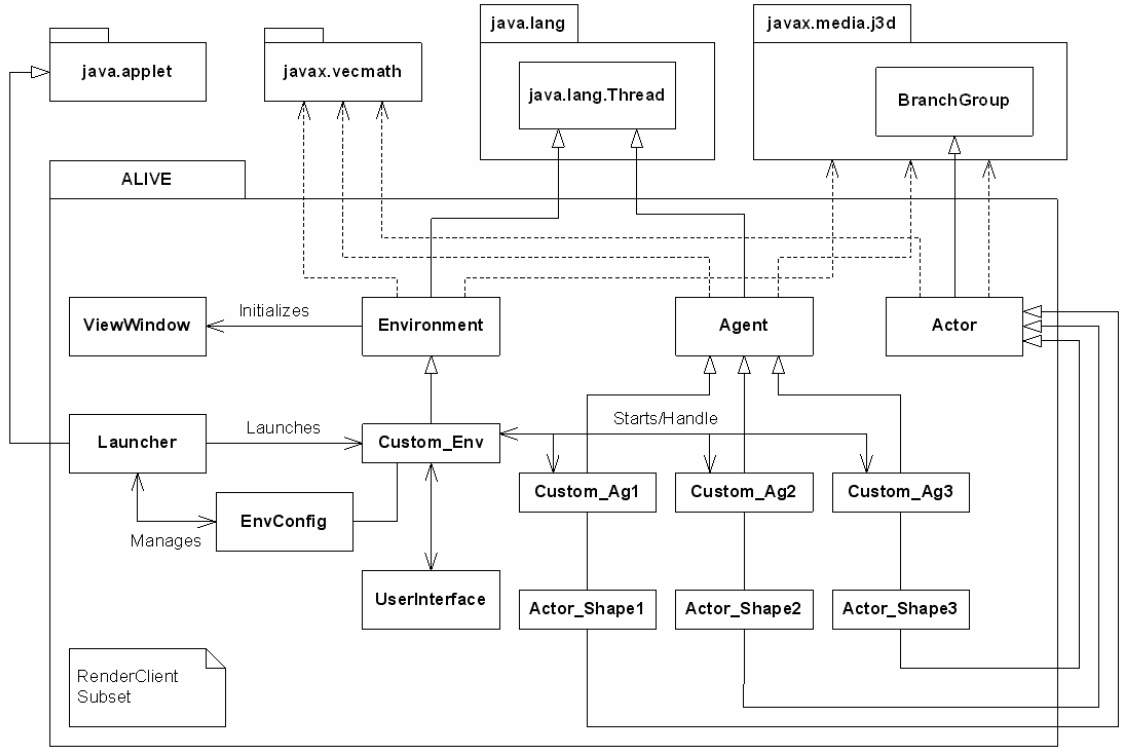

Figure 3. The UML diagram of an experiment developed within the framework. The classes Custon_Env, Custom_AgN (Agent) and Actor_ShapeN are user developed experiment classes extending the framework super classes.

In this context, the user-defined classes are written in a sort of pseudo code, composed of calls to the provided API methods and vector operations, combined with basic math in native language. They must define the physical laws of the environment and the behavior of the simulated agents. The heavy computation is done by the super classes and remaining project classes. **Figure 4** illustrates the level of code complexity.

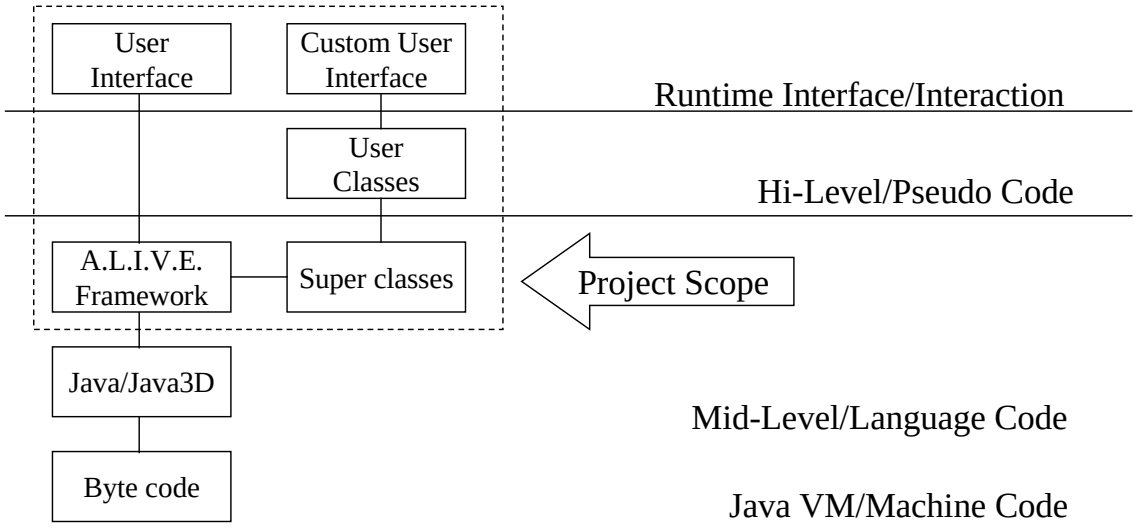

Figure 4. Levels of operability. Higher levels include visual interface and user-designed classes, going down requires higher programming skills and deeper language knowledge.

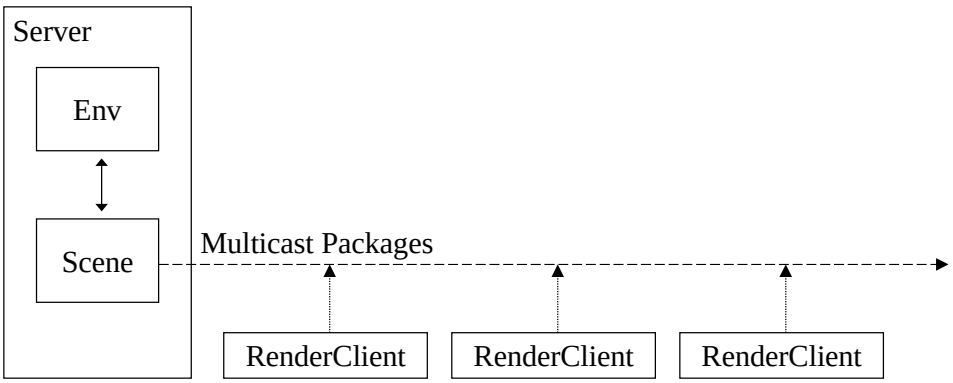

Figure 5. Distributed rendering making use of class RenderClient. This class makes possible the use of PC clusters for multi-screen rendering in systems like Cave.

Parallel computation speedups can be obtained in both multi-processed systems and distributed systems. For each agent program in execution there is an independent thread that can be executed in parallel if more than one processor is available. The maximum speedup in this case is obtained in multi-processed computers with shared memory. The use of clusters can also reduce the computational load in the rendering process. A remote computer connected to the network can execute the rendering job based on client-server communication, synthesizing the entire visual output, or part of it. As an example, the Cave facility employs a cluster of six PC to activate the five independent back projected walls. In this case, one PC runs the experiment, and sends multicast IP packages containing the role scene information to the other five computers

in the clusters, responsible for generating individual render outputs for each of the five virtual viewing cameras, each one pointing into a different direction in the virtual universe. The diagram for distributed rendering is schematized on **Figure 5**.

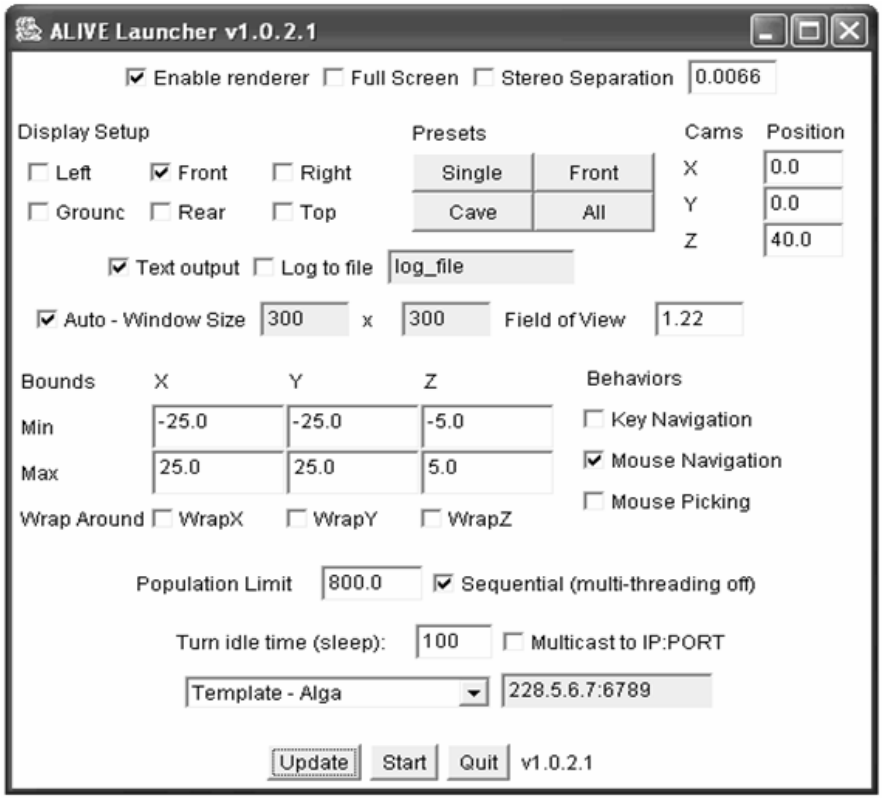

Figure 6. The experiment Launcher interface.

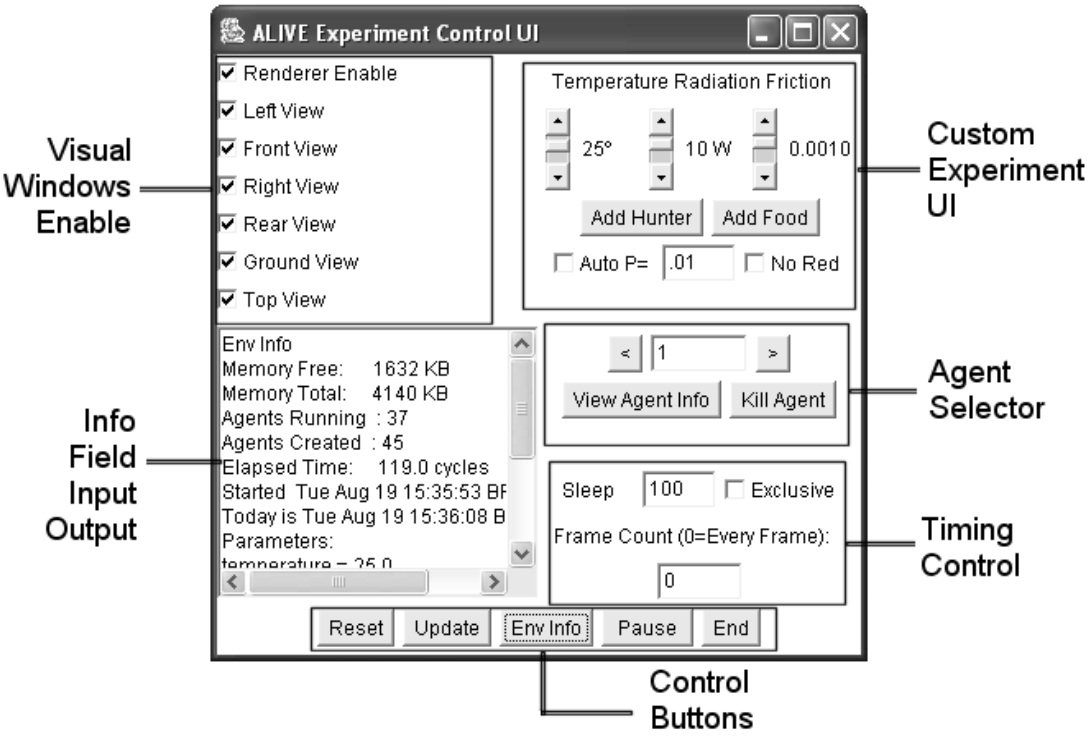

Figure 7. A The run time User interface

The Launcher class brings a configuration menu, presented in **Figure 6**, allowing the change of parameters and the selection of experiments. The User Interface class allows one to change parameters and visualization settings in run-time, turn the render on and off and to start or stop logging to a file. A Custom User Interface can be implemented for each experiment, allowing the user to change parameters and manage Agents during execution. The run time control interface is shown in **Figure 7**.

4. Experiments

Here are overviewed some case studies in ALIFE, showing some experiments developed by the group within the framework. These experiments were developed for study evolving characteristics in artificial beings, as well as demonstration and test of the framework. All of them are bundled with the actual compilation and its source and can be viewed in Applet mode in the project site (www.lsi.usp.br/rponeves). Due to limited number of pages and topic restrictions, only a brief introduction of the experiments is presented. Further information concerning ALIFE and the experiments functionality can be obtained in the project site.

4.1. Algae

The experiment is based on observations of the growth of colonies of photosynthetic algae, and shows how the natural selection apparatus works on the adjustment of luminosity frequencies absorption. The spectrum was divided in three bands, represented by red, green and blue colors. The environment luminosity can be adjusted. Thousands of years of evolution can be observed in a generations per second rate, as agents are born and die. **Figure 8** shows a screenshot of the experiment in execution.

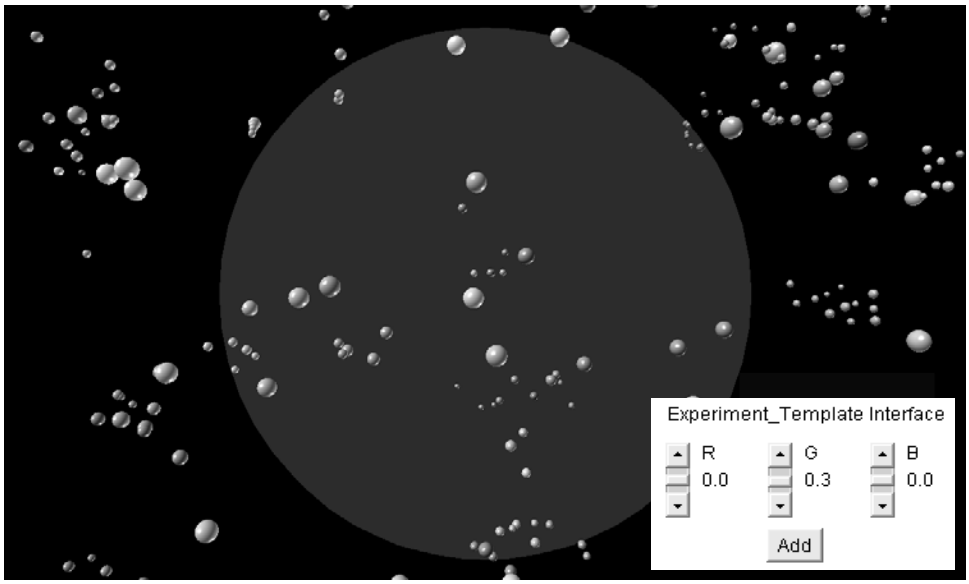

Figure 8. The Algae experiment. The detail shows the custom interface.

4.2. Predator & Prey System

The experiment was proposed to study the oscillation of population in time for a predator-prey system in a multi-agent context, suitably to the obtained by the traditional Lotka-Voltera equations. For this purpose, two types of agents were created, Predators

and Preys. The Prey agent simulates an ordinary photosynthetic bacteria growth pattern, multiplying in time according to the environment radiation. The Predator Agents consumes the prey-type agents with efficiency determined by the fitness of its chromosomes. **Figure 9** shows the environment screenshot and the schematics of Predator vision.

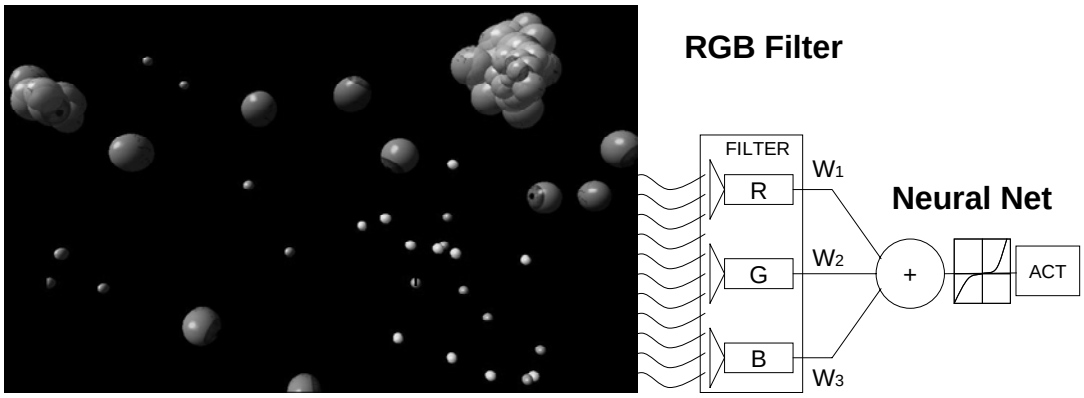

Figure 9. To the left, an experiment screenshot; to the right, the input luminosity in environment to the output action is schematized for Predators. One single neuron that defines the seek criteria. The DNA is composed of 14 chromosomes that determine the Predator's efficiency.

4.3. Fish Schooling

The focus of this experiment is to compare two types of intelligence, obtained by experience, or learned in life, and inherited by instinct. Each agent's (fish) DNA contains information to initialize the neural network with default values. These values are started randomly in first Agents and then are inherited suffering minor changes due to natural apparatus (mutation and cross-over). Once started, the neural network can regulate itself trough a simplified learning algorithm in order to increase the overall performance in searching food (these changes are lost when the agent dies). The food is dropped into the aquarium at an adjustable rate and the fishes try to catch them up. The neural net determines the relations between visual input and motion output in a fuzzy controller. **Figure 10** shows the experiment in execution.

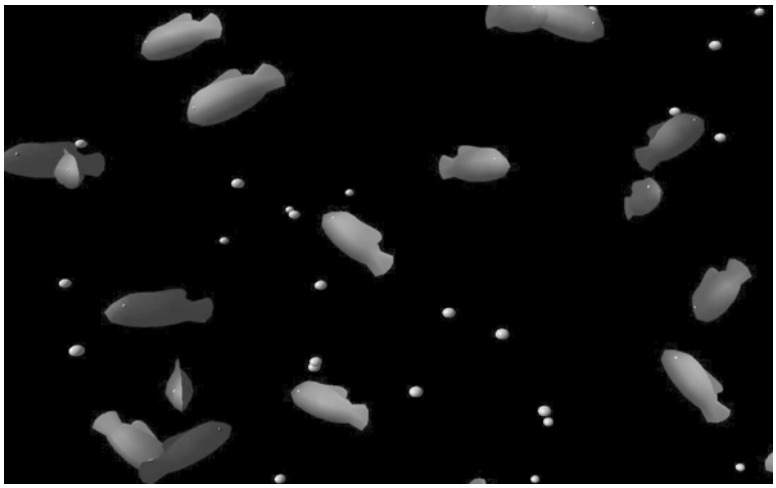

Figure 10. A screenshot of the experiment.

4.4. Flocking

The experiment demonstrates how a set of simple probabilistic rules can reproduce complex behaviors observed in nature. Flocking experiments are quite common in the Artificial Life scenery. The web is crowded of applets simulating swarms of insects or fish schooling. This experiment provides an interface that permits one to obtain a wide variety of flocking patterns changing probabilities in the decision process of agents individually. **Figure 11** shows the experiment screenshot.

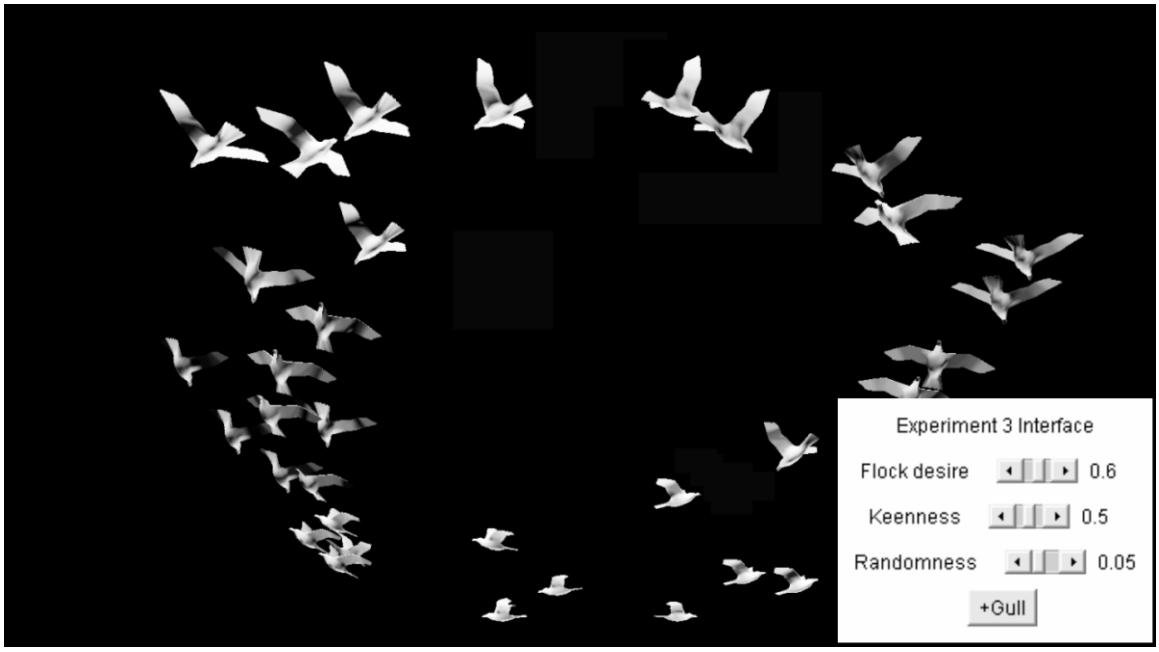

Figure 11. A screenshot of the Flocking experiment. Here agents, representing by seagulls, can assume several patterns of flocking, which can be obtained by changing the three parameters in the interface.

This section excludes some experiments available in the distributed package designed purely for interface and physics tests. Other third part experiments, that employ the framework, are under development and shall be linked in the project page in the near future.

5. Conclusions and future works

One of the strongest appeals of Artificial Life is the possibility to turn any personal computer into an advanced experimental laboratory (Langton, 1995). Yet, the ability to implement such experiments lies on hands of researchers who domain the core concepts and some programming language to implement them. The visualization of experiments is another issue, once 3D visualization tools and libraries are unlikely easy to use, most of actual ALIFE software ends up by the old text outputs or poor 2D graphics to represent the data and processes happening in the virtual environment. The presented framework provides a quick experiment prototyping tool, with ready made high-end visualization capabilities and multi-agent infrastructure. Yet, by making the project available in Sourceforje.net, users can contribute for improving the framework itself. The experiments within the framework can be published through the Internet in interactive web pages, allowing quick feedback between researchers. Hereby it is

possible to any interested visitor to download the software and start his/her own experiments, allowing people outside the scientific community to profit on experimenting in the virtual laboratory, this way providing an alternate scientific divulgation tool.

The Java3D rendering engine is regularly being updated [14], allowing to the evolved experiments to take advantage of emerging visualization technologies without the need of regular code modifications.

Some of the possible applications for the framework are: ALIFE experiment development; Biologic processes demonstrations in Virtual Reality; Multi-Agent problem solving in sciences/engineering, in simulations applying genetic algorithms in virtual evolving systems; System training for robotics; Simulation of genetic/evolutionary systems; User-Assisted Agent-Oriented pattern search in multi-dimensional space data ensembles; Research in upcoming technologies (such nanotechnologies).

References

Bentley, Peter J. (Editor) (1999), "Evolutionary Design by Computers". Morgan Kauffmann.

Bedau et al. (2000), "Open Problems in Artificial Life". Artificial Life 6, MA, 363-376. http://mitpress.mit.edu/journals/ARTL/Bedau.pdf, access June, 2003.

Cavalhieri, M: "Virtual Human Project", responsible, personal web-site http://www.lsi.usp.br/~mac/, access January, 2003.

Cruz-Neira, C. et al. (1992), "The CAVE Audio Visual Experience Automatic Virtual Environment", Communication of the ACM, June, 35(6):64-72.

Langton, G. C. (1995), "Artificial Life: An Overview". MIT Press, Cambridge, MA.

Lejter, M., Dean, T. (1996), "A Framework for the Development of Multi-agent Architectures". IEEE Expert, (December) 47-61.

Miranda, Fabio R. et al (2001), "Arena and WoxBOT: First Steps Towards Virtual World Simulations". SIBGRAPI 2001 - XIV Brazilian Symposium on Computer Graphics and Image Processing, Florianópolis, Brazil (october) IEEE Computer Society Press.

Mitchell, M. (1997), "An Introduction to Genetic Algorithms", MIT Press.

Netto, M. L. et al (2000), "WOXBOT: a Wide Open Extensible Robot for Virtual World Simulations". GRAPHICON 2000 - 10th International Conference on Computer Graphics and Vision, Moscow, Russia (August).

Netto, M., "Artificial Life Group (ARTLIFE) Site", http://www.lsi.usp.br/~artlife/ "ALIVE Project site", http://www.lsi.usp.br/~alive/, access June, 2003.

Netto, M. L., Kogler Jr., J. E. (EDITORS) (2001), "Virtual Life – Towards New Generation of Computer Animation", Computers & Graphics – an international journal on computer graphics and applications, volume 25, Special Issue n. 6 (december), Elsevier Science

Neves, Rogério P. O. and Netto, Marcio L. (2002), “Evolutionary Search for Optimization of Fuzzy Logic Controllers”. 1st International Conference on Fuzzy Systems and Knowledge Discovery, Volume I, on Hybrid Systems and Applications I.

Neves, R.: ALIVE project site, the program, related articles and links to other ALife sites. http://www.lsi.usp.br/~rponeves/, access June, 2003.

Shloam, Y. (1998), “Agent-Oriented Programming”. Readings in Agents, Edited by M.N. Huhns e M.P. Singh, Morgan & Kaufmann, S. Francisco 329-349.

Sun Java site: Support, papers, tutorials and books related to Java and Java3D programming. http://java.sun.org/

Thalmann, N. M. (Editor), Thalmann, D (1994), “Artificial Life and Virtual Reality”. John Wiley & Sons; 1 edition (November 15)

Virtual Reality Core on Laboratory of Integrated Systems: CAVE facility specifications http://www.lsi.usp.br/~rv/, access June, 2003.

Wooldridge, M., Jennings, N. (1995), “Agent Theories, Architectures, and Languages: a Survey”,Wooldridge and Jennings Eds., Intelligent Agents, Berlin: Springer-Verlag, 1-22