In Proceedings of Computer Graphics International (CGI), July 3-6, Hong Kong, 2001

Towards Real Time Virtual Human Life Simulations

Etienne de Sevin, Marcelo Kallmann and Daniel Thalmann

EPFL Computer Graphics Lab – LIG
CH-1015 – Lausanne – Switzerland
{esevin, kallmann, thalmann}@lig.di.epfl.ch

Abstract

This paper describes an approach to construct interactive virtual environments, which are suitable for the development of artificial virtual human life simulations. Our main goal is to have virtual human actors living and working autonomously in virtual environments. In our approach, virtual actors have their own motivations and needs, and by sensing and exploring their environment, an action selection mechanism is able to determine anytime the suitable actions to take. We adapt basic actor motivations and needs to urban situations, where most actions involve interactions with the environment. Thus, a specific technique to define actor-object interactions is used, where pre-defined interaction plans are put inside interactive objects, and just selected during the simulation. We explain in this paper the steps taken in order to construct and animate such environments, and we also present a test simulation example.

Keywords: Artificial Life, Agents, Virtual Humans, Virtual Environments, Behavioral Animation, Classifier Systems, Object Interaction, Python.

1 Introduction

Virtual human simulations are becoming each time more popular, and many systems are available targeting several domains, as: autonomous agents, human factors analysis, training, education, virtual prototyping, simulation-based design, and entertainment. As an example, an application to train equipment usage using virtual humans is presented by Johnson et al [1].

Simulations with autonomous virtual humans, or *actors*, may use different techniques for the behavioral programming. Most common approaches are based on scripts [2] and hierarchical finite state machines [3], but many other techniques exist, as the *parallel transitions network* [14].

Such techniques are powerful and may serve to define a large range of behaviors. However, achieving complex and emergent autonomous behaviors will always be a difficult and challenging task.

We show in this paper how we construct interactive virtual environments, which are suitable for autonomous actors simulations. Our main goal is to have actors living and working autonomously in virtual environments, according to their own motivations and needs. We think that behavioral programming techniques should operate on the motivational level of an actor, and not directly control actions.

We focus on common-life situations, where the actor senses and explores his environment, and following an action selection mechanism, determines the suitable actions to take. Actions often involve object interaction, and so a specific technique to model actor-object interactions is used, following the *smart object* approach [4]. Smart objects contain interactivity information based on pre-defined interaction plans, which are defined during modeling phase.

We construct our interactive virtual environment using the *Agents Common Environment* (ACE) system [13], which provides the basic requirements for the implementation of autonomous actors simulations:

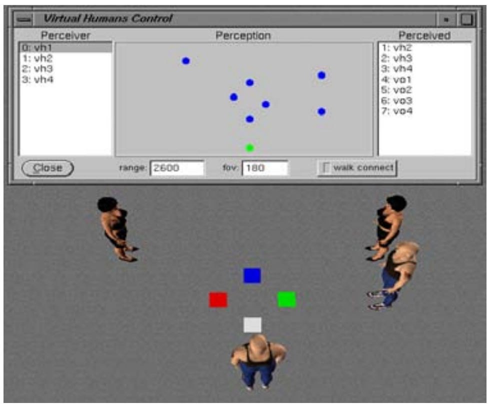
• Load and position different actors and smart objects.

• Apply an action to an actor, as: walking [7], inverse kinematics [10], facial expressions, etc. Actions can be triggered in parallel and are correctly blended, according to given priorities, by a specific internal synchronization module [8].

• Trigger a smart object interaction with an actor. Each smart object keeps a list of its available interactions, which depends on the object internal state. Each interaction is described by simple plans that are pre-defined with the use of a specific graphical user interface application called *somod*. These plans describe the correct sequence of actions and objects movements required to accomplish an interaction.

• Query *pipelines of perception* [9] for a given virtual human. Such pipelines can be configured in order to simulate, for example, a synthetic vision. In this case, the perception query will return a list with all objects perceived inside the specified range and field of view. As an example, figure 1 shows a map constructed from the results of the perception information received by an agent.

• Easy connection with external behavioral modules through Python scripts [5]. Python permits to create a re-use script as behavioral plug-ins, following a current trend in behavioral animation [6].



range: 2600
fov: 180

Fig. 1. Perception map of the lowest agent in the image. In this example, a range of 2.6 meters and a field of view of 180˚ is used. The darker points in the map represent the positions of each perceived agent and object.

We have thus implemented in Python a motivational action selection model, which permits to use internal actor motivations and environment information in order to select which actions and object interactions to take.

Following this architecture, the action selection algorithm works on a very high level layer, and ACE guarantees the smooth control of low-level motions, as walking and interacting with objects.

In the following sections we explain our action selection model, and show how we have built and used smart objects with coherent interaction capabilities.

2 The Action Selection Model

Different models for the action selection problem have been proposed: Tinbergen's [15] or Baerends's hierarchical decision structure (HDS) [16], Lorenz's hydraulic model [17], Rosenblatt & Payton's [18] free flow hierarchy or associated Maes's [19] network. Tyrrell has tested these models in complex simulated environments with many motivations and close to a wild animal environment. The result shows that the free flow hierarchy improved by Tyrrell [11] is the most suitable mechanism of action selection (figure 2).

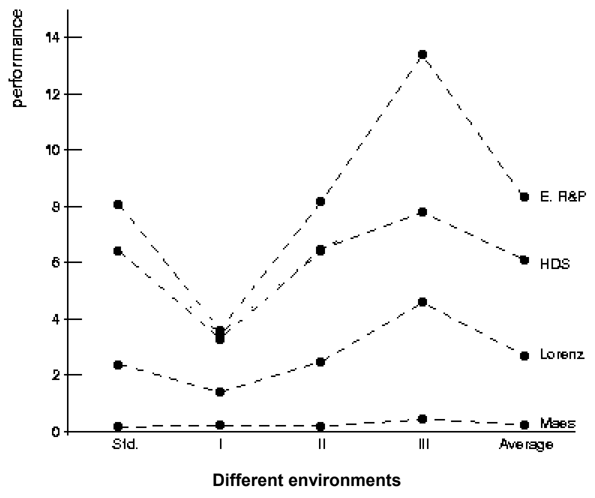


Fig. 2. Results of Tyrrell's test. For the robust results, different models of action selection model have been tested with many motivations in different environments (standard, I, II, III, and then averaged).

From his test, Tyrrell define six criteria to respect in a mechanism of action selection:

- Take into account motivations.
- Take into account environment information.
- Prefer to choose physiological actions over locomotion actions.
- Continue the current sequence of actions until the end.
- Stop the current sequence of actions if another motivation is urgent to satisfy.
- Prefer *compromise behaviors*, i.e., which respective actions satisfy more motivations.

We have implemented in Python a mechanism of action selection based on a free flow hierarchy [11] associated to a hierarchical classifier [12].

The free flow hierarchy respect Tyrrell's criteria and permits to take into account different types of motivations and also information coming from the environment perception. The key idea is that, during the propagation of the activity in the hierarchy, no choices are made before the lowest level in the hierarchy represented by the actions is reached.

The hierarchical classifier provides a good solution to model complex hierarchies by reducing the search domain of the problem, and using rules with weights. Also, we can easily make behavioral sequences (composed by a sequence of actions). So, the virtual human can move to a specific place to perform a physiological action and satisfy the motivations no matter where it is.

The hierarchy of our model contains four levels (figure 4):

1. Internal variables evolving linearly during time, representing different states of the virtual human.

2. Motivations corresponding to a “subjective evaluation” of the internal variables. A threshold system can permit to reduce or enhance motivations values according to the internal variables values. Whenever an internal variable gets high, the actor will rapidly react in order to satisfy the related motivation. Figure 3 shows the result obtained at this level.

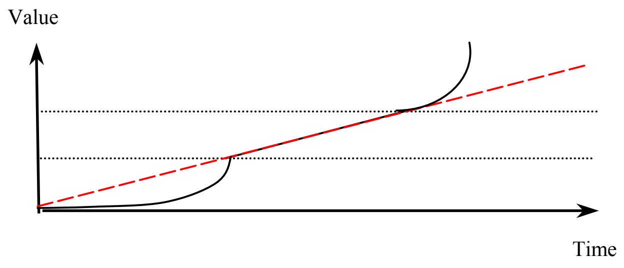


Fig. 3. “Subjective” evaluation of the motivations (solid curve) from the value of the internal variables (dashed line) with a threshold system.

3. Motivated behaviors having the only goal to generate actions to move the actor to a place where it can satisfy the correspondent motivation.

4. Actions influencing directly the internal variables. Locomotion actions increase the internal variables, while the physiological actions decrease them. If the actor has a choice between these two kinds of actions, it will choose a physiological one, as it is more beneficial for him. Indeed, the rule weight for physiological actions equals twice the rule weight of a locomotion action in the model, and in all the cases the system chooses always the most activated action at each iteration.

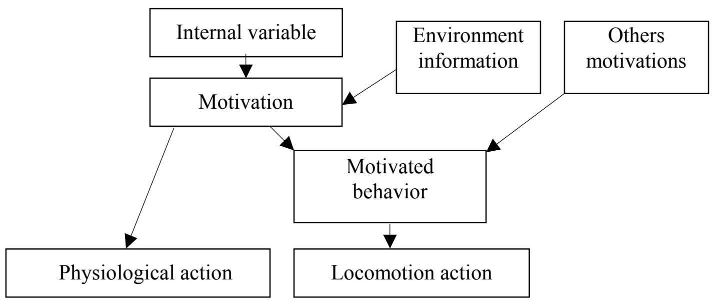


Fig. 4. Simplified motivational model of action selection for virtual humans

The six criteria of Tyrrell have been respected. First, the motivations and the environment information have been taken into account in the model. Secondly, the physiological actions are preferably chosen compared to locomotion actions due to the difference of their rule weights. Motivated behaviors satisfy motivations by generating behavioral sequences of actions. The actor can move to a specific place to perform a physiologic action. However, the system chooses always the most activated action, that’s why if a motivation becomes urgent, surmounting the current motivation, the system switches to perform the one with the higher priority. At the end, the motivated behaviors can summarize the activities from different motivations (compromise behaviors), and their chances to be chosen increase because their values are higher.

In another words, the main role of the action selection mechanism is to maintain the internal variables under the thresholds by choosing the correct actions. Actions involving interactions with smarts objects are preferably chosen because they are defined to be directly beneficial for the virtual human. Otherwise, the virtual human is instructed to walk and reach the place where the motivation can be satisfied.

Take as an example the eat motivation depicted in figure 5. The behaviors "go to known location" or "go to a visible food" control the actor displacement to a specific direction, using the low level action of walking.

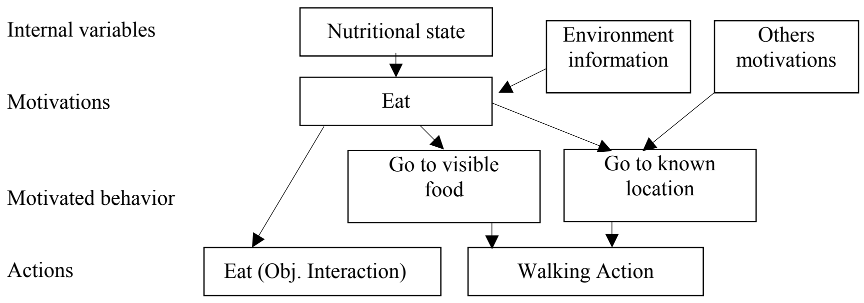


Fig. 5. A part of the hierarchical decision graph for the eat motivation, which is evaluated on each iteration. Some compromises can be made with the others motivations

The simulation being presented in this paper uses five main motivation types: eat, drink, rest, work, and go to the toilet. However, our action selection mechanism is not limited in the number of motivations and can thus be easily extended. During the simulation, the model is fed with parameters describing the current state of the actor concerning each of the motivations, and by flowing inside the hierarchical structure, will correctly trigger the concerned actions.

After an action is selected as a response to satisfy one of the actor's motivations, the state parameter of the internal variable is adapted accordingly. For example, after the action of eating is completed, the "Nutritional state" parameter will decrease. In this way, each action needs to be associated to an internal variable, closing the loop: motivation parameter evaluation, action selection, action animation, and internal variable parameter adjustment.

We have then defined five smart objects directly related to each motivation: a hamburger, a cup of coffee, a sofa, a desktop computer, and a toilet. These objects are shown in the next section.

3 Modeling the Required Smart Objects

As already mentioned, each motivation of the actor is directly related to an object interaction. Even for the motivations of resting, eating and drinking, we have created smart objects, containing interactions to eat, drink and rest. The advantage to define such simple interactions with smart objects is that we can easily define the movements of reaching, grasping and taking objects to eat, or, for instance, to control the virtual human's skeleton to sit in a sofa. All motions are internally controlled in ACE with inverse kinematics.

Figure 6 and 7 show snapshots of the modeling phase of some of the used smart objects. In the figure, it is possible to note the many geometric parameters, which are referenced from the interaction plans, to initialize and control the actions inside ACE.

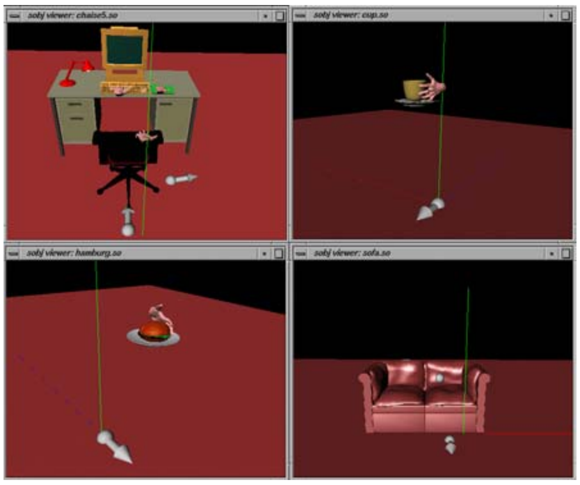
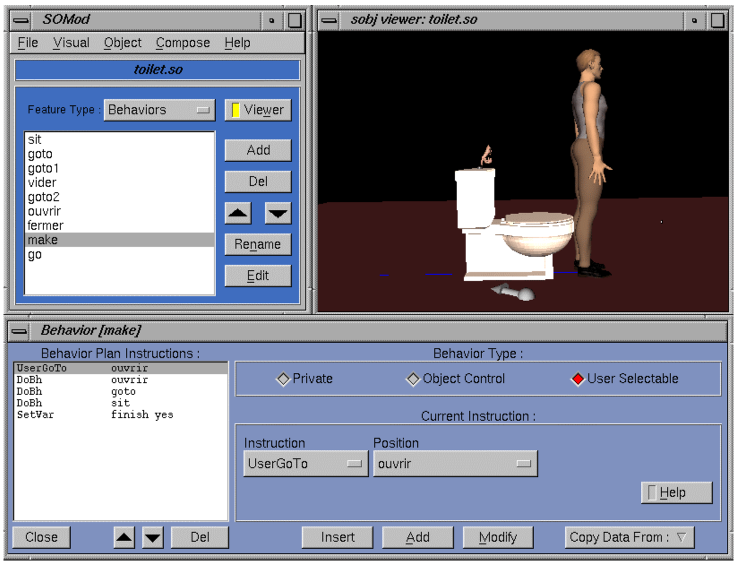


Fig. 6. Modeling phase of some used smart objects: the desk used to work, the sofa used to rest, and the hamburger and the cup of coffee, which can be taken to satisfy the needs of drinking and eating.

For example, regarding the hamburger and the coffee, the interaction consists of moving the object to the actor's mouth, using inverse kinematics. The following table lists each used modeled smart object, with their main interaction capability:

Action	*Smart Object*	*Interaction*
Eat	hamburger	eat
Drink	a cup of coffee	drink
Resting	sofa	sit
Work	computer and desk	sit and type
go to the toilet	toilet	use

Somod (the smart object modeler) is used to create each smart object, and in particular to define the behavioral and interaction information. For instance, figure 7 shows a programming session with somod, to define the interaction plans used for the toilet model.



Feature Type : Behaviors

sit
goto
goto1
vider
goto2
ouvrir
fermer
make
go

Behavior Plan Instructions :

```
UserGoTo     ouvrir
DoBh         ouvrir
DoBh         goto
DoBh         sit
SetVar       finish yes
```

Behavior Type : Private Object Control [✓ User Selectable]

Current Instruction :

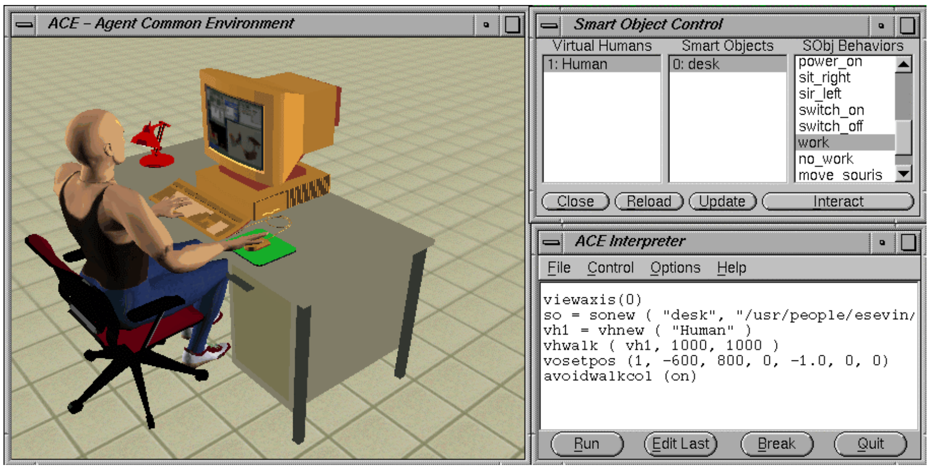
Instruction: UserGoTo

Position: ouvrir

Fig. 7. Modeling the interaction plan of a "simplified toilet".

From ACE, only the given name of each interaction is seen and available for selection. When smart objects are loaded, the system only exposes the possible interactions of each object, hiding the internal interpretation of the interaction plans from the user, which is transparently executed by ACE.

Figure 8 shows the ACE system being used to test the interaction capabilities of the modeled desk. The user can easily select (from Python, or from an user interface) the interaction plans available in the object.



```
viewaxis(0)
so = sonew ( "desk", "/usr/people/esevin/
vh1 = vhnew ( "Human" )
vhwalk ( vh1, 1000, 1000 )
vosetpos (1, -600, 800, 0, -1.0, 0, 0)
avoidwalkcol (on)
```

Fig. 8. Testing the interaction capabilities of the desk model.

4 Obtained Simulation in ACE

The action selection model was entirely developed in Python. The script makes use of two threads: one controlling the graphical refresh of the environment, and one controlling the continuously evaluation of parameters in the hierarchical action selection model.

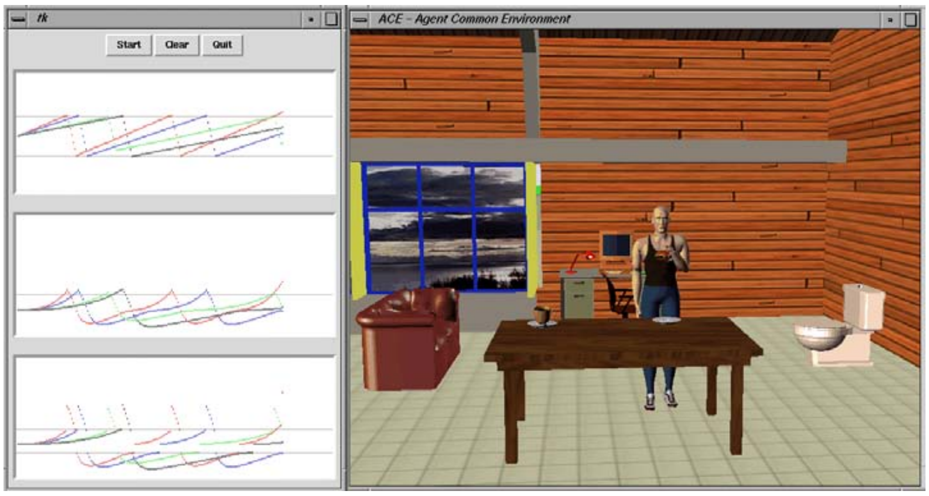


Fig. 9. A snapshot of the achieved simulation. The curves on the left show the variation of the internal motivational parameters of the virtual human, at different levels in the hierarchy.

The obtained simulation shows the virtual human actor living autonomously in the virtual environment, as exemplified in figure 9. It also shows the variation of the motivational parameters at different levels in the hierarchical selection model. The first graph (top left) shows the linear evolution of the internal variables during the simulation time due to the effects of actions. The second graph shows the "subjective" evolution of the motivations according to the threshold system and environment information. The last graph shows the evolution of physiological and locomotion actions according to their associated rule weights.

When the simulation starts, the actor has the initial behavior to explore the environment, collecting perceived information regarding the position of useful objects, like the hamburger and the coffee. After some time, the energy level drops, and the action to eat or drink is selected, according to the evaluation of the action suitability to perform. Other parameters regarding resting or the need to go to the toilet also change, controlling the actor accordingly. Whenever there are no motivations to satisfy, the actor will then go to work (default behavior).

5 Concluding Remarks

We have shown in this article our approach to construct complex autonomous actors simulations, which is based on three main steps: definition of the required actor-object interactions, modeling of the interactive smart objects, and simulation inside ACE using Python scripts.

The advantages of our approach are mainly due to the modularity achieved, specially regarding the separation of the object interaction information from the behavioral action selection module, which is programmed in a high level scripted language as Python. In particular, such modularity allows the parallel development of low level animation algorithms and behavioral algorithms without many conflicts.

As most of the action selection models are based on animal observations, we have shown in this paper our first results about translating these models to the urban human life.

Many enhancements are being done to this system in order to complete our behavioral platform. For instance, we intend to integrate the following modules:

- a navigation planning module that will enable us to perform tests with simulations in much larger environments.
- a training module based on the rule weights for a better adaptation of the virtual human with his environment.
- a multi-agent module to manage interactions between virtual humans.
- a emotional module to improve the communication with the real humans.

6 Acknowledgments

This research was supported by the Swiss National Foundation for Scientific Research.

7 References

1. W. L. Johnson, and J. Rickel, "Steve: An Animated Pedagogical Agent for Procedural Training in Virtual Environments", Sigart Bulletin, ACM Press, vol. 8, number 1-4, 16-12, 1997.

2. K. Perlin, and A. Goldberg, “Improv: A System for Scripting Interactive Actors in Virtual Worlds”, Proceedings of SIGGRAPH’96, 1996, New Orleans, 115-126.

3. Motivate product information, Motion Factory web address: http://www.motion-factory.com.

4. M. Kallmann and D. Thalmann, “A Behavioral Interface to Simulate Agent-Object Interactions in Real-Time”, Proceedings of Computer Animation 99, IEEE Computer Society Press, 1999, Geneva, 108-146.

5. M. Lutz, “Programming Python”, Sebastapol, O’Reilly, 1996.

6. N. Badler. "Animation 1100++", IEEE Computer Graphics and Applications, January/February 1100, 28-29.

7. R. Boulic, N. Magnenat-Thalmann, and D. Thalmann, “A Global Human Walking Model with Real Time Kinematic Personification”, The Visual Computer, 6, 344-358, 1990.

8. R. Boulic, P. Becheiraz, L. Emering, and D. Thalmann, “Integration of Motion Control Techniques for Virtual Human and Avatar Real-Time Animation”, In Proceedings of the VRST’97, 111-118, 1997.

9. C. Bordeux, R. Boulic, and D. Thalmann, “An Efficient and Flexible Perception Pipeline for Autonomous Agents”, Proceedings of Eurographics '99, Milano, Italy, 23-30.

10. P. Baerlocher, and R. Boulic, “Task Priority Formulations for the Kinematic Control of Highly Redundant Articulated Structures”, IEEE IROS’98, Victoria, Canada, 1998, 323-329.

11. T. Tyrrel, “Defining the Action Selection Problem”, Proceedings of the Fourteen Annual Conference on Cognitive Science Society”, Lawrence Erlbaum Associates, 1993.

12. J. Y. Donnart, and J. A. Meyer, “Learning Reactive and Planning Rules in a Motivationally Autonomous Animat”. IEEE Transactions on Systems, Man, and Cybernetics, part B: Cybernetics, 26(3), 381-395, June, 1996.

13. M. Kallmann, J. Monzani, A. Caicedo, and D. Thalmann, “ACE: A Platform for the Real Time Simulation of Virtual Human Agents”, EGCAS’1100 - 11th Eurographics Workshop on Animation and Simulation, Interlaken, Switzerland, 1100.

14. R. Bindiganavale, W. Schuler, J. Allbeck, N. Badler, A. Joshi, and M. Palmer, “Dynamically Altering Agent Behaviors Using Natural Language Instructions”, Proceedings of the 4th Autonomous Agents Conference, Barcelona, Spain, June, 293-300, 2000.

15. N, Tinbergen, The Study of instinct, Clarendon Press, 1951.

16. G.P, Baerends, “The Functional Organization of Behaviour”, Animal Behaviour, Vol. 24, pp. 726-738, 1976.

17. K, Lorenz, Foundations of Ethology, Heidelberg: Springler-Verlag, 1985.

18. J.K. Rosenblatt and D.W. Payton, “A fine-grained alternative to the subsumption architecture for mobile robot control”, In Proceedings of the IEEE/INNS Internationnal Joint Conference on Neuronal Networks, 1988.

19. P. Maes, “Bottump-up mechanism for behavior selection in an artificial creature”, In Meyer J.A. & Wilson S.W. (Eds). Proceedings of the First Internationnal Conference on Simulation of Adaptive Behavior, MIT Press/Bradford Books, 1991.