Autonomous Actors in Networked Collaborative Virtual Environments

Igor S. Pandzic[1], Tolga K. Capin[2], Elwin Lee[1],

Nadia Magnenat Thalmann[1], Daniel Thalmann[2]

[1] MIRALab - CUI

University of Geneva

24 rue du Général-Dufour

CH1211 Geneva 4, Switzerland

{Igor.Pandzic,Nadia.Thalmann}@cui.unige.ch

http://miralabwww.unige.ch/

[2] Computer Graphics Laboratory

Swiss Federal Institute of Technology (EPFL)

CH1015 Lausanne, Switzerland

{capin, thalmann}@lig.di.epfl.ch

http://ligwww.epfl.ch/

Abstract

Introducing seemingly autonomous virtual beings into virtual environments to co-habit and collaborate with us is a continuous challenge and source of interest. Latest proof of human excitement for virtual life is the current world-wide craze for electronic pets that must be fed and cared for lest they develop a bad character or die. Even more interesting is the inclusion of autonomous actors in Networked Collaborative Virtual Environments (NCVEs). They provide a meeting place for people from different geographical locations and virtual beings. In NCVEs we don't see our correspondents, only their graphical representations in the virtual world, same as for the virtual ones - therefore the communication with virtual beings can come naturally. There is no single solution to the simulation of autonomous behavior. This is an ongoing research topic. Therefore it is interesting to provide an open NCVE system for easy interfacing with various implementations of autonomous behavior. In this way, the system can serve as an application platform with existing algorithms, as well as a research testbed for new autonomous behavior algorithms.

This paper studies the requirements for such an open interface and, based on this study, presents an implementation within the Virtual Life Network (VLNET) system. Results are presented in terms of two case studies: a simple one implementing a dumb servant character and a more complex one connecting VLNET with the autonomous agent program Eliza [Weizenbaum66].

Our study is based on the assumption that the implemented autonomous actors are to be human-like. However, a lot of principles outlined here can be applied to non-human virtual beings.

Keywords: Networked Collaborative Virtual Environments, Virtual Humans, Autonomous Behaviors, Virtual Life

1. Introduction

Networked Collaborative Virtual Environments (NCVEs), the systems that allow geographically distant users to interact in a shared virtual environment, have been an active research area for several years [Barrus96, Capin97, Carlsson93, Macedonia94, Ohya95, Pandzic97, Singh95]. For many applications, ranging from games to teleshopping or education, it is useful to include autonomous virtual actors in the virtual environment. These computer-controlled, seemingly autonomous creatures would inhabit the virtual worlds, make them more interesting, help users to find their way or perform a particular task. For example, a virtual shop might have an autonomous virtual shopkeeper to help the customer to find the needed wares; complex virtual environments might have guides to lead visitors and answer questions; in a game, an opponent or a referee might be an autonomous virtual actor.

Simulation of Autonomous Behavior (AB) is also a big research topic. There are different approaches and different implementations. Examples that influenced our work are [Zeltzer82] on task level animation, [Reynolds87] on behavioral group animation, [Blumberg95] on autonomous creatures for interactive virtual environments, [Badler93] and [Thalmann96] on autonomous humanoids and [Noser96] on behavioral L-systems. Systems that implement such behaviors are typically rather complex. As NCVE systems are already very complex themselves, our approach to Autonomous Behaviors in NCVEs is interfacing the two systems externally rather then trying to integrate them completely in a single, large system. Such an approach also facilitates the development of various AB systems to be used with a single NCVE system, making it a testbed for various algorithms.

This interfacing leads to a kind of symbiosis between an NCVE system and an AB system, where the AB system provides the brains and the NCVE system the body to move around, be seen and act upon objects, but also to see, hear and get the external information to the brain. In order to implement this strategy, the NCVE system must provide an external interface to which AB systems can be hooked. This interface should be as simple as possible to allow easy connection of various autonomous behaviors. At the same time it should satisfy the basic requirements for a successful symbiosis of a NCVE system and an AB system: allow the AB system to control its embodiment and act upon the environment, as well as gather information from the environment upon which to act.

The next section studies the requirements for the interface between NCVE and AB systems. After that we present an implementation of such an interface within the Virtual Life Network (VLNET) system [Capin97, Pandzic97]. For the results, we present two case studies. The first one is an implementation of dumb servant characters that respond to simple commands. The second one shows how a more complex AB system (*Eliza*, [Weizenbaum66]) was successfully connected to VLNET to simulate a conversational autonomous agent. Finally, we present conclusions and give directions for future work.

Our study is based on the assumption that the implemented autonomous actors are to be human-like. However, a lot of principles outlined here can be applied to non-human virtual beings.

2. Interfacing autonomous behaviors to virtual actors in NCVEs

In this section we study the symbiosis approach to the introduction of autonomous virtual actors in NCVE systems. In this approach, the NCVE system provides all functionalities of the body while the Autonomous Behavior (AB) system acts as the brain. As Figure 1 illustrates, the AB system is plugged into the NCVE system through an open interface. Keeping this interface simple allows easy connection of various AB algorithms to a single NCVE system, making the NCVE system a practical testbed for AB algorithms.

We study the functionalities that the NCVE system must provide to the AB system through the open interface. We have identified following important functionalities that must be provided for a successful symbiosis:

- embodiment
- locomotion
- capacity to act upon objects
- feedback from the environment
- verbal communication
- facial communication
- gestural communication

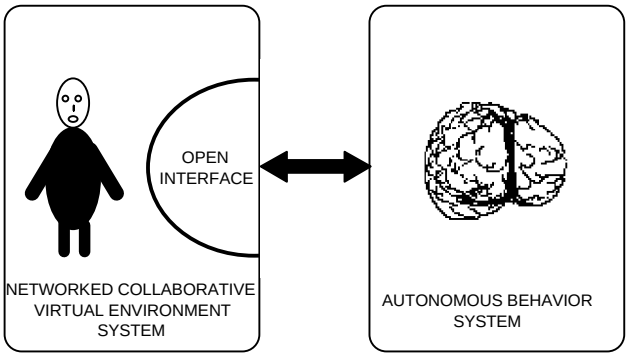

Figure 1: Symbiosis between the AB and NCVE systems

Embodiment, or a graphical representation, is a fundamental requirement to allow presence of the virtual actor in the environment. Though this can be a very simple graphical representation (e.g. a textured cube), some of the more advanced functionalities (e.g. facial and gestural communication) require a more human-like structure as support.

Locomotion is necessary for getting from one place to another, and might involve generation of walking motion or simple sliding around. Essentially, the AB system must be able to control the position of the embodiment in the environment.

Capacity to act upon objects in the environment is important because it allows the AB system to interact

with the environment. The interface should provide at least the possibility to grab and move objects.

Without some *feedback from the environment* our virtual actors might be autonomous, but blind and deaf. It is essential for them to be able to gather information from the environment about objects and other users.

If the AB system is to simulate virtual humans, it is necessary for real humans to be able to communicate with them through our most common communication channel - the *verbal communication*. Because the audio signal might in most cases be meaningless to the AB system, ASCII text seems to be the most convenient way to interface verbal communication to and from the AB system. This does not exclude the possibility of a speech recognition/synthesis interface on the human end, allowing us to actually talk to the virtual actors.

Finally, it is desirable for a virtual actor to have the capacity of *facial and gestural communication* and therefore the ability to have a more natural behavior by showing emotions on the face or passing messages through gestures.

In the following section we describe how these requirements are implemented in the VLNET system.

3. Autonomous behaviors in VLNET

Although Virtual Life Network (VLNET) [Capin97, Pandzic97] is developed as a general purpose NCVE system, we have paid particular attention to the support of autonomous virtual actors. In this section we show how the requirements outlined in the previous section are met by VLNET. In the following subsections we first give a brief introduction to VLNET, which allows us subsequently to discuss how each of the mentioned requirements is satisfied by VLNET interfaces.

3.1 Introduction to VLNET

VLNET is a general purpose client/server NCVE system using highly realistic virtual humans for user representation. VLNET achieves great versatility through its *open architecture* with a set of interfaces allowing external applications to control the system functionalities.

Figure 2 presents a simplified overview of the architecture of a VLNET client. The VLNET core performs all the basic system tasks: networking, rendering, visual data base management, user management including body movement and facial expressions. A set of simple shared memory interfaces is provided through which external applications can control VLNET. These interfaces are also used by the *VLNET drivers*. The drivers are small service applications provided as part of VLNET system that can be used to solve some standard tasks, e.g. generate walking motion, support navigation devices like mouse, SpaceBall, Flock of Birds etc. The connection of drivers and external applications to VLNET is established dynamically at runtime based on the VLNET command line.

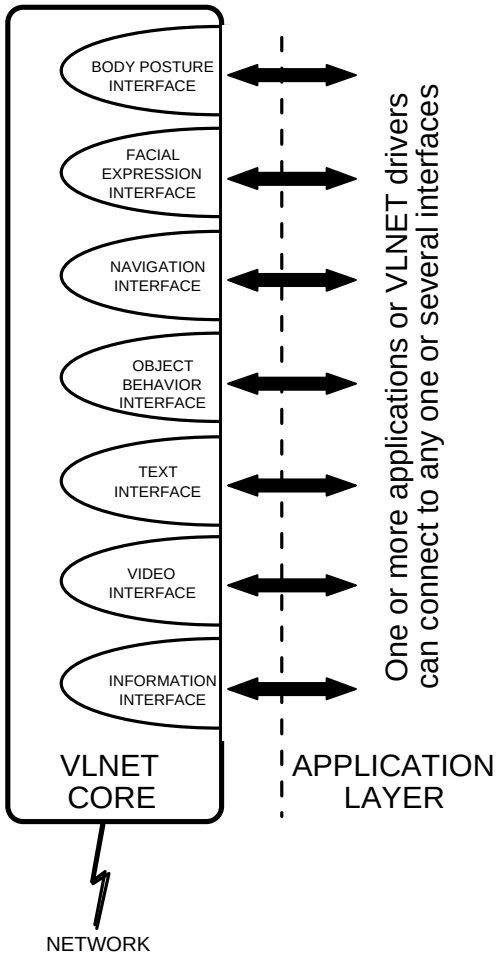

Figure 2: Simplified view of VLNET client architecture

The *Facial Expression Interface* is used to control expressions of the user's face. The expressions are defined using the Minimal Perceptible Actions (MPAs) [Kalra92, Kalra93]. The MPAs provide a complete set of basic facial actions, and using them it is possible to define any facial expression.

The *Body Posture Interface* controls the motion of the user's body. The postures are defined using a set of joint angles corresponding to 72 degrees of freedom of the skeleton model [Boulic90, Boulic95] used in VLNET.

The *Navigation Interface* is used for navigation, hand movement, head movement, basic object manipulation and basic system control. All movements are expressed using matrices. The basic manipulation includes picking objects up, carrying them and letting them go, as well as

grouping and ungrouping of objects. The system control provides access to some system functions that are usually accessed by keystrokes, e.g. changing drawing modes, toggling texturing, displaying statistics.

The *Object Behavior Interface* is used to control the behavior of objects. Currently it is limited to controlling motion and scaling defined by matrices passed to the interface. It is also used to handle the sound objects, i.e. objects that have prerecorded sounds attached to them. The Object Behavior Interface can be used to trigger these sounds.

The *Video Interface* is used to stream video texture (but possibly also static textures) onto any object in the environment. Alpha channel can be used for blending and achieving effects of mixing real and virtual objects/persons. The interface accepts requests containing the image (bitmap) and the ID of an object on which the image is to be mapped. The image is distributed and mapped on the requested object at all sites.

The *Text Interface* is used to send and receive text messages to and from other users. An inquiry can be made through the text interface to check if there are any messages, and the messages can be read. The interface gives the ID of the sender for each received message. A message sent through the text interface is passed to all other users in a VLNET session.

The *Information Interface* is used by external applications to gather information about the environment from VLNET. Because of its particular importance for implementation of autonomous actors we will present this interface in somewhat more detail. It provides high-level information while isolating the external application from the VLNET implementation details. It provides two ways of obtaining information, namely the request-and-reply mechanism and the event mechanism.

In the request-and-reply mechanism, a request is described and submitted to the VLNET system. Then, the request will be processed by the information interface engine in the VLNET system and a reply will be generated.

In the event mechanism, an event registration is described and submitted to the VLNET system. The event registration will be processed by the information interface engine and be stored in an event register. At each rendering frame, the VLNET system will process the event register and generate an event accordingly. There are two cases when an event will be generated. If it is the first rendering frame after the event is registered, then an event will definitely be generated. The second situation is when there is a change of information that is provided from the previous rendering frame. These event registrations will remain registered and be processed in each rendering frame until a removal from the event register is requested.

Figure 3 illustrates the information flow between the program controlling the autonomous actors and the VLNET system.

There are two message queues linking the program and the VLNET system. One message queue is used to submit requests, event registrations and event removals to the VLNET system. The other message queue is used to obtain replies and events from the VLNET system.

Within the VLNET system, the information interface engine is responsible for processing the requests, event registrations and event removals. The event registrations and event removals are sent to the event register, so that the event register can be updated accordingly. After processing the requests by the information interface engine, replies and events are generated and placed in the outgoing message queue from the VLNET system.

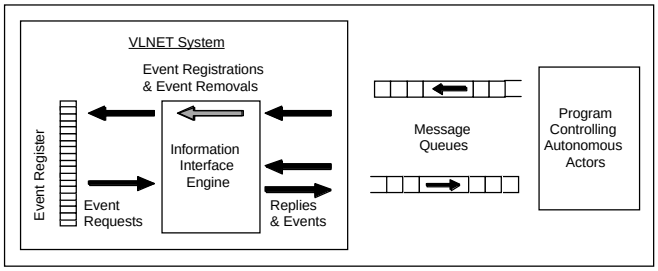

Figure 3: Data flow through the Information Interface

Following information can be requested from VLNET through the Information Interface:

- Description (name) of an object in the virtual world, given the object ID
- List of objects (object IDs and corresponding descriptions) whose description contains the given keywords
- List of users (user IDs and corresponding descriptions) whose description contains the given keywords
- Transformation matrix of an object in world coordinates, given the object ID
- Transformation matrix of a user in world coordinates, given the user ID
- Number of objects in the virtual world that is picked by an user, given the user ID
- Number of users in the virtual world
- Collisions between users/objects

We have given a briefest possible description of VLNET that allows us to show how it can be interfaced to an AB system. The focus of this presentation was on VLNET interfaces. For more details on VLNET the reader is directed to [Capin97, Pandzic97].

3.2 Support for Autonomous Behaviors in VLNET

In this subsection we show how all requirements analyzed in section 2 are satisfied in VLNET.

VLNET provides *embodiment* through the use of articulated, deformable body representations with articulated faces [Boulic95, Capin97, Pandzic97]. This highly realistic embodiment is also a good support for more advanced features like *facial and gestural communication.*

The VLNET interfaces explained in the previous subsection are the simple connection between VLNET and an AB system.

Locomotion capacity is provided through the navigation interface allowing the AB system to move its embodiment through passing of simple matrices. The same interface, through its possibility of picking up and displacing objects, provides the *capacity to act upon objects.* This capacity is further extended by the object behavior interface which allows to access any object in the environment or many objects simultaneously.

The information interface provides the AB system with the *feedback from the environment* with the possibility to request various types of data about the users and objects in the environment.

Facial communication is provided by the facial expression interface which allows the AB system to set any expression on the face of its embodiment.

Gestural communication is possible through the body posture interface, allowing the AB system to change postures of its embodiment and thus perform gestures.

Verbal communication is supported by the text interface. It allows the AB system to get text messages from other users and send text to them. On the other hand, it allows easy connection of speech recognition/synthesis module providing the human user with the possibility to speak with the virtual actor.

4. Case study 1: Dumb servant

This case study presents a simple implementation of "dumb servants", simple characters that react to commands issued by users. The dumb servant has the following characteristics:

- He/she communicates through natural language, understanding simple commands. Currently the commands have to be typed in but, as explained in subsection 3.2, connection of an external speech recognition module allowing to talk to the servant would be simple enough.
- He/she reacts to his/her name. If there is more then one servant, one can call them by their names. When called, the servant turns towards the caller.
- He/she can be ordered to come to the caller or go to different places that have names in the environment.
- He/she can be ordered to fetch objects and bring them to the caller.
- He/she is amiable. Whenever he/she executes an order, he/she smiles at the caller.

This simple implementation was done in a matter of hours with the purpose of showing how VLNET can be used to easily integrate simple behaviors. Based on the same architecture, the behavior of the servant could be vastly extended without any change to the interface with VLNET.

4.1 Implementation

The dumb servant is a program connected to VLNET through a set of interfaces (as described in subsection 3.1): information, text, navigation, object behavior and text interfaces. This connection is illustrated in Figure 4.

Text interface is used to receive messages. In view of the limited set of commands that the servant can receive, parsing is simple. The servant looks for his/her name in the text of the message and does not react if he/she does not find it. If the name matches, text interface is used to get the ID of the caller. With the ID, the position of the caller is obtained by passing a request to the information interface. This data already allows the servant to use the navigation interface to turn towards the caller or go to the caller if the command is to come.

The walking motion is generated by the standard VLNET walking driver connected to the body posture interface and to the navigation interface. This is a separate program that generates the appropriate leg movement based on the global motion of the body obtained from the navigation interface. Thus, the servant program only needs to move the body in the wanted direction and does not have to worry about generating realistic walking motion because this will be generated automatically.

When the servant is commanded to come to the caller he/she will always come just in front of the caller, facing him. The facial expression interface is then used to produce a smile.

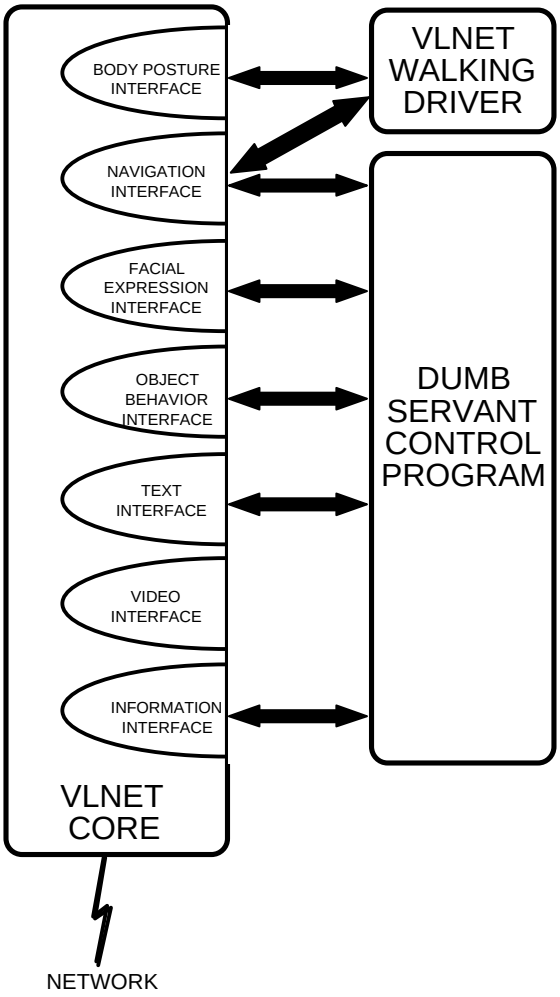

Figure 4: Connection of the servant application to VLNET interfaces

The objects in VLNET can have names, or short textual descriptions. As explained in subsection 3.1 through the information interface it is possible to get a list of objects in the environments with their names and IDs. Thus, from an object name the object ID can be obtained by looking into the list, then the ID is used to obtain the object's position. Using this, the servant can be ordered to go to the vicinity of a named object.

If ordered to fetch an object, the servant program uses the information interface to look up the object by its name, then gets its position. It moves to the position of the object, then uses the object behavior interface to move the object to the caller. As the object is brought, facial expression interface is used for the inevitable smile.

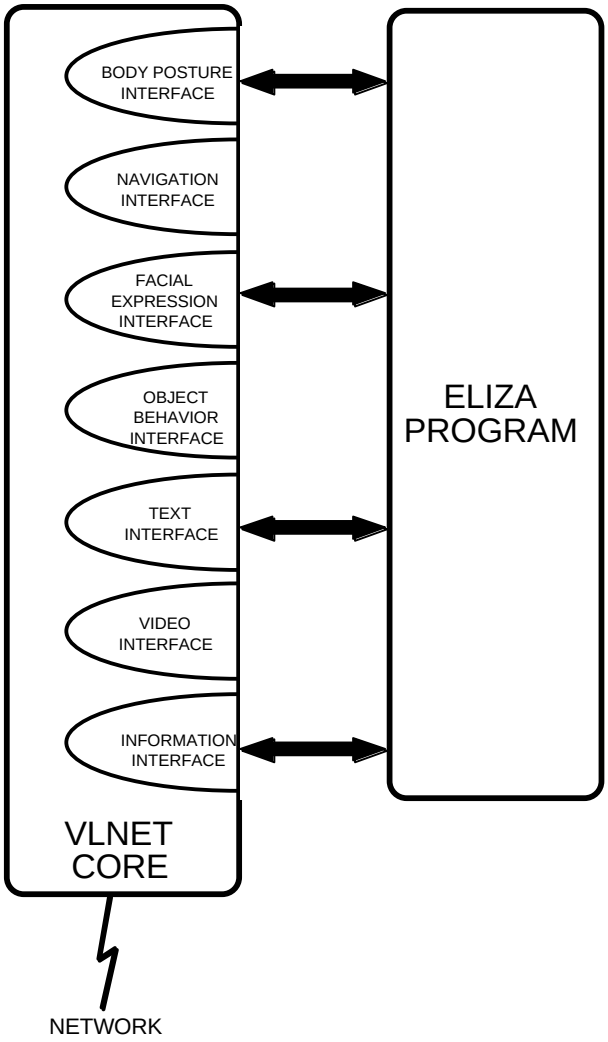

Figure 5: Connection of the conversational agent to VLNET interfaces

5. Case Study 2: Conversational embodied agent

This case study demonstrates the use of conversational embodied autonomous actors, that depend on textual communication with the participants. There is a difference between this implementation and the previous one: this autonomous virtual human is based on general conversation using natural language and body language, while the *dumb servant* understands simple commands such as picking objects and performs them.

The conversational virtual human has the following characteristics:

- It is based on a previously-developed freeware program, called Splotch. Splotch is a freeware C implementation of *Eliza* program, that is written by Duane Fields. Eliza is an autonomous agent program that studies natural language communication between real users and autonomous agents [Weizenbaum66] (various implementations of Eliza can be found at: (http://www-cgi.cs.cmu.edu/afs/cs.cmu.edu/project/ai-repository/ai/areas/classics/eliza/0.html). Although splotch is a simple version of Eliza, we have selected it due to its extendibility and ease of use, and because it was given in C source code form.
- He/she looks for participants around it, when there is at least one participant around him, he/she starts chatting with him.
- The conversation takes place textually through a chat window.
- He/she uses his/her nonverbal communication to strengthen and accompany his/her verbal communication.

5.1 Implementation

The conversational virtual human program is connected to VLNET through the textual, information, and body interfaces. Figure 5 illustrates the connection to the interfaces. The autonomous actor program obtains the participants' textual messages, and outputs answers, through the text interface.

We provided an initial implementation that is based on detecting the words used within the sentences. A set of key words are accompanied with gestures that the actor plays.For example, the autonomous actor uttering the word 'hmm' is assumed to be in paying attention to the participant's words, therefore the autonomous actor's virtual body is animated with a predefined keyframe sequence for attentive posture.

We have defined a pallette of gestures and postures that the autonomous actor can animate. Each keyframe sequence is accompanied with a set of keywords. Figure 6 illustrates our initial palette of gestures.

In the initial implementation, we only looked into which words the autonomous actor speaks Thus, at each text input from the remote participant, this input is passed to the Eliza program, the Eliza program outputs its answer, the answer text is searched for keywords, and one keyframe sequence is chosen using the found keywords. An extension would be to define the state of the actor in the autonomous behavior level, and select the gestures from this input, rather than analyzing the spoken words.

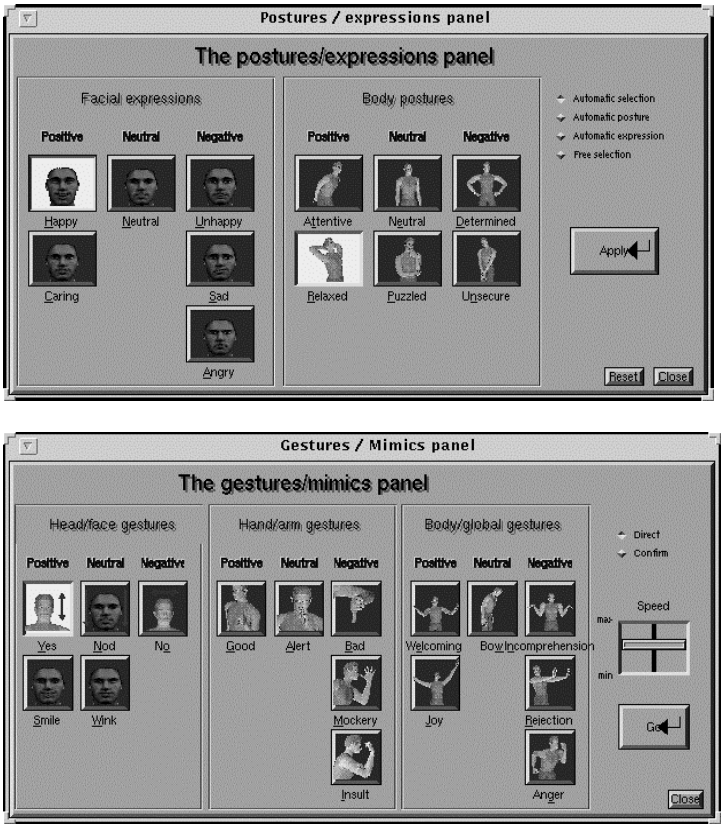

Figure 6: The palette of gestures

6. Conclusions and future work

We have discussed the challenges involved with inclusion of autonomous virtual actors in NCVEs and proposed a flexible and versatile solution within our Virtual Life Network system. We have presented two case studies where this system was successfully used to integrate autonomous actors in NCVE.

We intend to pursue the research in the direction of even more interesting autonomous actors and their interaction with human users.

7. Acknowledgments

This research is financed by "Le Programme Prioritaire en Telecommunications de Fonds National Suisse de la Recherche Scientifique" and the ESPRIT project VISTA.

Numerous colleagues at LIG and MIRALab have directly or indirectly helped this research by providing libraries, body and environment models, scenarios for applications.

8. References

[Badler93] Norman I. Badler, Cary B. Phillips, Bonnie Lynn Webber, Simulating Humans, Computer Graphics Animation and Control, Oxford University Press, 1993

[Barrus96] Barrus J. W., Waters R. C., Anderson D. B., "Locales and Beacons: Efficient and Precise Support For Large Multi-User Virtual Environments", Proceedings of IEEE VRAIS, 1996.

[Blumberg95] B.M. Blumberg, T.A. Galyean, Multi-Level Direction of Autonomous Creatures for Real-Time Virtual Environments, SIGGRAPH 95, Conference Proceedings, August 6-11, 1995, ACM Press, pp. 47-54.

[Boulic90] Boulic R., Magnenat-Thalmann N. M.,Thalmann D. "A Global Human Walking Model with Real Time Kinematic Personification", The Visual Computer, Vol.6(6),1990.

[Boulic95] Boulic R., Capin T., Huang Z., Kalra P., Lintermann B., Magnenat-Thalmann N., Moccozet L., Molet T., Pandzic I., Saar K., Schmitt A., Shen J., Thalmann D., "The Humanoid Environment for Interactive Animation of Multiple Deformable Human Characters", Proceedings of Eurographics '95, 1995.

[Capin97] Capin T.K., Pandzic I.S., Noser H., Magnenat Thalmann N., Thalmann D. "Virtual Human Representation and Communication in VLNET Networked Virtual Environments", IEEE Computer Graphics and Applications, Special Issue on Multimedia Highways, 1997.

[Carlsson93] Carlsson C., Hagsand O., "DIVE - a Multi-User Virtual Reality System", Proceedings of IEEE VRAIS '93, Seattle, Washington, 1993.

[Kalra92] Kalra P., Mangili A., Magnenat Thalmann N., Thalmann D., "Simulation of Facial Muscle Actions Based on Rational Free Form Deformations", Proc. Eurographics '92, pp.59-69., 1992.

[Kalra93] Kalra P. "An Interactive Multimodal Facial Animation System", PhD Thesis nr. 1183, EPFL, 1993

[Macedonia94] Macedonia M.R., Zyda M.J., Pratt D.R., Barham P.T., Zestwitz, "NPSNET: A Network Software Architecture for Large-Scale Virtual Environments", Presence: Teleoperators and Virtual Environments, Vol. 3, No. 4, 1994.

[Noser96] H. Noser, D. Thalmann, The Animation of Autonomous Actors Based on Production Rules, Proceedings Computer Animation'96, June 3-4, 1996, Geneva Switzerland, IEEE Computer Society Press, Los Alamitos, California, pp 47-57

[Ohya95] Ohya J., Kitamura Y., Kishino F., Terashima N., "Virtual Space Teleconferencing: Real-Time Reproduction of 3D Human Images", Journal of Visual Communication and Image Representation, Vol. 6, No. 1, pp. 1-25, 1995.

[Pandzic97] Pandzic I.S., Capin T.K., Lee E., Magnenat Thalmann N., Thalmann D., "A flexible architecture for Virtual Humans in Networked Collaborative Virtual Environments", Proceedings Eurographics 97 (to appear)

[Reynolds87] Reynolds C (1987), Flocks, Herds, and Schools: A Distributed Behavioral Model,Proc. SIGGRAPH 1987, Computer Graphics, Vol.21, No4, pp.25-34

[Singh95] Singh G., Serra L., Png W., Wong A., Ng H., "BrickNet: Sharing Object Behaviors on the Net", Proceedings of IEEE VRAIS '95, 1995.

[Thalmann96] D. Thalmann, H. Noser, Z. Huang, Chapter: How to Create a Virtual Life ?, in Interactive Computer Animation, eds. N.M. Thalmann, D. Thalmann, Prentice Hall Europe, 1996, pp 263 - 291

[Weizenbaum66].Weizenbaum, J., "ELIZA -- A computer program for the study of natural language communication between man and machine", Communications of the ACM 9(1):36-45, 1966.

[Zeltzer82] D. Zeltzer, Motor Control Techniques for Figure Animation, IEEE Computer Graphics and Applications, 2 (9), 53-59, 1982