Simulation Environment for Selected Parts of Artificial Life Domain

K. KOHOUT, P. NAHODIL
Dept. of Cybernetics
Czech Technical University in Prague
Karlovo náměstí 13, 121 35 Prague
CZECH REPUBLIC

Abstract: Our research is focused on the simulation of agent – animate. The architecture of these agents is mainly inspired by nature and therefore they are sometimes called artificial creatures. The main contribution of this paper is the description of designed simulation environment architecture for the Artificial Life (ALife) domain. It was named the World of Artificial Life (WAL). Our platform incorporates results of research in the domain of hybrid agent architectures. Based on this results it focuses on the proposal and implementation of the simulation environment. First, we formulate the problem we were solving and the important goals, then we present our proposal for the simulator, followed by brief overview of achieved results.

Key-Words: Anticipation, ALife, Hybrid Architecture, Behavior, Animate, Artificial creatures

1 Introduction

Nowadays there are a lot of freely available simulation platforms. Based on our analysis of current status of simulations for ALife domain [1] including our research group and others as well with the aim to find a platform capable of running required simulations, none of analyzed was not fully capable to satisfy our needs. We have found out that they are mainly focused just on one specific domain of Artificial Life. Almost none of them are capable of simulating Artificial Life on a more general level or provide a set of analytical tools to evaluate the simulation in a broader context. This led us to a decision to design our own simulation environment. The main goal was to develop a simulator on a high modularity level and simple enough to be usable by anyone interested in the ALife research. Our simulator helped us to focus on the studied topic while abstracting from implementation details of the environment itself. Special care has been applied to possibilities of analysis, either during the simulation or after the simulation from saved data. Visualization modules are covering not only displaying the simulated agent world in 3D, but are targeted on efficient analysis of agents' behavior. Visualization can provide simplified and an attractive view in order to present the simulation to a non-technical audience. It can also offer scientific views with various statistics and a value detail of the agent world. This view can satisfy needs of researchers for detailed analysis of behavior of the simulated system.

2 Designed Abstract Architecture

Our requirements for this platform were following.

- The ability to simulate various phenomena of Artificial Life from cellular automata, boids, bimorphs, ants colonies etc. up to complex and socially behaving agents.
- The ability to export inner data so it can be used for parameter visualization.
- Variability of simulation which can be easily modified in step by step run.
- Interesting visualization of agent world, in order to present the simulation to wider and non technical audience.
- Meaningful and helpful visualization of parameters in time.
- Modularity
- Easy extendibility
- Interoperability.

In order to implement such task a general abstract architecture was proposed and named WAL Abstract Architecture, WALA^2 in short. This should provide a general guide or instructions on how an application for simulation of Artificial Life should be defined and implemented with care for high interoperability and modularity between various implementations. This design was not the work of one person, but result of tight cooperation and many discussions of all MRG members. While designing this abstract architecture we kept in mind that our implementation can be superseded in the future by better ones, however if we will stick with the philosophy and recommendations of the abstract architecture, agents will be transferable with none or minor reprogramming and redesign. Another goal of WALA^2 is to assure that agents can also run and compete on other implementations of same architecture. The reason for this is the evaluation of results when different agents architectures. The approaches can be evaluated in base environment, or to simply test agents' behavior in different environment that it was designed for. We can observe if he can adapt and to new circumstances and survive.

WALA^2 itself defines modular block architecture of platform as it is shown on Fig. 1.

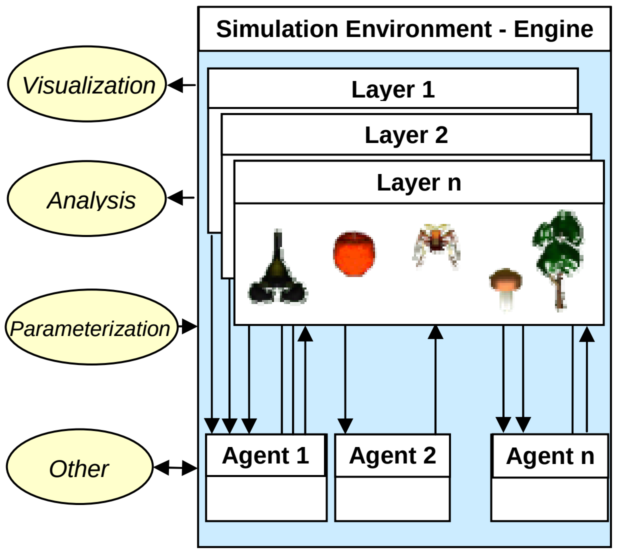


Fig. 1. Block scheme of WAL abstract architecture

This architecture was designed to enable easy parameterization of simulation and distributivity of its parts (body and mind can be separated and even run on different computation units). One of the benefits is separation of the environment into layers. This is not layered architecture in agent design, but is the environment design. This decomposition of environment leads to simpler and more comprehensive simulation and also gives the opportunity to describe more complex environments.

2.1 Platform - engine

The core part of the simulation environment will be referred to as the engine or platform. It is the base unit and it controls the run of the simulation on a program level. This means it synchronizes the whole application – gives impulses on start and end of each step. It contains an interface for modules and it contains and maintains its parts. There are two components of the environment, the layers and the agents. In the one simulation step the engine asks all layers to evaluate actions of all agents and environmental changes according to these actions. The distribution of evaluation to layers means distribution of simulation control so that each layer can run in different computation thread (on multiprocessor unit they also might run on different processors). The main data structure where parameters of all layers and agents are stored is maintained by engine. This data can be view or modified by agent actions or even by external modules. It is important to distinguish between the control part of the engine, which interacts mostly with the operating system (graphical interface, loading and saving configuration, user interaction etc.) and the part providing and simulating the virtual world for agents. The first is done by above described engine. The second function is described further in the text and is handled by layers.

2.2 Engine interface

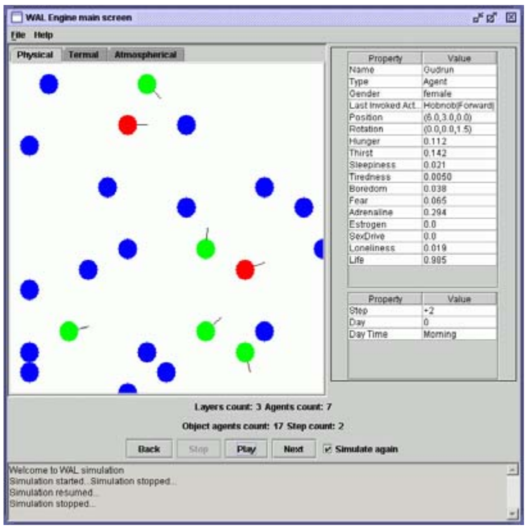
Interface between simulation environment and its program surrounding (as for example visualization, analysis tool or parameterization) is an important part of the application. This is what makes WAL modular and distributable. From speed of data processing point of view it is suitable to exchange information in binary format. It is also possible to use text based formats such as XML. The textual format is in principle highly redundant (but descriptive) and its processing could be slow. Still, it can be used for offline analysis. The engine contains all data (including the data of layers and the agents) in inner tree-based data structure. It can provide all of these data or just part of it to external modules. Each connected external module can ask for data and uses them for its purposes. The inner data representation is not defined in abstract architecture. It can be implemented in various ways and it does not matter as long as the interface for exchanging of this data remains the same. This interface should work in both directions for exporting the data to be read by external modules as well as for receiving updates of the data structure from modules. The running simulation can also be stopped at any moment and even traced back to certain point and run again to observe if any change of behavior will occur in exactly the same situation. Change of simulation parameters should be available while the simulation is running. As an agent we understand any object in simulation either virtually alive (creature, predator) or virtually non-living (trees, food, water, rocks). Sensors and effectors of the agent are their interface with virtual world and therefore they are part of environment and layers. Besides that agent mind and control are not part of the environment and can be also remote. The binary format is good for both, as it can be parsed very quickly.

2.3 Simulation world in layers

This is the part of the application directly interacting with the agent via its sensors and effectors. Layers define the virtual or simulated world in which agents live. It serves for logical and computational separation of operations which has to be controlled by the engine. The layer is a logically separable part of the environment which can be used as standalone and which, when combined with others, defines the environment as whole. All agents having influence in a particular layer or being influenced by this layer must be registered in it. This means that the layer has the full information for computing the next step. The layer will evaluate and execute agents' actions in each step. According to the executed actions of the agents the layer will modify its own values and then will provide a new sensoric data to agents. The layer must have the ability to register and deregister the agent and also fill the sensors of those registered agents. This means, that it must have an interface for communicating with sensors. For example the thermal layer should have the ability to provide information for sensors of temperature. In each step the layer must read agents executed actions, evaluate them (whether they are possible or executable), modify the environment and provide new senzoric data. Basically there were two types of layers defined. The point layer in which can the value be evaluated directly or can be obtained immediately from layers data. The gradient layer where the value in the point of interest cannot be evaluated just from the actual information but also the history of the value must be taken in account. The physical body of the agent and its sensors and effectors are also part of the environment, so it is necessary to interpret them in it. The layer must have the information how much space the body, sensors and effectors takes. This could enable building of more complex agent from basic blocks. The potential of the layers was then used in several works. The implementations had up to seven different layers [5]. The advantage of layers is that they can solve communication between agents in sense of "physical" distribution of the signal. We can implement the acoustic layer, and propagate the sound based on the physical laws. This solves the problem of transporting the message, not the understanding and context of the message. To sum up layers in one simple sentence, they implement various physical laws. Complete physical description is almost unachievable. Layers can bring us closer to it.

2.4 Human interface and analysis

All that described above is just an algorithm with no human interface. Visualization of designed world can be both attractive and a useful tool. For this purpose external visualization module or internal (default) can be used. Internal visualization is meant to debug and observe simulation by creator (Fig. 2.)



Property	Value
Name	Gudrun
Type	Agent
Gender	female
Last Invoked Act...	Hobnob(Forward)
Position	(6.0,3.0,0.0)
Rotation	(0.0,0.0,1.5)
Hunger	0.112
Thirst	0.142
Sleepiness	0.021
Tiredness	0.0050
Boredom	0.038
Fear	0.065
Adrenaline	0.294
Estrogen	0.0
SexDrive	0.0
Loneliness	0.019
Life	0.985

Property	Value
Step	+2
Day	0
Day Time	Morning

Layers count: 3 Agents count: 7

Object agents count: 17 Step count: 2

[✓] Simulate again

```
Welcome to WAL simulation
Simulation started...Simulation stopped...
Simulation resumed...
Simulation stopped...
```

Fig. 2. Example of internal visualization

The external module can be the exact opposite. It can be used to present this simulation to wider audience than the science community (Fig. 3). On-line or off-line tools for parameters flow analysis are also supported. Proposed environment is compatible with highly effective 3D visualization tool called VAT. It can be used to observe agents parameters at any time of simulation. I will not go into much detail about the parameter visualization problem. Information about it can be found in [4]. By using the third dimension for data visualization the analysis is more comprehensive and computer graphics knows various methods to visualize even more than thee dimensions. That is why the 3D analytical tools are strongly supported. This leads to many advantages, because the value of the parameter can be also mapped to shape, height, width or length or many others. Another advantage is possibility to use sensitivity analysis. Analytical tools provide offline or online evaluation of simulation together with fast orientation in complex situations. It also enables the possibility of backward analysis of interesting simulation. It can be also used to observe relations between senzoric inputs and executed actions (i.e. what action was triggered under when there was a specific sensory input and vice versa).

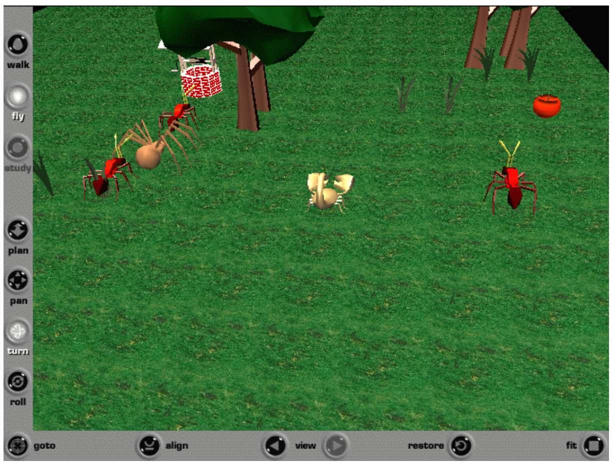


Fig. 3. Example of external 3D visualization

2.5 Influencing the simulation

Parameterization provides the ability to alter the simulation either as an initial setup of simulation or direct change to the simulation in runtime. This means changing of agents or layers parameters while the simulation is still in progress. For example, you can set a new target for agent, or decrease the temperature in a particular place. This also covers creation of new object (agent, food, etc…) while simulation is still running. This should provide the ability to run longer simulations, where users' intervention could alter it according to its state. Moreover, combined with online visual analytical tools, there should be possible to observe the interesting moments of the simulation and change the scenario, to see how the agents will adapt to this change. Pausing and resuming the simulation and tracing step by step (even backwards) are also part of parameterization. Backward run is one of the key elements which we are missing in simulations. This is useful to observe emergent behavior, when we can trace back the simulation to some interesting situation, run it again, and observe if the situation will end exactly the same as previous or it will differ.

2.6 Agent of WAL

WALA[2] separates the body of agent (physical representation of agent) from the mind (control mechanism). The agent's body is part of the environment and therefore it is covered here in environment architecture. The mind of an agent on the other hand communicates with the body through data from sensors. Please note that even agents own state has to be observed by sensors. This covers the state of agents sensors and effectors (some of them can be damaged or partial malfunctioning) and the Vegetative System Block. After agents sensors are filled with data, the body sends them to the mind of agent, where they are processed. How the data is processed is not a subject of this abstract architecture. Several approaches to agent mind design can be found in [2], [5], [6]. Finally, the mind evaluates the situation and selects the action or actions for execution. The body tries to perform these actions using its effectors. The layer mentioned above will then evaluate the cause of actions by setting new senzoric data. Agent's body is part of the environment and as such it has physical properties such as position, shape, temperature, etc… The decomposition and the interface is shown on Fig. 4.

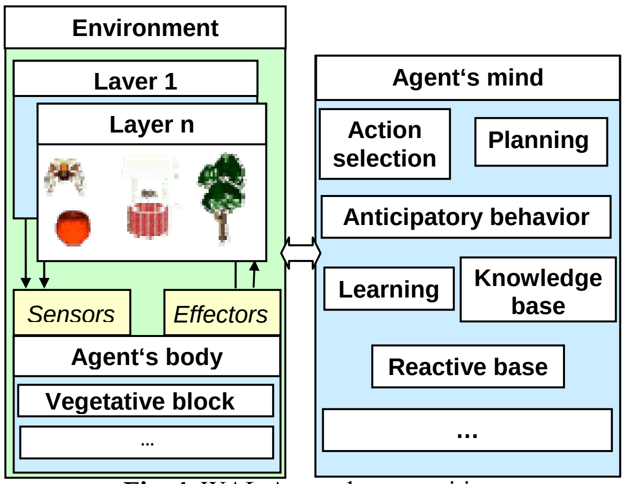


Fig. 4. WAL Agent decomposition

2.6.1 Agent's sensors

Sensors are the agent's senses which they use to perceive the surroundings and also its state. The richness of information about the surrounding world the agent can obtain depends just on the type and number of the sensors. Layers should know all types of possible sensors to be able to fill them with data. This means that creating a new layer necessarily requires also the creation of adequate sensors or altering the layer so it can work with current sensors and vice versa. When adding a new sensor it is also necessary to alter the layers so they know this sensor and are able to fill it. Sensors are the part of an environment containing exact data (the numerical value). It is not always wanted to provide this data to agent's mind. Human beings are also not able to say, "That building is 354.56 meters from me". Rather a human being can say it is far/near. In addition we can guess "it might be 350 to 400 meters". To simulate this kind of perception even in an agents world, we want to implement such fuzzy information rather than crisp values. This can be done by filtering the exact floating point value to fuzzy value.

2.6.2 Agent's effectors

Effectors serves agent for interaction with its surroundings. We use simplified effectors. For example we use effectors of motion which can move agents in certain direction with certain speed. Of cause we could go deeper in detail and implement effectors such as leg or wheel, but this would distract us (by solving inverse and forward kinematics tasks etc…) from observing the behavior. We do not require this level of detail, but we are not running away from it, and the possibility to implement it is still open. In the field of effectors there is space for improvement. Instead of using effectors as part of the environment and controlling their action, we implement the action result. In the movement example above, we move the agent from one position to another, instead of sending a signal to agent's locomotion system. This again requires implementation of physical laws such as friction.

2.7 Communication

Communication between the agent body and the layers is internal communication so there is no need for explicit data sending. This communication can be done via the internal data structure, which is in this case, the shared medium. The communication between body and mind is in general done via messages. It can be done even remotely via various media (for example over TCP/IP see Fig. 5). Communication on the agent level means sending a message from one agent to another or group of agents. Here we would like to let layers decide who receives the information and who not. We are trying to reflect the real world behavior where information is carried via different media and can be received by various entities based on their sensor capabilities. When one agent wants to send a message ("say something") to another it will use its effectors and certain media (acoustic wave). The information can be then received not only by the addressee but also by another agent who is in the range of the signal (even if it did not request this information). It can disregard it, or abuse it for its own purposes.

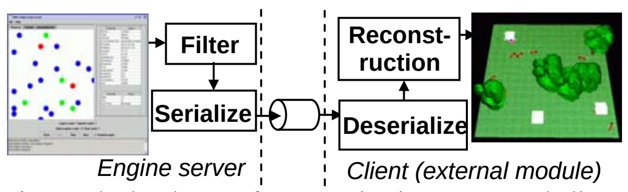


Fig. 5. Block scheme of communication to external client

3 Simulations and results

We mentioned already above that there are two components to run the simulation. First of them is the hereby described environment, second is the control of the agent itself (agent mind). There has been several agent behavior control architectures introduced. The basis for the agent architecture was designed by D. Kadleček [2], [3]. This agent architecture with was redesigned for WAL environment by K. Kohout in [1]. Several simulations, concluding the Lotka-Volterra (also know as predator-prey) system (see Fig. 6) and several task oriented scenarios were tested.

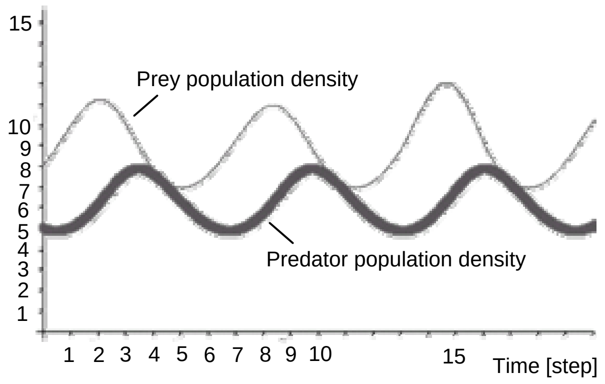


Fig. 6. Lotka-Volterra simulation results

In second simulation a task was given to agent. This means that agent except assuring his survival should complete another task. In our case it was delivery of messages. This simulation served as a test of the thresholds set correctly. In case the agent's "need" to fulfill the task was low it was focusing almost only on his survival. In case it was high the agent was busy with his task and he fulfilled his survival needs only when needed.

3.1 Case study

We mentioned usefulness of external modules namely for simulation analysis. We would like to prove this on a case study performed while redesigning the agent to WAL environment. The above mentioned VAT tool was used for the analysis. The simulation scenario concludes a single agent which was intended to move an object between two places. The agent had enough food and water to satisfy its needs. Fig. 7 shows the visualized data from this simulation. On the left there is a 3D mesh; on the right is a detail of values in the 60th step. Even a very first look at the 3D mesh could advise that there is something wrong with the simulation. Almost all of parameters are zero (the mesh is flat).

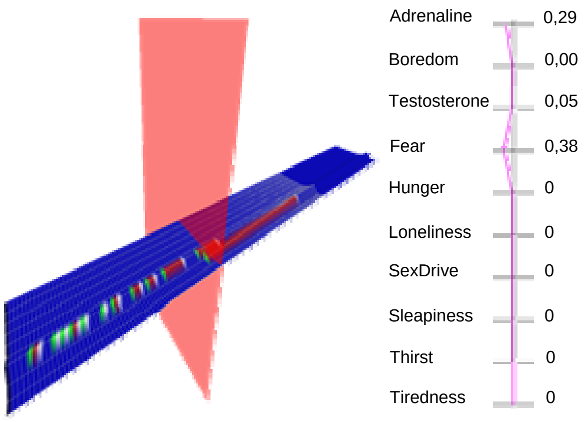


Fig. 7. Use case – simulation analysis

This means that the agent is not hungry, thirsty; it is neither tired nor sleepy. But we implemented and designed all these features. The reason for this could be a data export failure, a mistake in implementation of the inner agent vegetative block (part taking care of "chemicals" in an agent's body) or a bad initial configuration of an agent. Because we run the simulation previously and export of data and the vegetative block were working properly, there is no problem with implementation itself. Brief check of the configuration showed that - there was to high value set to the time function for the chemical increasing/decreasing. Fig. 8 shows the mesh after the configuration mistake correction. Values are now changing with the time.

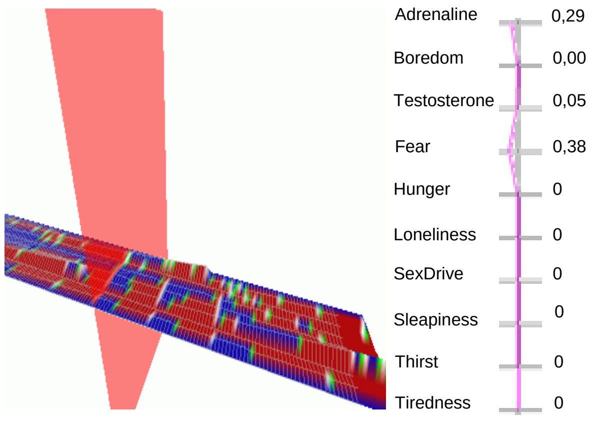


Fig. 8. Use case – simulation analysis - correct setup

4 Conclusion

In this paper, we have described the simulation environment architecture. It was used by several agent architectures to place the agent into. The first agent architecture tested in this environment was described above in section 3. This agent architecture was superseded by several other architectures namely Lemming, designed by L. Foltýn [5], ACS proposed by M. Mach [6] and AnimatSim introduce by A. Svrček [7]. They were focused on different topics or different approaches to agent learning. AnimatSim is focused on reinforced learning, Lemming uses the TDIDT algorithm to create a knowledge about environment and reason about it. ACS uses the Hidden Markov models to interpret the environment and reinforced learning to adapt to it. They have one thing in common. They were designed in various programming languages C, Java, Matlab. The communication with WAL was theoretically described but never fully implemented in them. Therefore the competition was not realized. This will be our focus in the near future.

References:

[1] Kohout, K. Simulation of Animates Behavior. Diploma thesis. Prague: Czech Technical University in Prague, Faculty of Electrical Engineering, Department of Cybernetics, 2004.

[2] Kadleček, D., Nahodil, P. New Hybrid Architecture in Artificial Life Simulation. In Lecture Notes in Artificial Intelligence No. 2159, Berlin: Springer Verlag, 2001. s. 143-146.

[3] Kadleček, D. Simulation of an Agent – Mobot in a Virtual Environment. Diploma thesis. Prague: Czech Technical University in Prague, Faculty of Electrical Engineering, Department of Cybernetics, 2001.

[4] Kadleček, D., Řehoř. D., Nahodil, P., Slavik, P., Kohout, K. Transparent visualization of multi-agent systems. In Proceedings of 4th International Carpathian Control Conference, 26.-29. 5. 2003, Vysoké Tatry. pp. 723 – 726. ISBN 80-7099-509-2.

[5] Foltýn, L. Realization of Intelligent Agents Architecture for Artificial Life Domain. Diploma thesis. Prague: Czech Technical University in Prague, Faculty of Electrical Engineering, Department of Cybernetics, 2005.

[6] Mach, M. Data mining knowledge mechanism of environment based on behavior and functionality of it's partial objects. Diploma thesis. Prague: Czech Technical University in Prague, Faculty of Electrical Engineering, Department of Cybernetics, 2005.

[7] Svrček, A. Selection and evaluation of robots-animates behavior. Diploma thesis. Prague: Czech Technical University in Prague, Faculty of Electrical Engineering, Department of Cybernetics, 2005.