A flexible architecture for Virtual Humans in Networked Collaborative Virtual Environments

Igor Pandzic[1], Tolga 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

Although a lot of research has been going on in the field of Networked Collaborative Virtual Environments, most of the existing systems still use fairly simple embodiments for the representation of participants in the environments. This kind of systems can greatly benefit from a more sophisticated human representation. The users' more natural perception of each other (and of autonomous actors) increases their sense of being together, and thus the overall sense of presence in the environment. The communication facilities are also improved because a realistic embodiment allows for gestural and facial communication.

We present a flexible framework for the integration of virtual humans in the Networked Collaborative Virtual Environments. It is based on a modular architecture that allows flexible representation and control of the virtual humans, whether they are controlled by a physical user using all sorts of tracking and other devices, or by an intelligent control program turning them into autonomous actors. The modularity of the system allows for fairly easy extensions and integration with new techniques making it interesting also as a testbed for various domains from "classic" VR to psychological experiments. We present results in terms of functionalities, applications tried and measurements of performance and network traffic with an increasing number of participants in the simulation.

1. Introduction

The Networked Collaborative Virtual Environments (NCVE) are often described as systems that permit to the users to feel as if they were together in a shared Virtual Environment. Indeed, the feeling of "being together" is extremely important for collaboration, as well as for the sense of presence felt by the subjects. A very important factor for the feeling of being together in a virtual world is the way users perceive each other: their embodiment. In a broader sense, this includes not just the graphical appearance but also the way movements, actions and emotions are represented.

Although Networked Collaborative Virtual Environments have been around as a topic of research for quite some time, in most of the existing systems the embodiments are fairly simple, ranging from primitive cube-like appearances, non-articulated human-like or cartoon-like avatars to articulated body representations using rigid body segments [Barrus 96, Carlsson 93, Macedonia 94, Singh 95]. Ohya et al. [Ohya 95] report the use of human representations with animated bodies and faces in a virtual teleconferencing application. In the approach we adopted when developing the Virtual Life Network (VLNET) system [Capin 94, Thalmann 95, Pandzic 96], more sophisticated virtual humans are simulated, including full anatomically based body articulation, skin deformations and facial animation (figure 1). Managing multiple instances of such complex representations and the involved data flow in the context of a Networked Collaborative Virtual Environment, while maintaining versatile input and interaction options, requires careful consideration of the software architecture to use. In view of the complexity of the task and versatility we wanted to achieve, a highly modular, multiprocess architecture was a logical choice. Some of the modules are fixed as part of the system core, while the others are replaceable external processes allowing greater flexibility in controlling the events in the environment, in particular the movement and facial expressions of the virtual humans.

We first introduce in general the problems linked with involving Virtual Humans in NCVEs. Next section contains the presentation of overall properties of the VLNET system, followed by a detailed description of the software architecture adopted for the system: VLNET server, then VLNET client with its processes, engines, interfaces and drivers. The results section covers some experimental applications of VLNET, as well as performance and network measurements with an increasing number of users. Finally we present our conclusions and discuss possibilities for future work.

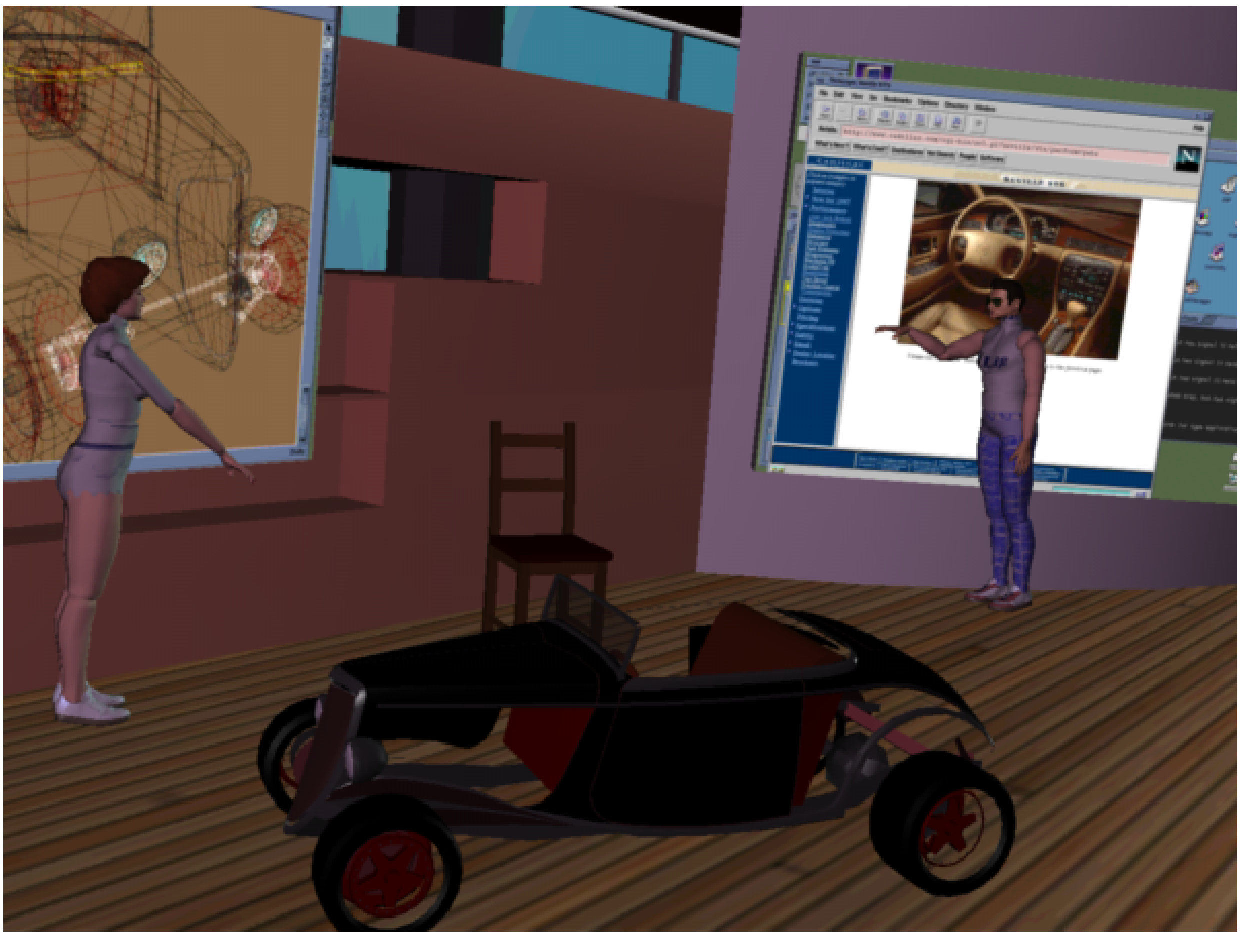


Figure 1: An example session of the Virtual Life Network

2. Introducing Virtual Humans in NCVEs

The participant representation in a networked VE system has several functions:

- perception (to see if anyone is around)
- localization (to see where the person is)
- identification (to recognize the person)
- visualization of interest focus (to see where the person's attention is directed)
- visualization of actions (to see what the person is doing)
- communication (lip movement in sync with speech, facial expressions, gestures)

Virtual Humans can fulfill all these functions in an intuitive, natural way resembling the way we achieve these tasks in real life. Even with limited sensor information, a virtual human frame can be constructed in the virtual world, reflecting the activities of the real user. Slater and Usoh [Slater94] indicate that such a body, even if crude, already increases the sense of presence that the participants feel. The participants visualize the environment through the eyes of their virtual actor, and move their virtual body by different means of body control. In addition, introducing the human-like autonomous actors for various tasks increases the level of interaction within the virtual environment.

Introducing virtual humans in the distributed virtual environment is a complex task combining several fields of expertise [Capin 97]. The principal components are:

- virtual human simulation, involving real time animation/deformation of bodies and faces
- virtual environment simulation, involving visual data base management and rendering techniques with real time optimizations
- networking, involving communication of various types of data with varying requirements in terms of bitrate, error resilience and latency
- interaction, involving support of different devices and paradigms
- artificial intelligence (in case autonomous virtual humans are involved), involving decision making processes and autonomous behaviors

Each of the involved components represents in itself an area of research and most of them are very complex. When combining them together the interaction between components and their impact on each other have to be considered. For example, using virtual humans sets new requirements on interaction which has to allow not only simple interaction with the environment, but at the same time the visualization of the actions through the body and face representing the user; the necessity to communicate data through the network forces more compact representation of face and body animation data.

Considering the total complexity of the above components, a divide-and-conquer approach is a logical choice. By splitting the complex task into modules (see figure 3), each with a precise function and with well defined interfaces between them, several advantages are achieved:

- high flexibility
- easier software management, especially in a team work environment
- higher performance
- leveraging the power of multiprocessor hardware or distributed environments when available

Flexibility is particularly important, because of the multitude of emerging hardware and software technologies that can potentially be linked with NCVE systems (various input devices and techniques, AI algorithms, real-time data sources driving multi-user applications). This is especially interesting in a research environment where a NCVE system can be used as a testbed for research in fields of AI, psychology, medical information systems etc. In general, a good NCVE system must allow implementation of different applications while transparently performing its basic tasks (networking, user representation, interaction, rendering...) and letting the application programmer concentrate on the application-specific problems.

From the software management point of view, a monolithic system of this complexity would be extremely difficult to manage, in particular by a team of programmers.

By carefully assigning tasks to processes and synchronizing them intelligently, higher performance can be achieved [Rohlf 94].

Finally, a multi-process system will naturally harness the power of multi-processor hardware if it is available. It is also possible to distribute modules on several hosts.

Once it is decided to split the system into modules, roughly corresponding to the above listed components, it is necessary to define in detail the task of each module and the means of communication between them. Next section describes in detail how it is done in the case of the Virtual Life Network system.

3. Virtual Life Network

Based on the considerations from the previous section we have developed the Virtual Life Network (VLNET) system. From the networking point of view, VLNET is based on a fairly simple client/server architecture. Next two subsections discuss in more detail the server and the client architecture.

3.1. VLNET Server

A VLNET server site consists of a HTTP server and a VLNET Connection Server. 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.

Figure 2 schematically depicts a 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 spawns the World Server for the requested world if one is not already running and sends to the Client the port address of the World Server. Once the connection is established, all communication between the clients in a particular world passes through the World Server.

In order to reduce the total network load, the World Server performs the filtering of messages by checking the users' viewing frusta in the virtual world and distributing messages only on as-needed basis. Clients keep the possibility of contourning this mechanism by requesting a higher delivery insurance level for a particular message, e.g. for heartbeat messages of a dead reckoning algorithm [Capin 97-1].

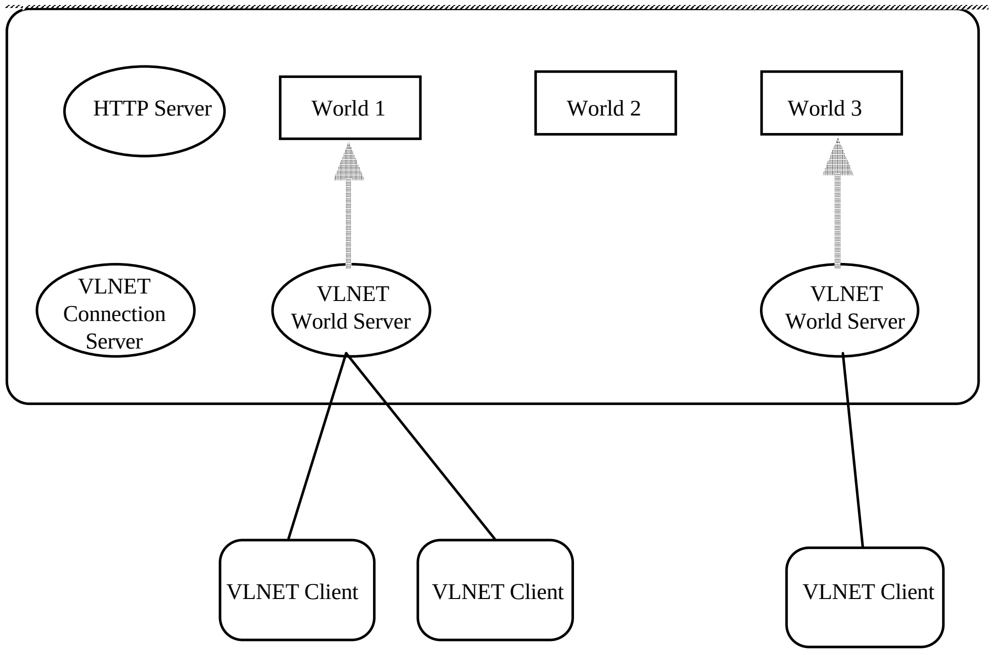


Figure 2: Connection of several clients to a VLNET server site

3.2. VLNET Client

The design of the VLNET Client is highly modular, with functionalities split into a number of processes. Figure 3 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. In the next subsections we explain in some detail the VLNET Core with its various processes, as well as the drivers and the possibilities for system extension they offer.

3.2.1. VLNET Core

The VLNET core is a set of processes, interconnected through shared memory, that perform basic VLNET functions. The Main Process performs higher level tasks, like object manipulation, navigation, body representation, while the other processes provide services for networking (Communication Process), database loading and maintenance (Database Process) and rendering (Cull Process and Draw Process).

3.2.1.1. The Main Process

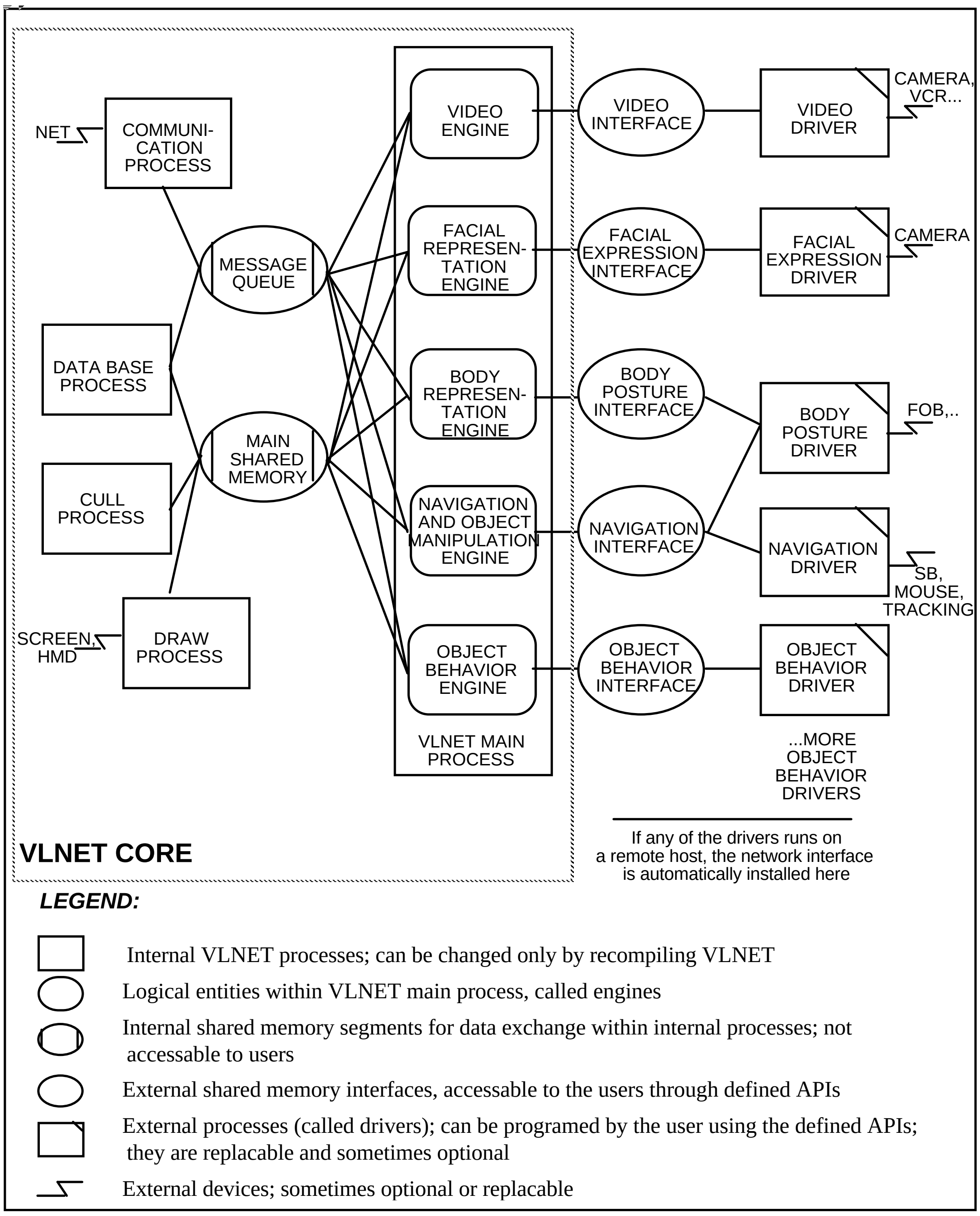
The Main Process consists of five logical entities, called engines, covering different aspects of VLNET. It also initializes the session and spawns all other processes and drivers. Each engine is equipped with an interface for the connection of external drivers.

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.

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. The body representation is based on the Humanoid body model [Boulic 95]. 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 90, Pandzic 96] to generate the natural body movement based on the navigation.

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. The facial representation is a polygon mesh model with Free Form Deformations simulating muscle actions [Kalra 92].



LEGEND:

□	Internal VLNET processes; can be changed only by recompiling VLNET
◯	Logical entities within VLNET main 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

Figure 3: Virtual Life Network system overview

The **Video Engine** manages the streaming of dynamic textures to the objects in the environment and their correct mapping. Its interface provides the user the possibility to stream video textures on any object(s) in the environment. The facial texture mentioned concerning the Facial Representation Engines is a special case and it is handled by both engines.

All the engines in the VLNET core process are coupled to the main shared memory and to the message queue. They use the data from the culling process in order to perform the "computation culling". This means that operations are performed only for user embodiments and other objects when they are within the field of view, e.g. there is no need to perform facial expressions if the face is not visible at the moment. Substantial speedup is achieved using this technique.

3.2.1.2. Cull and Draw Processes

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 94] processes.

3.2.1.3. The Communication Process

The Communication Process handles all network connections and communication in VLNET. All other processes and engines are connected with it through the Message Queue. They fill the queue with outgoing messages for the Communication Process to send. The messages are sent to the server, distributed and received by Communication Processes of other clients. The Communication Process puts these incoming messages into the Message Queue from where the other processes and engines can read them and react. 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 (80 bytes), satisfying all message types in VLNET. For certain types of messages (positions, body postures, facial expressions) a dead-reckoning algorithm is implemented within the Communication Process [Capin 97-1]. The video data from the Video Engine is a special case and is handled using a separate communication channel. It is given lower priority than the other data. By isolating the communications in this separate process, and by keeping the VLNET Server relatively simple, we leave the possibility to switch relatively easily to a completely different network topology, e.g. multicasting instead of client/server.

3.2.1.4. The Data Base Process

The Data Base Process takes care of the off-line fetching and loading of objects and user representations. By keeping these time consuming operations in the separate process non-blocking operation of the system is assured.

<u>3.2.2. The Drivers</u>

The drivers provide the simple and flexible means to access and control all the complex functionalities of VLNET. Simple, because each driver is programmed using a very small API that basically consists of exchanging crucial data with VLNET through shared memory. Flexible, because using various combinations of drivers it is possible to support all sorts of input devices ranging from the mouse to the camera with complex gesture recognition software, to control all the movements of the body and face using those devices, to control objects in the environment and stream video textures to them, to build any amount of artificial intelligence in order to produce autonomous or semi-autonomous virtual humans in the networked virtual environment.

The Drivers are directly tied to the Engines in the VLNET Main Process. Each engine provides a shared memory interface to which a driver can connect. Most drivers are optional and the system will provide minimal functionality (plain navigation and manipulation of objects) without any drivers. The drivers are spawned by the VLNET Main Process on the beginning of the session, based on the command line where all combinations of drivers can be specified. The drivers can be spawned on the local host or on a remote host, in which case the transparent networking interface processes are inserted on both hosts. In a simple case, as with most drivers shown in figure 3, a driver controls only one engine. However, it is possible to control more than one engine with a single driver, insuring synchronization and cooperation.

3.2.2.1. Driver types

The **Facial Expression Driver** is used to control expressions of the user's face. The expressions are defined either using the Minimal Perceptible Actions (MPAs) [Kalra 93]. The MPAs provide a complete set of basic facial actions, and using them it is possible to define any facial expression. Examples of existing facial expression drivers include a driver that uses the video signal from the camera to track facial features and map them into the MPAs describing expressions [Pandzic 94] and a driver that lets the user choose from a menu of expressions or emotions to show on his face. The facial expression driver is optional.

The **Body Posture Driver** 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 used in VLNET. An obvious example of using this driver is direct motion control using magnetic trackers [Molet 96]. A more complex driver is used to control body motion in a general case when trackers are not used. This driver connects also to the Navigation Interface and uses the navigation trajectory to generate the walking motion and arm motion. It also imposes constraints on the Navigation Driver, e.g. not allowing the hand to move further then arm length or take an unnatural posture. This is the standard body posture driver which is spawned by the system unless another driver is explicitly requested.

The **Navigation Driver** is used for navigation, hand movement, head movement, basic object manipulation and basic system control. 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. Typical examples are a spaceball driver, tracker+glove driver, extended mouse driver (with GUI console). There is also an experimental facial navigation driver letting the user navigate using his head movements and facial expressions tracked by a camera [Pandzic 94]. If no navigation driver is used, internal mouse navigation is activated within the Navigation Engine.

The **Object Behavior Driver** is used to control the behavior of objects. Currently it is limited to controlling motion and scaling. Examples include the control of a ball in a tennis game and the control of graphical representation of stock values in a virtual stock exchange.

The **Video Driver** 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. This type of driver is also used to stream facial video on the user's face for facial communication [Pandzic 96-1].

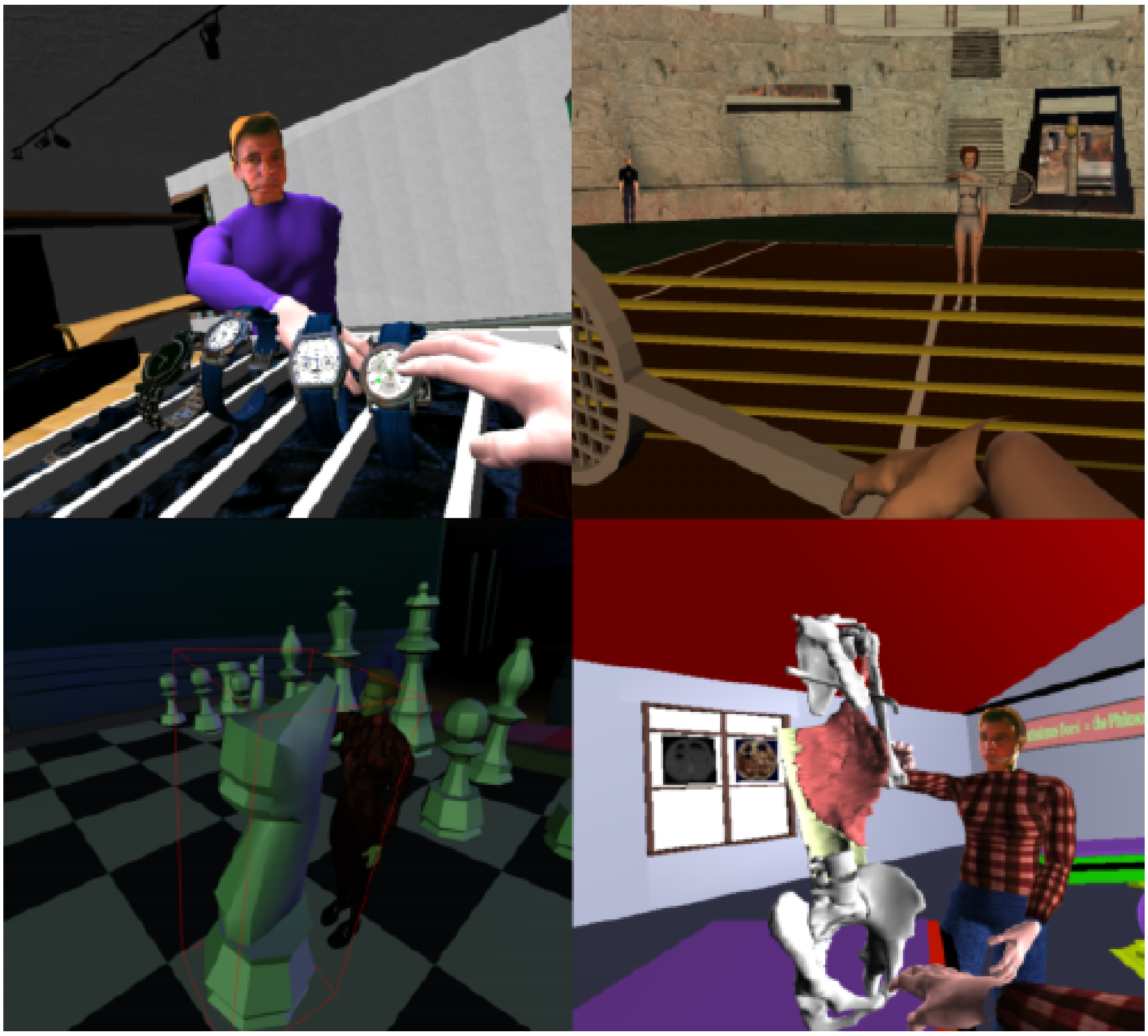


Figure 4: Snapshots from applications

4. Results

4.1. Applications

Several experimental applications were developed using the VLNET system. Some snapshots are presented in figure 4.

<u>4.1.1. Entertainment</u>

NCVE systems lend themselves to development of all sorts of multi user games. We had successful demonstrations of chess and other games played between Switzerland and Singapore, as well as between Switzerland and several European countries.

A virtual tennis game has been developed [Noser 96] where the user plays against an opponent who is an autonomous virtual human. The referee is also an autonomous virtual human capable of refereeing the game and communicating the points, faults etc. by voice.

Currently a multi user adventure game is under development.

<u>4.1.2. Teleshopping</u>

In collaboration with Chopard, a watch company in Geneva, a teleshopping application was successfully demonstrated between Geneva and Singapore. The users were able to communicate with each other within a rich virtual environment representing a watch gallery with several watches exposed. They could examine the watches together, exchange bracelets, and finally choose a watch.

4.1.3. Medical education

In collaboration with our colleagues working on a medical project we have used the 3D data of human organs, muscles and skeleton reconstructed from MRI images [Beylot 96] to build a small application in the field of medical education. The goal is to teach a student the position, properties and function of a particular muscle. In the virtual classroom several tools are at the disposition of the professor. MRI, CT and anatomical slice images are on the walls showing the slices where the muscle is visible. A 3D reconstruction of the skeleton is available together with the muscle, allowing to examine the shape of the muscle and the points of attachment to the skeleton. Finally, an autonomous virtual human with a simple behavior is there to demonstrate the movements resulting from a contraction of the discussed muscle.

4.1.4. Stock exchange

Currently an application is being developed to visualize in 3D the real time updates of stock exchange data and allow interactions of users with the data and with each other.

4.2. Performance and networking

Considering the graphical and computational complexity of the human representations used in VLNET, we are currently not aiming for a system scalable to large number of users, but rather trying to obtain a high quality experience for a small number of users. The graphs of performance and network traffic (figures 6 and 7) show that the system is indeed not scalable to any larger number of participants. Nevertheless, the results are reasonable for a small number of users and, more importantly, their analysis permits to gain insight to the steps needed to insure a better scalability.

4.2.1. Experiment design

In order to conveniently simulate a growing number of users we have designed simple drivers to generate some reasonable motion and facial expressions. The navigation driver we used generates a random motion within a room, and the facial expressions driver generates a facial animation sequence repeatedly. By launching several clients on different hosts using these drivers we can easily simulate a session with a number of persons walking in the room and emoting using their faces.

Although we looked for a way to simulate something close to a real multi-user session in a controlled way, there is a difference in the fact that simulated users move their bodies and faces all the time. In a real session, the real users would probably make pauses and thus generate less activity. Therefore we expect somewhat better results in a real session then the ones shown here, although for practical reasons we did not make such tests with real users.

In order to evaluate the overhead induced by the use of high level body representation in the NCVE, we have undertaken three series of measurements: with full body representation, simplified body representation and without a body representation. The full body representation involves complex graphical representation (approx. 10 000 polygons) and deformation algorithms. The simplified body representation consists of a body with reduced graphical complexity (approx. 1500 polygons), with facial deformations and with a simplified body animation based on displacement of rigid body elements (no deformation). The tests without the body representation were made for the sake of comparison. To mark the positions of users we used simple geometric shapes. No facial or body animation is involved. Figure 5 shows some snapshots from the measurement sessions.

The network traffic (figure 7) was measured on the server. We have measured incoming and outgoing traffic in kilobits per second.

For the performance measures (figure 6), we connected one standard, user controlled client to be able to control the point of view. The rest of the clients were user simulations as described above. Performance is measured in number of frames per second. Since it varies depending on the point of view because of rendering and computation culling, we needed to insure consistent results. We have chosen to take measurements with a view where all the user embodiments are within the field of view, which represents the minimal performance for a given number of users in the scene. The performance was measured on a Silicon Graphics Indigo Maximum Impact workstation with 200 MHz R4400 processor and 128M RAM.

4.2.2. Analysis of performance results

Figure 6 shows the variation of performance with respect to the number of users in the simulation with different complexities of body representation as explained in the previous subsection. It is important to notice that this is the minimal performance, i.e. the one measured at the moment when all the users' embodiments are actually within the field of view. Rendering and computation culling boost the performance when the user is not looking at all the other embodiments because they are not rendered, and the deformation calculations are not performed for them. The embodiments that are out of the field of view do not decrease performance significantly, which means that the graph in figure 6 can be also interpreted as the peak performance when looking at N users, regardless of the total number of users in the scene. For example, even if the total number of participants is 9, performance can still be 9 frames per second at the moment when only one user is in the field of view. This makes the system much more usable then the initial look at the graphs might suggest.

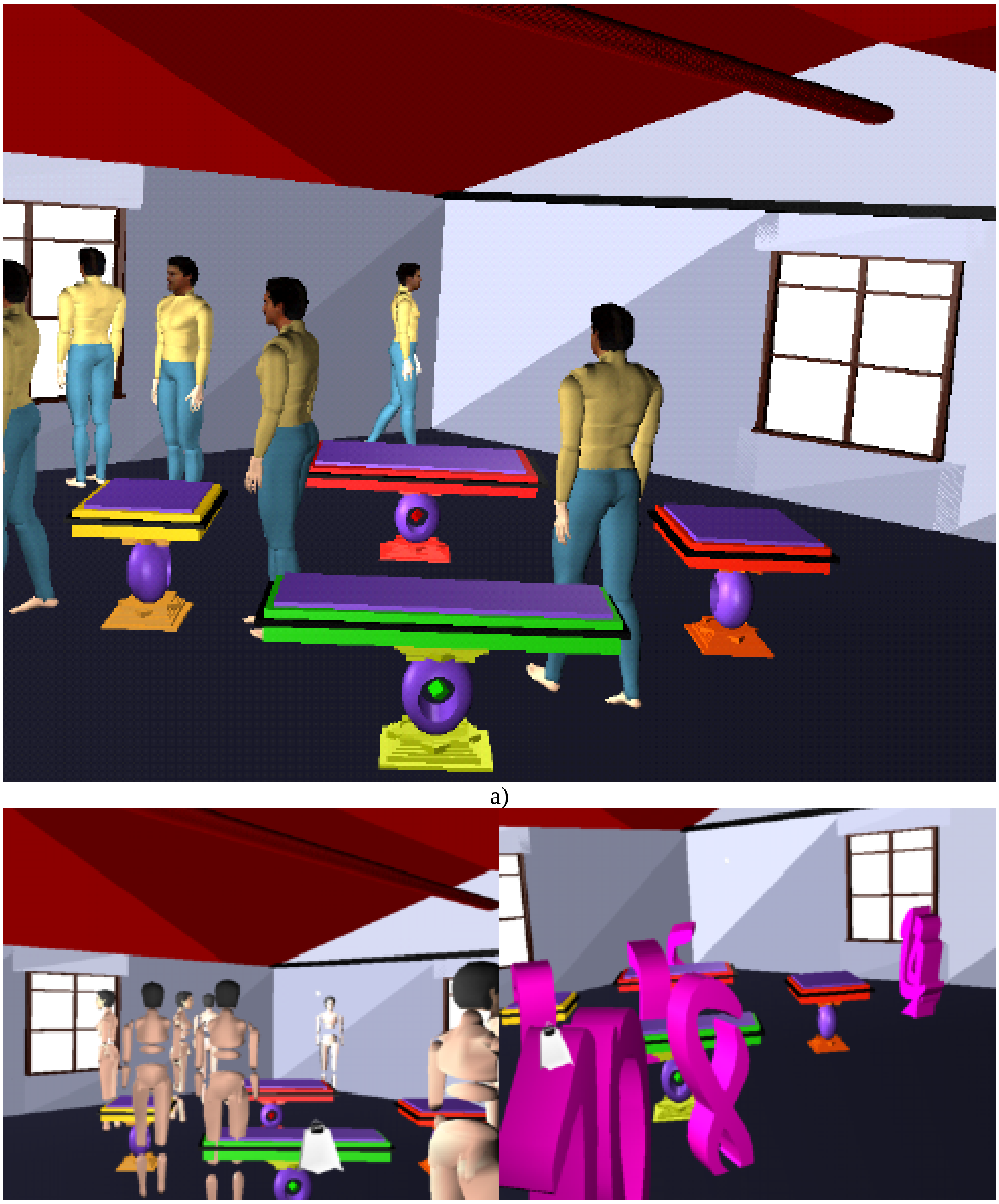


Figure 5: Snapshots from performance and network measurements: a) full bodies; b) simplified bodies; c) no body representation

We have also measured the percentage of time spent on two main tasks: rendering and application-specific computation. With any number of users, the rendering takes around 65 % of the time, and the rest is spent on the application-specific computation, i.e. mostly the deformation of faces and bodies. For the case of simplified body representation, the percentage is 58 %. On machines with less powerful graphics hardware, an even greater percentage of the total time is dedicated to rendering.

The results show that the use of complex body representation induces a considerable overhead on the rendering side and somewhat less on the computation side. The fact that the performance can be boosted significantly by simply changing the body representation proves that the system framework does not induce an overhead in itself. The system is scalable in the sense that for a particular hardware and requirements a setup can be found to produce satisfying performance. Most importantly, this proves that the system should lend itself to the implementation of extended Level of Detail [Rohlf 94] techniques managing automatically the balance between performance and quality of the human representation.

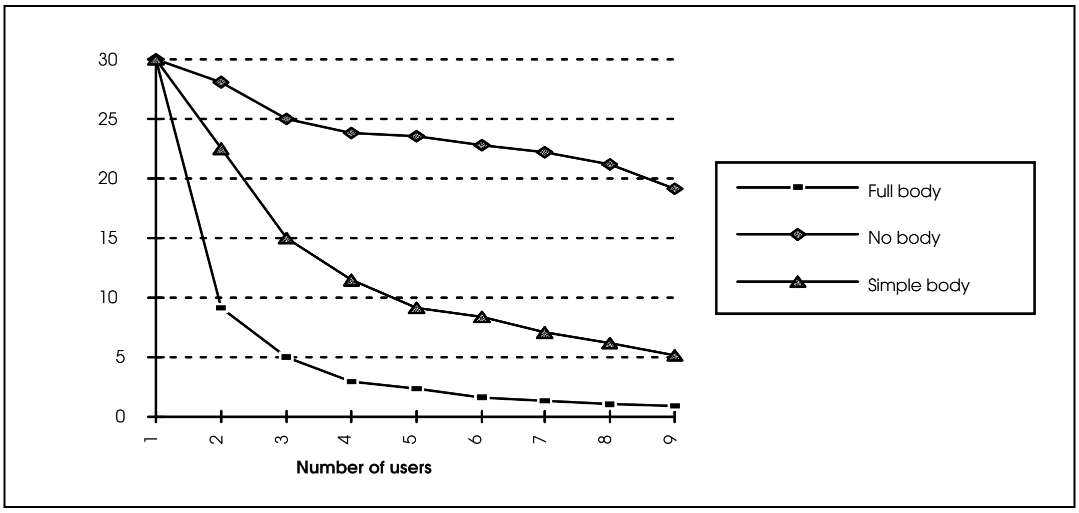


Figure 6: Minimal performance with respect to the number of users in the session

4.2.3. Analysis of the network results

Figure 7 shows the bit rates measured on the VLNET server during a session with varying number of simulated users walking within a room as described in the subsection on experiment design. We have measured separately the incoming and outgoing traffic, then calculated the total.

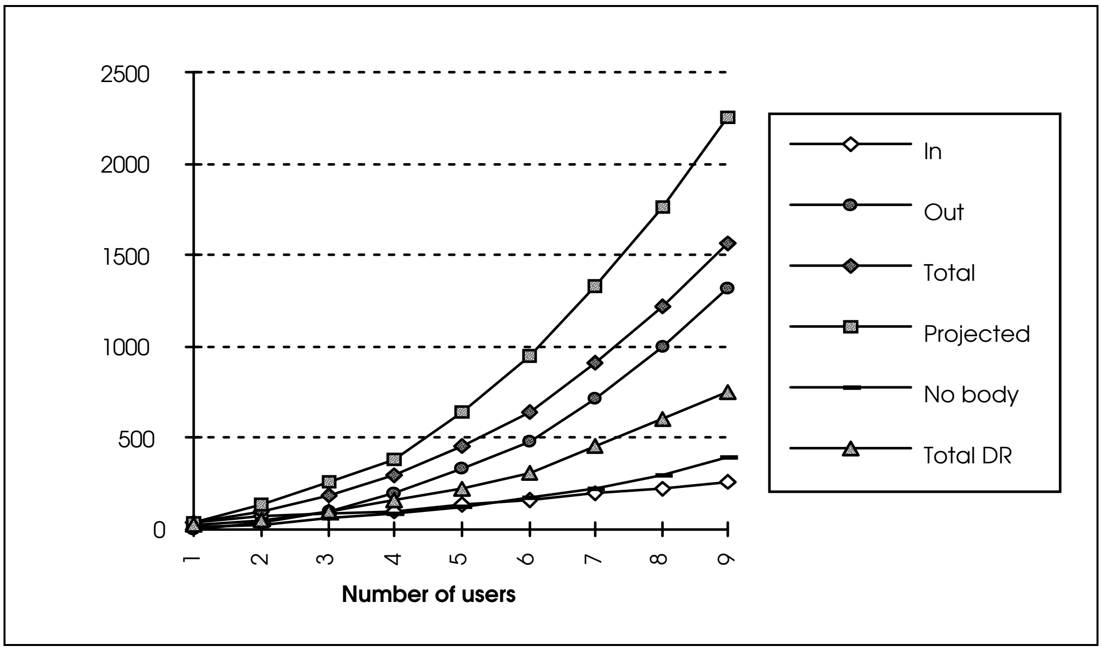


Figure 7: Network traffic measurements with respect to the number of users in the session

Obviously, the incoming traffic grows linearly with the number of users, each user generating the same bitrate. It can be remarked that the bitrate generated per user is roughly 30 Kbit/sec covering the transmission of body positions, body postures and facial expressions.

It is worthwhile remarking that the In traffic curve measured at the server corresponds also to the maximal incoming traffic on a client. The client will receive the incoming bitstream corresponding to the graph in the situations where all the embodiments of other users are in the viewing frustum of the local user, i.e. in the worst case. Otherwise, when looking at N users the incoming traffic will correspond to N users at the graph. This is similar to the situation with performance measurements.

The outgoing traffic represents the distribution of the data to all the clients that need it. The Total traffic is the sum of incoming and outgoing traffic. The Projected traffic curve represents the traffic calculated mathematically for the case of total distribution (all to all). The filtering technique at the server (see subsection 3.1.) insures that the messages are distributed only on as-needed basis and keeps the Total curve well below the Projected curve. Further reduction is achieved using the dead-reckoning technique [Capin 97-1], as illustrated by the curve labeled "Total DR".

Network traffic is the same when using full and simplified body representations, because same messages are transferred. In the case when no body representation is used, less network traffic is generated because there is no need to send messages about body postures and facial expressions. A user without a body representation sends only position updates, generating approximately 8 Kbit/sec. The total traffic curve without a body representation is shown in figure 7 with label "No body".

5. Conclusions

We have discussed problems involved with introducing high level human representations in the Networked Collaborative Virtual Environments and presented one solution to this problem - the Virtual Life Network. The system's modular architecture, the function of each component and their interconnections were described in detail. Various experimental applications highlighted in the results section show that this kind of system lends itself to a wide variety of collaborative applications. The performance and network measurements quantify the overhead induced on the CPU and the network by the introduction of complex virtual humans. They show that the overhead is considerable, but at the same time they show the potential of the system towards scalability.

6. Future work

The work on more realistic real time body representations and performance improvements, as well as methods for capturing and interpreting body and face movement, is ongoing. The performance results indicate a path for future research in the direction of more scalable body representations. Level of Detail technique can be extended to the body representation not only in graphical, but also in procedural sense by switching to simpler animation/deformation techniques for lower levels of detail. This method can be further extended to act in reducing network traffic.

7. Acknowledgments

This research is financed by "Le Programme Prioritaire en Telecommunications de Fonds National Suisse de la Recherche Scientifique" and the TEN-IBC project VISINET.

Numerous colleagues at LIG and MIRALab have directly or indirectly helped this research by providing libraries, body and environment models, scenarios for applications, in particular Eric Chauvineau, Hansrudi Noser, Marlene Poizat, Laurence Suhner and Jean-Claude Moussaly. Dr. Jean Fasel, CMU, University of Geneva, provided the advice for the medical application.

8. References

[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.

[Beylot 96] Beylot P., Gingins P., Kalra P., Magnenat-Thalmann N., Maurel W., Thalmann D., Fasel J. "3D Interactive Topological Modeling using Visible Human Dataset", *Proceedings of EUROGRAPHICS 96*, Poitiers, France, 1996.

[Boulic 90] 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.

[Boulic 95] 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.

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

[Capin 97] 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 (to appear).

[Capin 97-1] Capin T.K., Pandzic I.S., Thalmann D., Magnenat Thalmann N. "A Dead-Reckoning Algorithm for Virtual Human Figures", *Proc. VRAIS'97*, IEEE Press, 1997 (to appear).

[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.

[Kalra 93] Kalra P. "An Interactive Multimodal Facial Animation System", *PhD Thesis nr. 1183*, EPFL, 1993

[Macedonia 94] 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.

[Molet96] Molet T., Boulic R., Thalmann D., "A Real Time Anatomical Converter for Human Motion Capture", *Proc. of Eurographics Workshop on Computer Animation and Simulation*, 1996.

[Noser 96] Noser H., Pandzic I.S., Capin T.K., Magnenat Thalmann N., Thalmann D. "Playing Games through the Virtual Life Network", *Proceedings of Artificial Life V*, Nara, Japan, 1996.

[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.

[Pandzic 94] Pandzic I.S., Kalra P., Magnenat-Thalmann N., Thalmann D., "Real-Time Facial Interaction", *Displays*, Vol. 15, No 3, 1994.

[Pandzic96] Pandzic I.S., Capin T.K., Magnenat Thalmann N., Thalmann D. "Motor functions in the VLNET Body-Centered Networked Virtual Environment", *Proc. of 3rd Eurographics Workshop on Virtual Environments*, Monte Carlo, 1996.

[Pandzic96-1] Pandzic I.S., Capin T.K., Magnenat Thalmann N., Thalmann D. "Towards Natural Communication in Networked Collaborative Virtual Environments", *Proc. FIVE 96*, Pisa, Italy, 1996.

[Rohlf94] Rohlf J., Helman J., "IRIS Performer: A High Performance Multiprocessing Toolkit for Real-Time 3D Graphics", *Proc. SIGGRAPH'94*, 1994.

[Semwal96] S. K. Semwal, R. Hightower, S. Stansfield, "Closed Form and Geometric Algorithms for Real-Time Control of an Avatar", Proc. VRAIS 96. pp. 177-184.

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

[Slater 94] Slater M., Usoh M. "Body Centered Interaction in Immersive Virtual Environments", in *Artificial Life and Virtual Reality*, N. Magnenat Thalmann, D. Thalmann, eds., John Wiley, pp 1-10, 1994.

[Thalmann95] D. Thalmann, T. K. Capin, N. Magnenat Thalmann, I. S. Pandzic, “Participant, User-Guided, Autonomous Actors in the Virtual Life Network VLNET”, *Proc. ICAT/VRST ’95*, pp. 3-11.

[Thalmann96] D. Thalmann, J. Shen, E. Chauvineau, “Fast Realistic Human Body Deformations for Animation and VR Applications”, Proc. Computer Graphics International ’96, Pohang, Korea,1996.