In Proceedings of the Workshop on Deformable Avatars, Lausanne, Switzerland, 2000, 240-247

CONSTRUCTING VIRTUAL HUMAN LIFE SIMULATIONS

Marcelo Kallmann, Etienne de Sevin and Daniel Thalmann
Swiss Federal Institute of Technology (EPFL), Computer Graphics Lab (LIG), Lausanne, Switzerland, CH-1015, {kallmann, desevin, 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 the suitable actions to take. Such actions often involve interaction with the environment and 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, Object Interaction, Python.

1. INTRODUCTION

Virtual human simulations are becoming each time more popular. 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 their behavioral programming. Common approaches are based on scripts [2] and hierarchical finite state machines [3].

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 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.

We have thus implemented in Python a motivational action selection model [11], 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 show how we have built smart objects with coherent behavioral information, and how they are coherently linked to our action selection model.

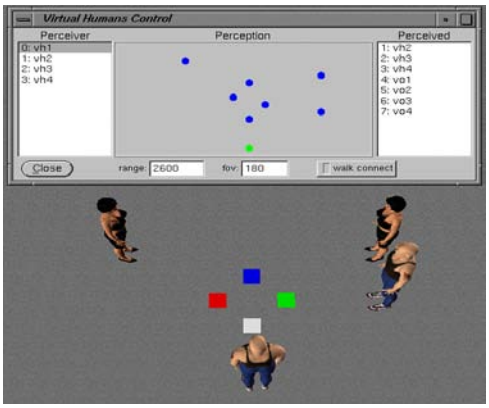

Figure 1. Perception map of the lowest agent in the image. In this example, a range of 2.6 meters and a field of view of 180° is used. The darker points in the map represent the positions of each perceived agent and object.

2. THE ACTION SELECTION MODEL

We have implemented in Python a motivational model for the action selection problem specifically for virtual human actors. This model is composed of a free flow hierarchy [11], associated to a hierarchical classifier system [12]. Such a model permits to take into account different types of motivations and also information coming from the environment perception. During the propagation of the activity in the hierarchy, no choices are made before the lowest level in the hierarchy represented by the actions.

Motivations correspond to a "subjective evaluation" of internal variables. When such variables pass over a threshold, the motivation becomes stronger. This generates a higher activity being propagated through the hierarchy, and the actions having more influence to reduce the motivation have more chance to be chosen. The system chooses always the most activated action at each iteration.

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 reach the place where the motivation can be satisfied.

Take as an example the eat motivation depicted in figure 2. 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.

Note also that distinct motivations can control the same action, and in this case their influences are added.

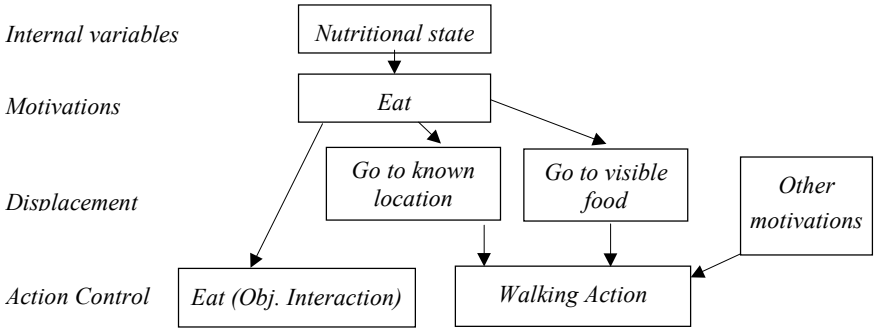

Figure 2. A part of the hierarchical decision graph for the eat motivation, which is evaluated on each iteration.

The simulation being presented in this paper uses five main motivation types: eat, drink, rest, work, and go to the toilet. The action selection mechanism is then fed with the parameters describing the current state of the actor concerning each of these 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 motivation is adapted accordingly. For example, after the action of eating is completed, the "hungry" parameter will decrease. In this way, each action needs to be associated to a motivation, closing the loop: motivation parameter evaluation, action selection, action animation, and motivation parameter adjustment.

We have then defined five smart objects directly related to each motivation: a hamburger, a glass of water, a sofa, a desktop computer, and a toilet. The construction of these objects is described in the next section.

3. MODELING OF 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 3 shows a snapshot 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.

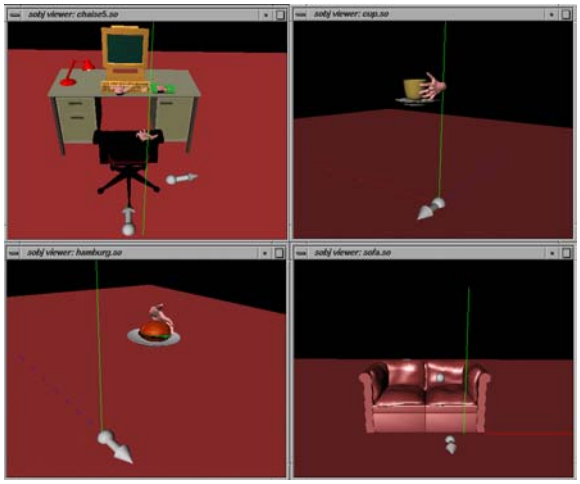

Figure 3. Modeling phase of some used smart objects.

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 is used to create each smart object, in particular to define the behavioral and interaction information. For instance, figure 4 shows the programmed interaction plans used for the toilet model.

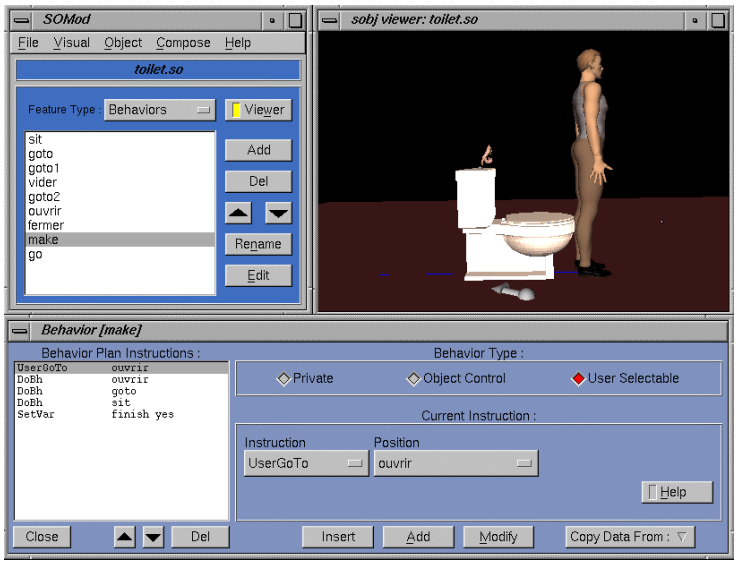

Figure 4. Modeling the interaction plan of a “simplified toilet”.

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 5 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.

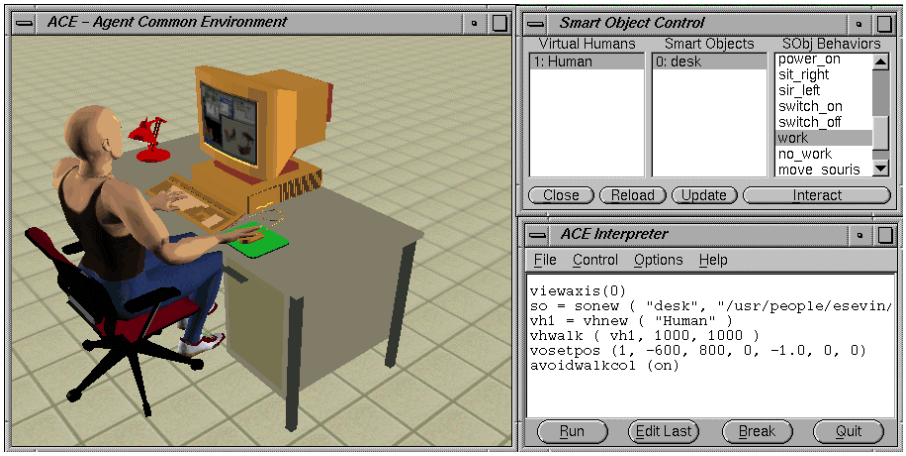

Figure 5. Testing the interaction capabilities of the desk model.

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.

The obtained simulation shows the virtual human actor living autonomously in the virtual environment, as exemplified in figure 6.

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 working, resting or the need to go to the toilet also change, controlling the actor accordingly.

Figure 6 also shows the variation of the motivational parameters at different levels in the hierarchical selection model.

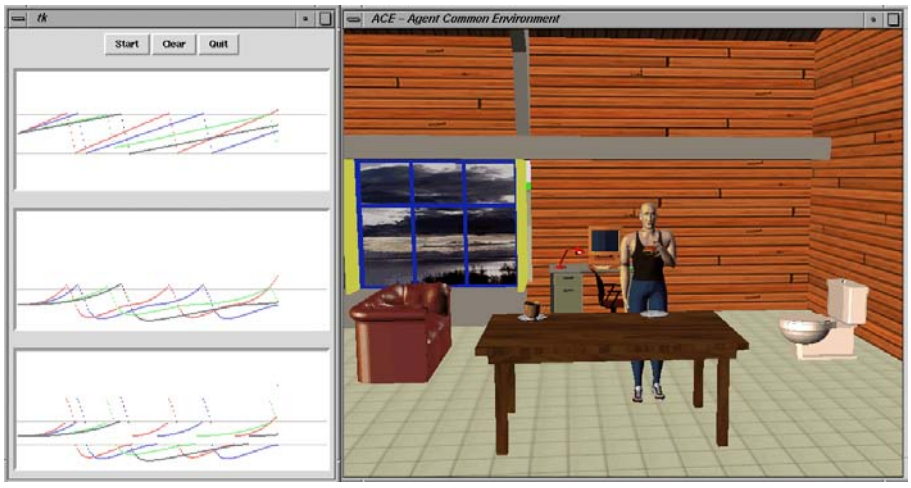

Figure 6. A snapshot of the achieved simulation. The curves on the left show the variation of the internal motivational parameters of the virtual human, at different levels in the hierarchy.

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.

Many enhancements are being done to this system, as for instance, the integration of a complete navigation planning module that will enable us to perform tests with simulations in much larger environments, and also with many actors at a same time.

6. ACKNOWLEDGMENTS

This research was supported by the Swiss National Foundation for Scientific Research and by the Brazilian National Council for Scientific and Technologic Development (CNPq).

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 2000”, 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.