Virtual Life Network: a Body-Centered Networked Virtual Environment*

Igor-Sunday Pandzic[1], Tolga K. Capin[2],
Nadia Magnenat Thalmann[1], Daniel Thalmann[2]

[1]MIRALAB-CUI, University of Geneva
CH1211 Geneva 4, Switzerland

[2]Computer Graphics Laboratory (LIG), Swiss Federal Institute of Technology (EPFL)
CH-1015 Lausanne, Switzerland

In order to feel the sense of presence in the Virtual Environment, it is important for the participants to become a part of this environment and interact with it through natural behaviors. This is even more emphasized in Networked Collaborative Virtual Environments where the participants need to see and interact with each other. We present the Virtual Life Network (VLNET), a joint research effort in the field of Networked Collaborative Virtual Environments at MIRALab - University of Geneva and Computer Graphics Laboratory of the Swiss Federal Institute of Technology - Lausanne. In VLNET each participant is represented by a virtual human actor with realistic appearance and movements similar to the actual body. Interacting with the environment through his virtual body, the participant is perceived by himself and others in a natural way. Since it is generally not possible to track all degrees of freedom of the human body in order to reproduce the realistic body motion, we introduce the motor functions that generate natural motion for standard tasks such as walking and arm motion; they are based on limited tracked information (hand and head positions). Using the same virtual human representation, but with the addition of high-level control, autonomous virtual actors can be introduced into the environment to perform some useful tasks or simply to make the environment more appealing. To further enhance the realistic feel of the virtual environment and to simplify object manipulation we provide the facility of defining object behaviors by attaching motor functions to the objects.

***Keywords**: Virtual Actors, Virtual Life, Computer Animation, Networked Virtual Environments, Multimedia*

1 Introduction

Increasing hardware and network performance together with advances in virtual reality technology have made networked virtual environments (VEs) a popular area of research. There has been an increasing number of efforts for building networked VEs in the past few years, and solutions have been proposed for building toolkits for communication in networked virtual worlds (Amselem, 1995; Carlsson & Hagsand, 1993; Macedonia et al., 1994; Singh et al., 1995), and special-purpose applications (Maxfield, Fernando & Dew, 1995; Stansfield et al., 1995; Gisi & Sacchi, 1994; Broll, 1995).

Networked virtual environments share problems with single-user environments, but some additional factors have to be considered. A virtual environment should give the users a feeling of *presence* within this environment providing better interaction and an intuitive interface. Presence requires that participants become part of the environment, and interact with the environment using natural means. The degree of presence is expected to increase with an increasing level of physical or social interaction with appropriate reactions from the environment. Therefore, a multi-user virtual environment system should provide efficient and accurate representation and interaction of the objects with realistic animation, as well as efficient communication management. A user feels a better degree of presence if other participants within the same environment believe that he is present and active in the same environment, and they show it. This property increases collaboration and interaction among participants within the VE. As the body movements help in showing intentions and real actions more clearly, hence decreasing ambiguity in interaction, it is important to represent the participants by virtual human bodies in shared environments.

There has been similar research to represent virtual humans in virtual environments (Granieri et al., 1995; Yoshida et al., 1995). In the VLNET (Virtual Life Network) system (Capin et al., 1995), we attempt to provide a more realistic representation through the use of motor functions, combined with interaction with the environment. The motor functions encompass more than inverse kinematics or displaying previously-recorded key frames, as they take into consideration other parameters specific to the motion.

*Published in Presence, MIT, Vol. 6, No 6, 1997, pp. 676-686

Typically, the VEs are created by bringing together different models, possibly with different scalings and even different formats. The lack of any corresponding interaction information concerning other objects makes it difficult to manipulate the scene, for example placing an object comfortably in the right location without it floating in air. Therefore, goal-oriented methods have to be provided for animating the objects depending on the user input. We propose different classes of motor functions that can be combined for this problem.

After giving a brief overview of the system structure and networking solutions, we present in more detail the important aspects of the system: the virtual human representation and animation, autonomous virtual actors and the issues of object manipulation and behaviors. Finally we present the results, experimental applications, conclusions and directions for future work.

2 System overview

From the networking point of view, VLNET is a based on a fairly simple client/server architecture. The server is mostly responsible for session management and message distribution. It is designed to work in pair with a standard HTTP server for database distribution. The design of the VLNET client is highly modular, with functionalities split into several processes. This allows not only performance improvements, but also the possibility to easily replace certain modules and obtain different functionalities. Next two sections discuss in more detail the networking issues and the client architecture.

2.1 The network structure

The communication in VLNET is based on a relatively simple client/server model. Figure 1 illustrates a VLNET server site with several clients connected to different worlds. A standard HTTP server and a VLNET Connection Server have to run permanently on a VLNET server site. They can serve several worlds, which can be either VLNET files or VRML 1.0 files. For each world, a World Server is spawned as necessary, i.e. when a client requests a connection to that particular world. The life of a World Server ends when all clients are disconnected.

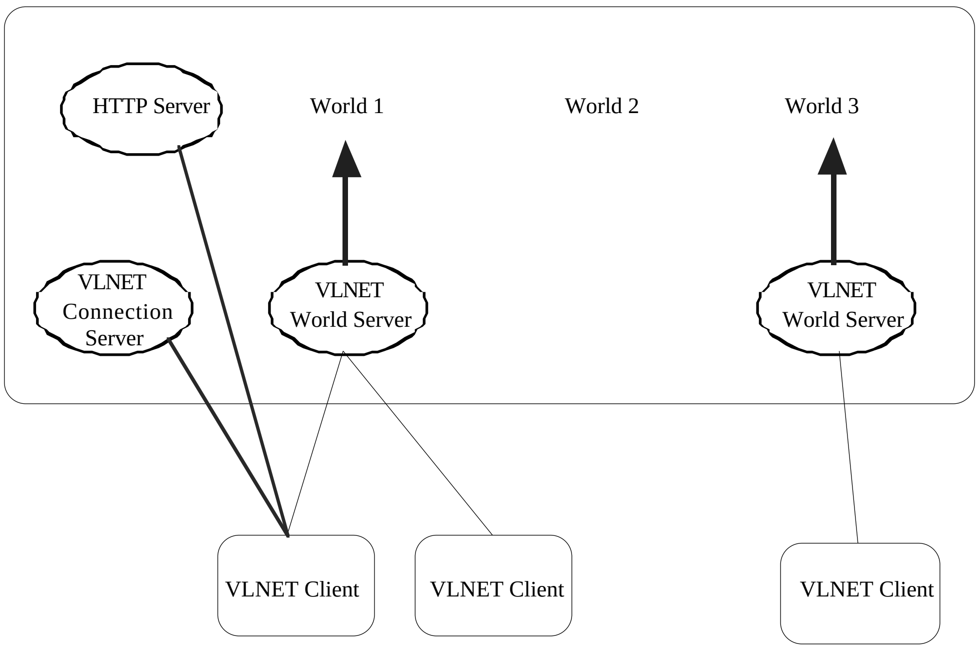
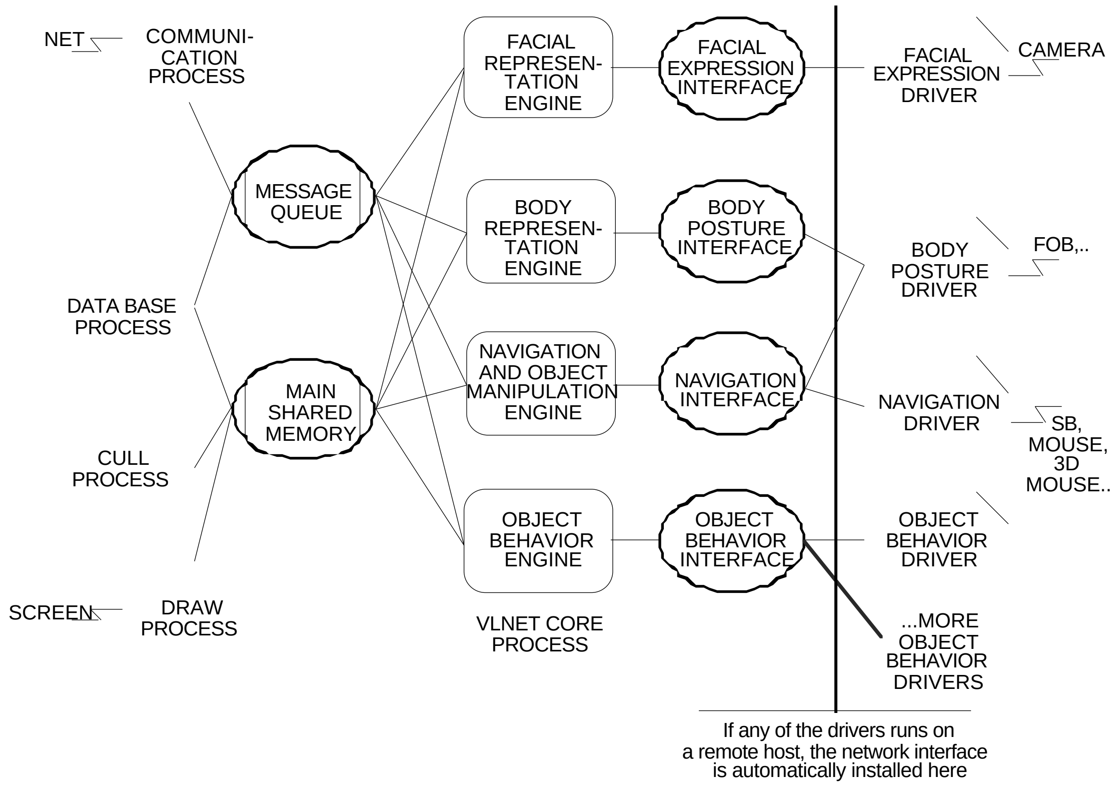


Fig. 1. VLNET server site with several connected clients

A VLNET session is initiated by a Client connecting to a particular world designated by a URL. The Client first fetches the world database from the HTTP server using the URL. After that it extracts the host name from the URL and connects to the VLNET Connection Server on the same host. The Connection Server gives the Client the port address of the World Server, and spawns the World Server for the requested world if one is not already running. The Client can provide the user data (the files describing the body the user wants to be represented with) by sending a URL. This data is distributed to other Clients in the session. The Client also has to fetch the user data from the other Clients. Once the connection is established, all communication between the clients in a particular world passes through the World Server.

2.2 VLNET Client architecture

The design of the VLNET Client is highly modular, with functionalities split into a number of processes. Figure 2 presents an overview of the modules and their connections. VLNET has an open architecture, with a set of interfaces allowing a user with some programming knowledge to access the system core and change or extend the system by plugging custom-made modules, called drivers, into the VLNET interfaces. These drivers only have to use a defined API to connect to VLNET. They can run on the local host, or on a remote host.



LEGEND:

Internal VLNET processes; can be changed only by recompiling VLNET

▢ Logical entities within VLNET core process, called engines

⬡ Internal shared memory segments for data exchange within internal processes; not accessable to users

⬡ External shared memory interfaces, accessable to the users through defined APIs

╲ External processes (called drivers); can be programed by the user using the defined APIs; they are replacable and sometimes optional

⎯∕⎺ External devices; sometimes optional or replacable

Fig. 2. Virtual Life Network Client overview

The VLNET core consists of four logical units, called engines, each with a particular task and an interface to external applications (drivers).

· The **Object Behavior Engine** takes care of the predefined object behaviors, like rotation or falling, and has an interface allowing to program different behaviors using external drivers.

· The **Navigation and Object Manipulation Engine** takes care of the basic user input: navigation, picking and displacement of objects. It provides an interface for the navigation driver. If no navigation driver is activated, standard mouse navigation exists internally. Currently, navigation drivers exist for the SpaceBall and Flock of Birds/Cyberglove combination. New drivers can easily be programmed for any device.

· The **Body Representation Engine** is responsible for the deformation of the body. In any given body posture (defined by a set of joint angles) this engine will provide a deformed body ready to be rendered. This engine provides the interface for changing the body posture. A standard Body Posture Driver is provided, that connects also to the navigation interface to get the navigation information, then uses the Walking Motor and the Arm Motor (Boulic et al., 1990; Pandzic et al., 1996) to generate the natural body movement based on the navigation. Another possibility is to replace this Body Posture Driver by a simpler one that is directly coupled to a set of Flock Of Birds sensors on the users body, providing direct posture control.

- The **Facial Representation Engine** provides the synthetic faces with a possibility to change expressions or the facial texture. The Facial Expression Interface is used for this task. It can be used to animate a set of parameters defining the facial expression.

All the engines in the VLNET core process are coupled to the main shared memory and to the message queue. Cull and Draw processes access the main shared memory and perform the functions of culling and drawing as their names suggest. These processes are standard SGI Performer (Rohlf & Helman, 1994) processes.

The Communication Process receives messages from the network (actually from the VLNET World Server) and puts them into the Message Queue. All the engines read from the queue and react to messages that concern them (e.g. Navigation Engine would react to a Move message, but ignore a Facial Expression message which would be handled by the Facial Representation Engine). All the Engines can write into the outgoing Message Queue, and the Communication Process will send out all the messages. All messages in VLNET use the standard message packet. The packet has a standard header determining the sender and the message type, and the message body. The message body content depends on the message type but is always of the same size (74 bytes), satisfying all message types in VLNET.

The data coming to any Engine through its external Interface is packed into a message packet and put into the Message Queue by the Engine. The Communication process sends out the packet, and the Communication Processes of other participants receive it and put it into the Message Queue. The appropriate Engine reads it from the Message Queue and processes it. In this way the data input from any Driver comes to the appropriate Engine at each participating site.

The Data Base Process takes care of the off-line loading of objects and user representations. It reacts to messages from the Message Queue demanding such operations.

3 The virtual humans

To improve the interaction among the participants in a multi-user virtual environment we employ the full-body participant representation using virtual actors. The body of a virtual actor realistically represents the real participant's body and the body animation is naturally correlated to the user actions. In addition to the user-guided actors, we have implemented virtual actors with self-determined actions: the autonomous actors. While having a high-level behavior control, the autonomous actors use the same representation and low-level motion control as the guided actors.

To enhance the communication facility between the users we include facial expressions on the virtual actors using facial animation techniques or by transmitting the real facial images captured by a camera and mapping them on the face of the virtual actor.

Each user can provide his/her personalized body for the representation in the Virtual Environment. The body can vary in shape, size, texture representing the clothes, facial features and texture. Tools that allow this kind of modeling are separate from the VLNET application.

3.1 Virtual body representation

The body representation in VLNET is based on the HUMANOID articulated body model (Boulic et al., 1995). The body can be represented in three levels of detail ranging from 2000 to 40000 triangular facets. For realistic modeling of human shapes, we make use of deformed body surfaces attached to the human skeleton (Shen & Thalmann, 1995; Boulic et al., 1995), rather than simple geometric primitives representing body links with a simple skeleton. This model allows the parametric representation of different human bodies. The human skeleton that we use is based on the anatomical structure of the real skeleton, while still allowing real-time control. It consists of 74 degrees of freedom without the hands, with an additional 30 degrees of freedom for each hand. The skeleton is represented by a 3D articulated hierarchy of joints, each with realistic maximum and minimum limits. A metaball structure is attached to the skeleton to simulate the muscle structure, and the final triangle mesh representing the skin is

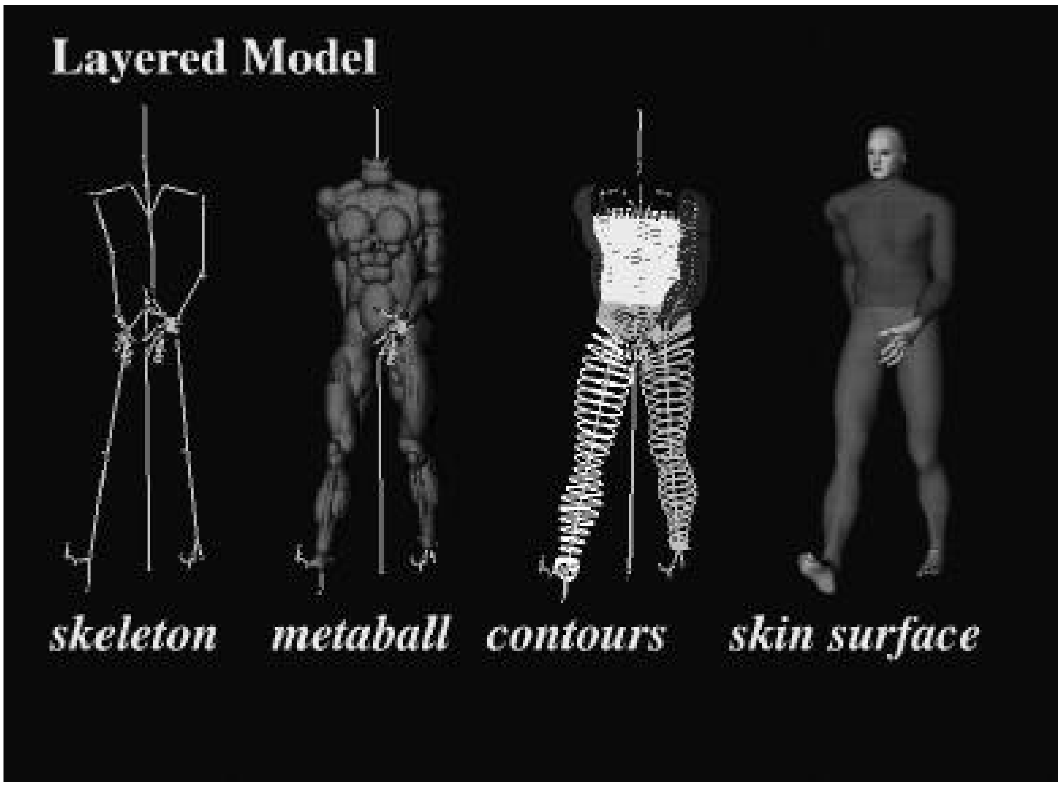


Fig. 3. The HUMANOID body model; skeleton structure, metaball structure, body contours, final body representation

calculated based on the positions of the metaballs when the skeleton moves (Shen & Thalmann, 1995). To ensure the calculation of the skin deformations in real time, most of the skin is precomputed in the neutral position of the skeleton, and only the parts susceptible to frequent and strong deformations (e.g. around the joints) are recalculated in each frame. This method can produce realistic skin deformation for an animated body in real time. Figure 3 illustrates the layers of the body model: skeleton, metaballs, body contours and skin surface.

3.2 Motion control for guiding virtual actors

The high quality visual representation is only one step towards a believable body model; at least as important is the natural body movement corresponding to the user actions. This could be best achieved by using a large number of sensors to track all degrees of freedom in the real body. However, this is generally not possible or not practical. Normally, only a few degrees of freedom will be tracked, and the rest has to be interpolated using the behavioral human animation knowledge and different motion generators.

Each user sees the virtual environment through the eyes of his body, and can control the movement of the body by various sensor devices (varying from spaceball and dataglove, to numerous sensors attached to body). In addition to his eye position, the user also has control of his virtual hand to interact with the environment (pick up and reposition objects). We selected these two modes of control, as most conventional input devices sense position and orientation of the head (e.g. head-mounted displays) and the hand (e.g. dataglove), so that a Navigation Driver (see figure 2) can easily be built for most devices.

In the VLNET system, we provide a set of motor functions that are responsible for different human motion: walking motor for *navigation*, and arm motor for *manipulation* of objects. These motor functions are more powerful than playing previously-recorded motions: they are based on approximations coming from biomechanical experiments, and they attempt to consider different parameters of the motion they are responsible for, in order to give parametrized motion (for example step length in walking as a function of velocity). They are actually implemented within the Body Posture Driver, reading the navigation data from the Navigation Interface and feeding the body posture data to the Body Representation Engine (see figure 2).

When the user navigates through the environment, the walking motor is used to perform a natural walking motion. The participant uses input devices (e.g. spaceball, dataglove with gesture interpretation) to update the eye position of the virtual actor. Based on this control, the incremental change of the eye position is computed and the rotation and velocity of the body center is estimated. The walking motor uses the instantaneous velocity to compute the length and duration of the walking cycle, from which it computes the joint angles of the body. The walking motor is based on the HUMANOID walking model (Boulic, Magnenat-Thalmann & Thalmann, 1990), guided interactively by the user or automatically generated from the given trajectory. Figure 4 shows an example of the walking motion in real time.

For object manipulation and the arm motion in general, the arm motor has to compute the joint angles of the arm based on the 6 degrees of freedom of the hand determined by user input. There are multiple solutions of joint angles reaching the same hand position, and the most realistic one has to be chosen. At the same time the joint constraints have to be taken into account. These considerations make the arm motor much more complicated then a simple inverse kinematics problem. For the arm motor we use the captured data obtained using sensors and stored into a precomputed table of arm joints. This table divides the normalized volume around the body into a discrete number of subvolumes (e.g. 4x4x4) and stores the mapping from subvolumes into joint angles of the right arm. Figure 4 shows an example of arm motion produced by this mechanism.

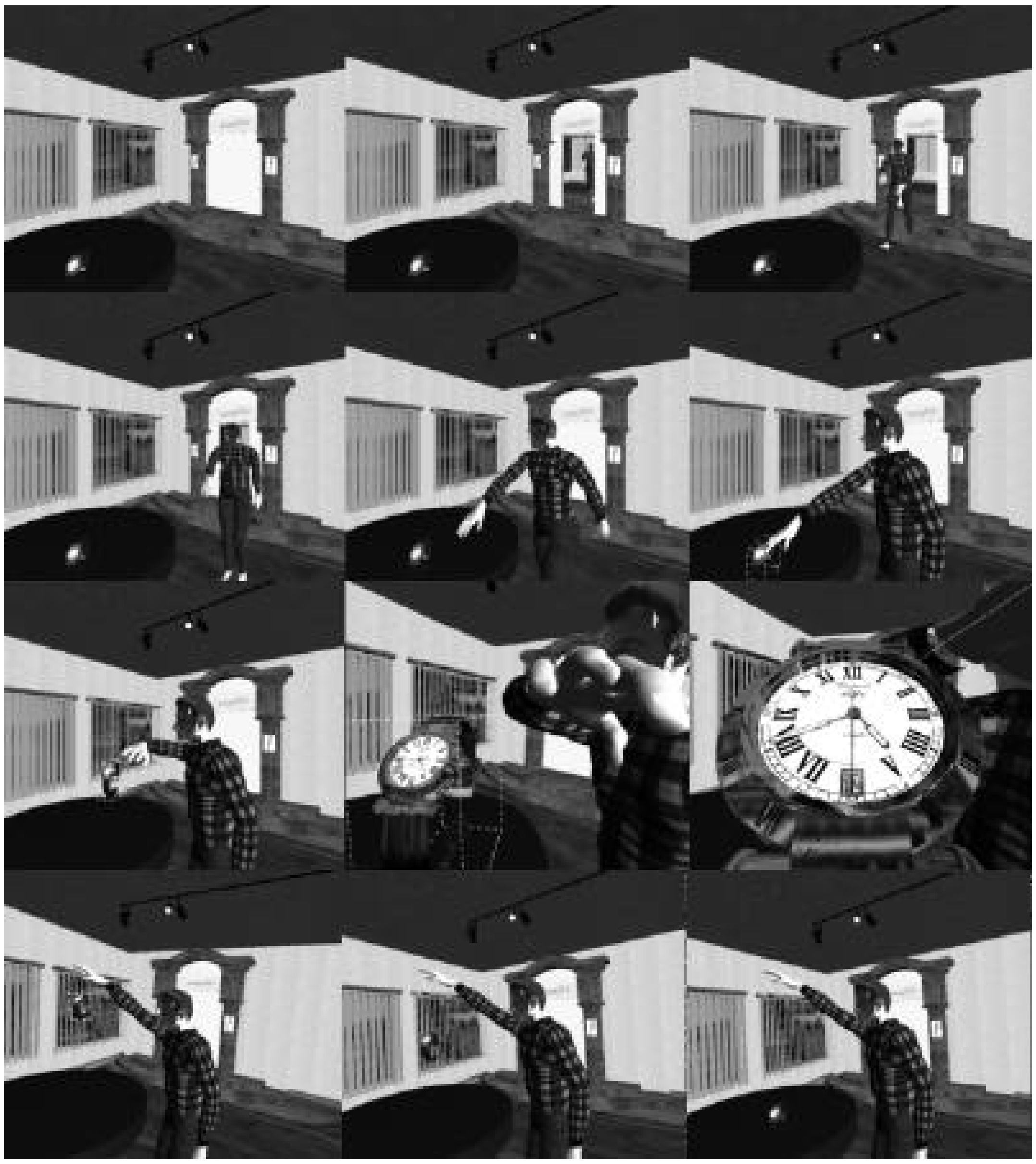


Fig. 4. The sequence of frames illustrating the walking and arm motors for the human body and different types of motor functions for objects: automatic door (user-dependent), hands of the watch showing time (time dependent), falling object (environment-dependent).

The users can also select a posture for the upper body for different emotions: tiredness, happiness, paying attention, etc. Currently, the user explicitly selects one emotion, using commands similar to *smileys* that are used commonly in text messages to express different emotions. The emotion motor sets the body joints at the vertebrae ending at the shoulders, based on this input. There is a need to define an emotion motor function that automatically recognizes the appropriate motion using data sensed from the real user. This current motor function is an introductory step to building an automatic emotion motor.

3.3 Facial expressions

Facial expressions are among the most important means of human communication, expressing intentions, thoughts and feelings. Therefore we include the facial communication in our multi-user virtual environment to enhance the communication between the users. There are two methods for facial communication: video texturing and model-based coding of facial expressions.

3.3.1 Video Texturing

In this approach we implement the facial communication by capturing the user's face using a camera and distributing it in real time to other users to be texture-mapped on the face of the virtual actor. Thus the virtual actor has the real moving face of the remote user.

The original images captured by the camera are first processed to extract the subset of the image containing the user's face. This processing is based on a comparison with an initial background image (the requirement is that the background is static). The extracted facial image is compressed at each frame and distributed to other users. This data does not pass through standard VLNET Communication Process which uses a fixed packet size, but through a special communication channel opened for this purpose. At the receiving side, an additional service process is charged with the receipt and decompression of the images. The main application gets the decompressed images through shared memory from the service process, decoupling the facial video frame rate from the application frame rate.

The facial images are texture-mapped on a simplified model of a human head with attenuated features. This is a compromise between mapping on a simple shape (e.g. box, ellipsoid) which would give unnatural results and mapping on a full-featured human head model where more precise image - feature alignment would be necessary. The texture mapping is illustrated in figure 5.

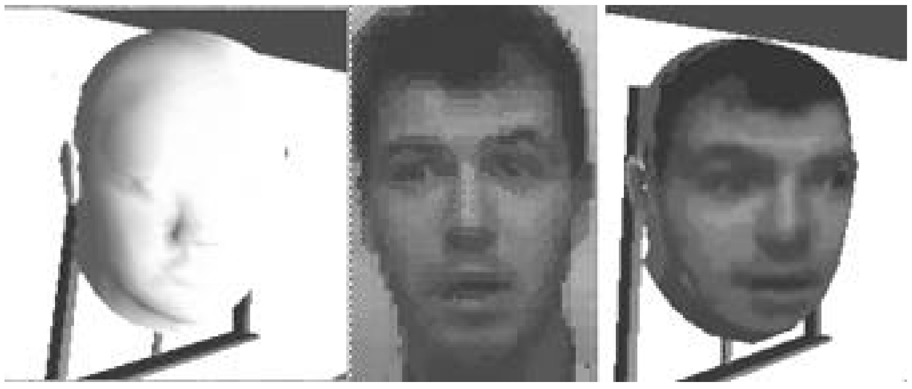


Fig. 5. Mapping of the face to the 3D virtual actor. Usage of simple head provides a compromise between 3D geometry and texture quality.

3.3.2 Model-based coding of facial expressions

Instead of transmitting whole facial images as in the previous approach, in this approach the images are analyzed and a set of parameters describing the facial expression is extracted (Pandzic et al., 1994). As in the previous approach, the user has to be in front of the camera that digitizes the video images of head-and-shoulders type. The set of extracted parameters includes global head motion, eyes aperture, gaze direction, eyebrow positioning, jaw aperture, mouth shape. These parameters are packed into standard VLNET message packets and transmitted.

On the receiving end, the Facial Representation Engine receives messages containing facial expressions and performs the facial animation accordingly. This method can be used in combination with texture mapping. The model needs an initial image of the face together with a set of parameters describing the

position of the facial features within the texture image in order to fit the texture to the face. Once this is done, the texture is fixed with respect to the face and does not change, but it is deformed together with the face, in contrast with the previous approach where the face was static and the texture was changing.
The main drawback of the facial module is that it is not possible to incorporate it with users wearing HMD, as the face cannot be captured. However, it provides a good medium of interaction with shutter glasses or in the absence of these devices.

3.4 Autonomous virtual actors

Autonomous virtual actors can be designed to populate the environment and perform some useful tasks, like guiding the participants, playing some game with them, or simply making the environment more interesting and appealing. The autonomous actors are connected to the system in the same way as human participants using the VLNET core, but the user guidance modules are replaced by autonomous behavior modules. As these virtual actors are not guided by the users, they should have sufficient autonomous behaviors to act autonomously to accomplish their tasks. This requires building behaviors for motion, as well as appropriate mechanisms for interaction.

Given the open architecture of VLNET, the system is suitable as a testbed for all kinds of autonomous virtual humans in the Networked Virtual Environment. The external drivers (figure 2) are simply replaced by the autonomous behavior module, replacing the user input by an autonomous decision mechanism.
Animation of autonomous actors is an active area of research (Thalmann, 1994; Phillips & Badler, 1991). A typical behavioral animation system consists of three key components, which are connected to each other:

· the perceptual system, which senses the environment, and provides the perception information about the objects with which the actor can interact (Renault, Magnenat-Thalmann & Thalmann, 1990),
· the organism system, which is concerned with the rules, skills, motives, drives and memory,
· the locomotor system, which is responsible for using motors to generate motion.

The perceptual system should be realistic by providing the limitations of sensing the surrounding, and should be improved through the synthetic vision (Renault, Magnenat-Thalmann & Thalmann, 1990). It should represent what the actor would see in the real world (For example, the actor should not be able to see through walls). The organism system should consider the different goals with the inference mechanism in order to achieve motions. For the locomotor system, motors similar to those for guided actors (e.g. for walking), can be used.

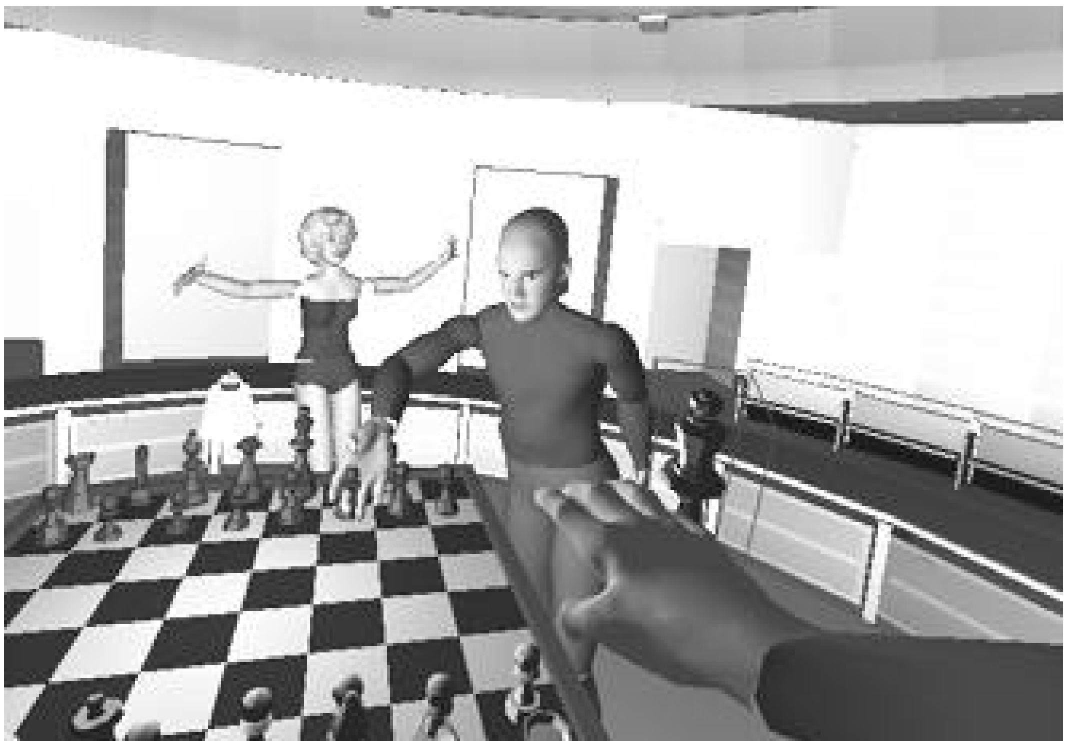


Fig. 6. An example shared environment with two guided actors and one autonomous actor

For initial implementation, we have constructed a test in which the autonomous actor, using simple behaviors, interacts with the participant. In this experiment, virtual Marilyn as shown in Figure 6 tries to attract the participant's attention by coming near him and opening her arms. If the user walks away from her, she follows the participant. If the user comes too close to her, she escapes from him and goes to a distant corner of the environment. She waits there until the user comes near her, and begins following him again. This test, although very simple, provided good interaction with 11 subjects we tried. The subjects in general claimed that the walking behavior and body gestures for the autonomous actor were important factors in their interaction. The experiment had very simple characteristics: the autonomous actor does not have a realistic vision subsystem, has very simple behaviors, and she does not recognize the gestures of the user. However, this simple experiment showed that autonomous actors are a good tool for increasing immersion in virtual environments.

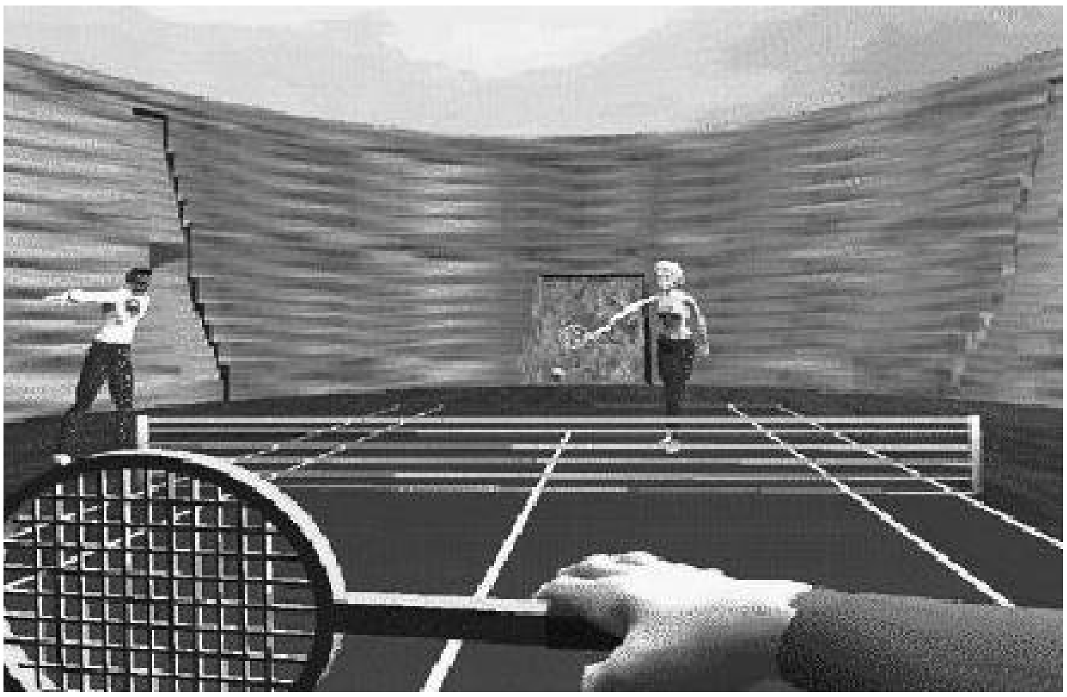


Fig. 7. Playing tennis against a virtual opponent with a virtual referee

Another experiment is illustrated in figure 7, where the user can play tennis against a computer-guided opponent, with the computer-guided referee (Noser et al., 1996.). Both the player and the referee have vision-based autonomous behaviors.

4 Interaction with Virtual Environment and Object Behaviors

It is expected that the participants feel a higher degree of presence if the environment *reacts* to their actions in a realistic way. For example, the user should be able to interact with the environment, reposition objects by picking them up with his virtual hand, and releasing them, making them fall. In order to pick up an object, the user moves his hand near the object and explicitly requests picking (e.g. by clicking spaceball button, closing dataglove). The objects stay picked until released explicitly by the user.

Typically the VEs are created by bringing together different models, possibly with different scalings and even different formats. Unlike CAD models, these models lack any corresponding interaction information between objects. This makes it difficult to manipulate the scene. A dynamic simulation with collision response would solve this problem. However for medium-sized environments this is a time-consuming solution, resulting in unwanted delays in the simulation. Therefore, we adopt a solution which compromises between realistic appearance and goal-oriented behaviors. We propose three classes of motor functions that can be attached to the objects.

A set of behaviors can be associated dynamically with any object in the environment. The object behaviors are implemented as different motor functions which give them a means of interacting with the users and the other objects. The types of motor functions can be divided into 3 classes:

- *continuous motor functions:* these functions require transformation update of the object regularly, within a specific period of time without any delay. For example, hands of a clock to show the time are in this category.
- *user-dependent motor functions:* these functions depend on the user input. This can be an explicit user input (for example, request for changing servers, see below); or implicit input (for example, automatic door behavior driven by position of the user).
- *environment-dependent motor functions:* these functions are dependent on the environment as well as the object itself. We define different built-in motor functions corresponding to this category: magnet, vertical displacement, horizontal displacement, axis alignment. Magnet allows to attach different objects to each other with a predetermined transformation matrix (e.g. the watch body and bracelet are always attached with one transformation). Vertical displacement is called when the object is released; and is used for making the objects fall until it collides with an object, simulating gravity.

A subset of these behaviors can be added optionally to the objects during the scene creation. A new motor function can be added only by programming. However, external Object Behavior Drivers can be attached to the system to provide more object behaviors. Figure 4 demonstrates some examples of motor functions attached to objects.

5 Results

We have tested the VLNET system over the ATM network between Geneva and Singapore, provided during the Telecom'95 exhibition in Geneva. Our results showed that the ATM network is suitable for guaranteeing quality of service for small-sized packets between the server and the clients. Other experiments were undertaken between multiple users located in Switzerland and Japan over the Internet network and Swiss ATM Pilot network, as well as with sites in Belgium and Great Britain.

We have built experimental worlds for a number of applications:

- Teleshopping: The VLNET system has been experimentally used by *Chopard Watches, Inc.* , Geneva, to collaboratively present and view the computer-generated models of the recently designed watches with remote customers and colleagues in Singapore and Geneva. The models were developed using AutoDesk software, and were easily included in the virtual environment.
- Business: Experiments are continuing for building a virtual meeting room for distant users with utilities like images, slide shows, movies as well as 3D objects.
- Entertainment: The VLNET environment is also used by remote partners for playing chess and puzzles. These models were created using the IRIS Inventor^{TM} and WaveFront^{TM} packages. Another gaming application is playing tennis against a computer-guided opponent.
- Interior design: Currently, experiments are continuing on furniture design by the customer and the sales representative to build a virtual house. The model was created using the WaveFront package.
- Medicine: By importing medical images or 3D representation of organs reconstructed from medical images (Kalra et al., 1995.) the VLNET system can be used for education or consultations in medicine.

Figure 8 shows snapshots of the system in use for some of the mentioned applications.

6 Conclusion and future work

We have presented the Virtual Life Network, a Networked Collaborative Virtual Environment system using the virtual humans. The open architecture of VLNET makes it a flexible framework for various kinds of collaborative applications, possibly including autonomous virtual humans.

Future work might concentrate on the aspects of facial and gestural communication, object manipulation and behaviors, and algorithms for guidance of autonomous virtual humans.

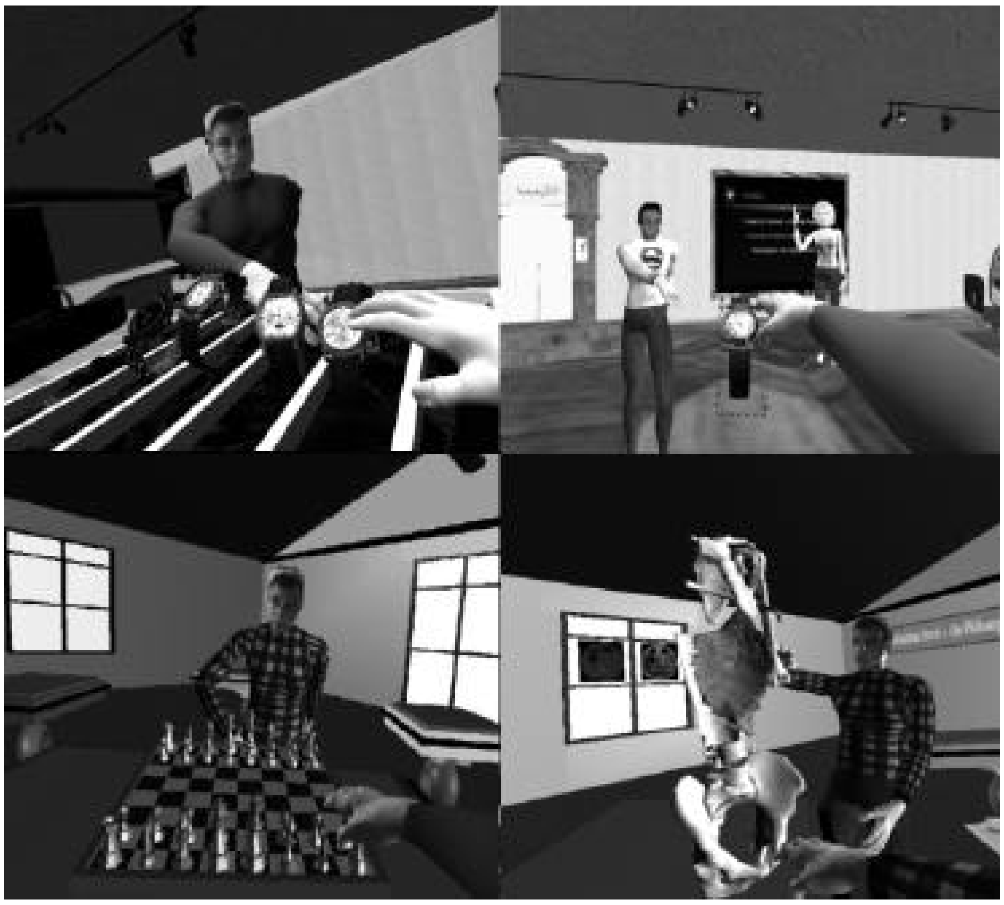


Fig. 8. Some application examples (teleshopping, business, entertainment, medical)

Acknowledgments

The research was partly supported by ESPRIT project HUMANOID (P 6079), Swiss National Foundation for Scientific Research, Silicon Graphics, Federal Office of Education and Science, the Department of Economy of City of Geneva and the VISINET project. We would like to thank assistants of LIG and MIRALAB for the models and libraries.

References

Amselem D. (1995). A Window on Shared Virtual Environments. Presence: Teleoperators and Virtual Environments, Vol. 4, No. 2.

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. (1995). The Humanoid Environment for Interactive Animation of Multiple Deformable Human Characters", *Proceedings of Eurographics '95*. Maastricht.

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

Broll W. (1995). Interacting in Distributed Collaborative Virtual Environments", *Proceedings of IEEE VRAIS'95*.

Capin T.K., Pandzic I.S., Magnenat-Thalmann N., Thalmann, D. (1995). Virtual Humans for Representing Participants in Immersive Virtual Environments. *Proceedings of FIVE '95*, London.

Carlsson C., Hagsand O. (1993). DIVE - a Multi-User Virtual Reality System. *Proceedings of IEEE VRAIS '93,* Seattle, Washington.

Gisi M.A., Sacchi C. (1994). Co-CAD: A Collaborative Mechanical CAD System. *Presence: Teleoperators and Virtual Environments*, Vol. 3, No. 4.

Granieri J.P., Becket W., Reich B.D., Crabtree J., Badler N.I. (1995). Behavioral Control for Real-Time Simulated Human Agents. *Proceedings of ACM Symposium on Interactive 3D Graphics*, Monterey, California.

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

Maxfield J., Fernando T., Dew P. (1995). A Distributed Virtual Environment for Concurrent Engineering. *Proceedings of IEEE VRAIS '95*.

Noser, H., Pandzic, I.S., Capin, T.K., Magnenat Thalmann, N., Thalmann, D. (1996). Playing Games through the Virtual Life Network, *Proceedings of Artificial Life V*, Nara, Japan.

Pandzic I.S., Kalra P., Magnenat-Thalmann N., Thalmann D. (1994). Real-Time Facial Interaction. *Displays*, Vol 15, No 3.

Pandzic, I.S., Capin, T.K., Magnenat Thalmann, N., Thalmann, D.(1996) Motor functions in the VLNET Body-Centered Networked Virtual Environment. *Proc. of 3rd Eurographics Workshop on Virtual Environments*, Monte Carlo

Phillips, C.B., Badler, N.I. (1991). Interactive Behaviors for bipedal articulated figures. *Computer Graphics*. Vol. 25, no. 5.

Rohlf J., Helman J. (1994) IRIS Performer: A High Performance Multiprocessing Toolkit for Real-Time 3D Graphics, *Proc. SIGGRAPH'94*.

Shen, J., Thalmann, D. (1995). Interactive Shape Design Using Metaballs and Splines. *Eurographics Workshop on Implicit Surfaces*. Grenoble.

Renault, O., Magnenat-Thalmann, N., Thalmann, D. (1990). A Vision-based Approach to Behavioral Animation. *The Journal of Visualization and Computer Animation*, Vol.1, No.1

Singh G., Serra L., Png W., Wong A., Ng H. (1995). BrickNet: Sharing Object Behaviors on the Net. *Proceedings of IEEE VRAIS '95*.

Stansfield S., Miner N., Shawver D., Rogers D. (1995). An Application of Shared Virtual Reality in Situational Training. *Proceedings of IEEE VRAIS '95*.

Thalmann, D. (1994). Automatic Control and Behavior of Virtual Actors. *Interacting with Virtual Environments*, MacDonald L., Vince J. (Ed)

Yoshida M., Tijerino Y., Abe S., Kishino F. (1995). A Virtual Space Teleconferencing System that Supports Intuitive Interaction for Creative and Cooperative Work. *Proceedings of ACM Symposium on Interactive 3D Graphics*. Monterey, California.