# A Versatile Navigation Interface for Virtual Humans in Collaborative Virtual Environments

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

[1]MIRALAB - CUI
{Igor.Pandzic,Nadia.Thalmann}@cui.unige.ch
http://miralabwww.unige.ch/

[2]Computer Graphics Laboratory
Swiss Federal Institute of Technology (EPFL)
{capin, thalmann}@lig.di.epfl.ch
http://ligwww.epfl.ch/

## Abstract

Navigation within the scope of a Networked Collaborative Virtual Environment (NCVE) using an articulated body representation is more complex then just moving the viewpoint based on user input. In such context, navigation englobes problems such as mapping of user's actions on the embodiment and body constraints in addition to the usual ones such as universal support for different devices and global motion constraints. We take a broader look at the problems, clasify them and present a solution for navigation in NCVEs.

## 1. Introduction

In its basic form, navigation is a fairly simple problem: using some hardware device, ranging from a mouse to a data glove to all sorts of alternative devices, the user "flies" or "walks" through the environment. We consider the problem within the scope of Collaborative Virtual Environments using animated Virtual Humans as embodiments for users and autonomous actors. The notion of navigation is first extended to include basic manipulation of objects - picking up, carrying, dropping, the basic tasks one would want to perform in a virtual environment. Further, the problem is extended to the mapping of user's actions to the movement of his/her virtual body. We also analyze the problem of implementing constraints on navigation. Finally, support should be possible, and straightforward, for a wide variety of hardware devices that might be used for navigation. Within the framework of our Virtual Life Network (VLNET) system, we present a versatile navigation interface that covers the above problems, while at the same time allowing easy development of support for various devices and navigation paradigms.

In the next section we analyze the problems involved with navigation in the context of a Collaborative Virtual Environment with Virtual Humans used for user representation. After that we briefly describe the VLNET system in order to put into context the following section which discusses the solutions for navigation in VLNET. Finally we present conclusions and ideas for future work.

## 2. Navigation in Collaborative Virtual Environments

Basic navigation involves using some input device to control walk-through or fly-through motion. In the context of Collaborative Virtual Environments [Barrus96, Carlsson93, Macedonia94, Ohya95, Singh95, Capin97, Pandzic97], this notion is vastly extended, especially when they involve human-like embodiments for the users. In such context, navigation involves (at least) the following problems:

- walking or flying
- basic object manipulation
- mapping of actions on embodiments
- general input device support
- implementing constraints

Walking and flying represent navigation in its basic sense, allowing the user to explore the environment from any point of view.

Basic object manipulation capabilities allow the user to pick up and displace objects in the scene. This may be extended by object behaviors that can make objects react in some other way to being picked up.

Mapping of actions on the embodiment is important for two reasons. First, it allows the local user to see what he/she is doing, e.g. by seeing his hand grab an object. Second, in a multi-user session it allows users to intuitively understand what the others are doing. Mapping of actions on the embodiment involves generation of walking motion while moving, as well as generation of natural arm motion while manipulating objects.

General device support means that it should be straightforward to connect any device to the system. This implies general solutions that will accommodate different kinds of devices, e.g. *incremental* devices like Spaceball vs. *absolute* devices like magnetic trackers, devices that generate *events* like a button generating a "grab" event vs. devices generating *states* like a data glove generating a "grab" state while the fist is tightened.

Constraints are an extremely important component of any navigation. They avoid user getting lost, turning upside-down or coming into all sorts of impossible situations. They allow to tailor the navigation paradigm in a precise manner. We divide them in two groups:

- global motion constraints
- body posture constraints

The global motion constraints involve some global knowledge of the virtual world (e.g. up direction) and/or collision detection. They determine if the user can walk or fly, where it is possible to go, what are the possible orientations. A typical set of constraints for walking might include an inclination constraint keeping the user upright, a vertical collision constraint keeping him/her on the floor (and at the same time making it easy to climb/descend stairs or ramps) and

*ACM VRST '97* Lausanne Switzerland
Copyright 1997 ACM 0-89791-953-x/97/9..$3.50

a horizontal collision constraint keeping the user from going through the walls.

Body posture constraints keep the user's embodiment in natural-looking postures, e.g. the head can't wander from its position on the shoulders, the arm can reach only that far.

In the following sections we will describe how we solved the mentioned problems within the Virtual Life Network system.

## 3. Virtual Life Network (VLNET)

This section provides a brief overview of the VLNET system [Capin97, Pandzic97] in order to put into proper context the discussion on navigation in VLNET following in the next section.

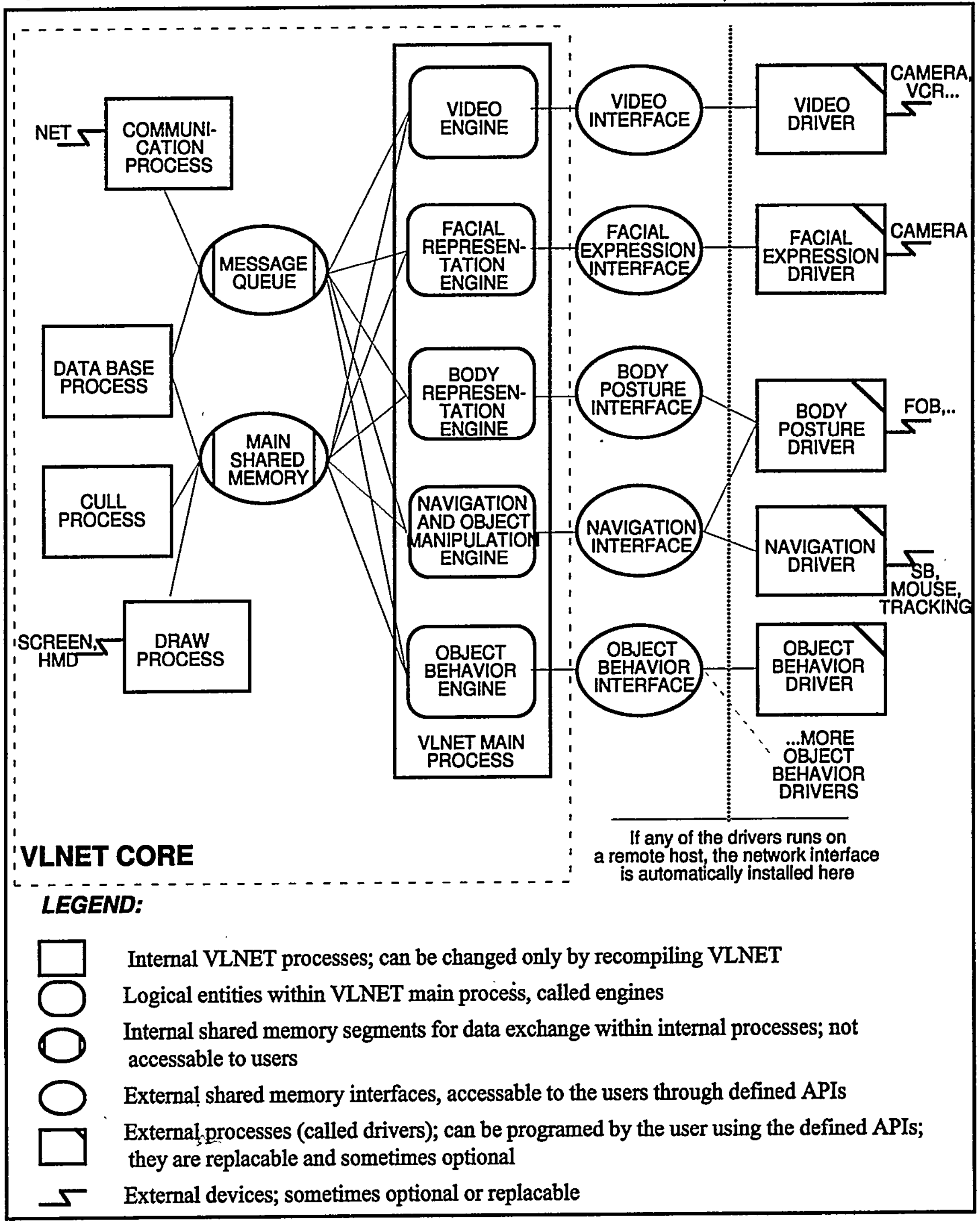
NET
COMMUNI-CATION PROCESS
DATA BASE PROCESS
CULL PROCESS
SCREEN, HMD
DRAW PROCESS
MESSAGE QUEUE
MAIN SHARED MEMORY
VIDEO ENGINE
FACIAL REPRESEN-TATION ENGINE
BODY REPRESEN-TATION ENGINE
NAVIGATION AND OBJECT MANIPULATION ENGINE
OBJECT BEHAVIOR ENGINE
VLNET MAIN PROCESS
VIDEO INTERFACE
FACIAL EXPRESSION INTERFACE
BODY POSTURE INTERFACE
NAVIGATION INTERFACE
OBJECT BEHAVIOR INTERFACE
VIDEO DRIVER
CAMERA, VCR...
FACIAL EXPRESSION DRIVER
CAMERA
BODY POSTURE DRIVER
FOB,..
NAVIGATION DRIVER
SB, MOUSE, TRACKING
OBJECT BEHAVIOR DRIVER
...MORE OBJECT BEHAVIOR DRIVERS
If any of the drivers runs on a remote host, the network interface is automatically installed here
VLNET CORE
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 1:** Virtual Life Network system overview

Virtual Life Network is a Networked Collaborative Virtual Environment system using highly realistic Virtual Humans for the participant representation. 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. The design of the VLNET client is highly modular, with functionalities split into several processes. Figure 1 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. The VLNET core consists of a number of processes performing the basic functions like object updating, rendering, networking. These processes communicate through shared memory. The VLNET main process 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. 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 *Facial Representation Engine* provides the synthetic faces with a possibility to change expressions or the facial texture. The *Video Engine* allows to stream the video textures to any object(s) in the environment.

The VLNET drivers provide the simple and flexible means to access and control all the complex functionalities of VLNET. 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. Furthermore, it is possible to control all the movements of the body and face using those devices, to control objects in the environment and to build any amount of artificial intelligence in order to produce autonomous or semi-autonomous agents in the networked virtual environment.

The Drivers are directly tied to the Engines in the VLNET Main Process, each engine providing a shared memory interface to which a driver can connect.

For more details on the VLNET system the reader is directed to [Capin97, Pandzic97].

## 4. Navigation in VLNET

We will now isolate from the big picture (figure 1) the parts of VLNET involved in navigation in order to analyze the solutions offered in VLNET to the problems posed in the beginning of the paper. Figure 2 shows the modules involved with navigation, indicating their functions and the logical data flow between them.

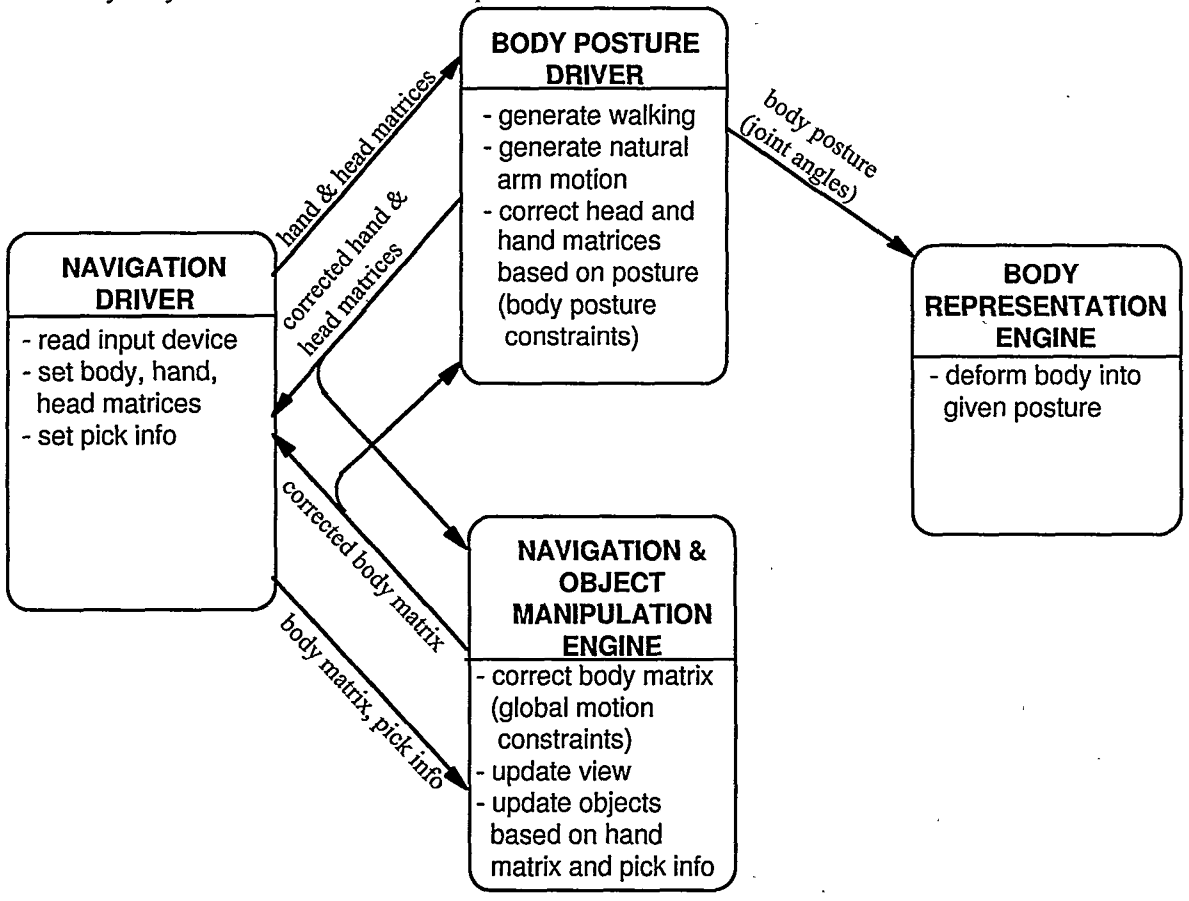
BODY POSTURE DRIVER
- generate walking
- generate natural arm motion
- correct head and hand matrices based on posture (body posture constraints)
body posture (joint angles)
BODY REPRESENTATION ENGINE
- deform body into given posture
hand & head matrices
corrected hand & head matrices
NAVIGATION DRIVER
- read input device
- set body, hand, head matrices
- set pick info
corrected body matrix
body matrix, pick info
NAVIGATION & OBJECT MANIPULATION ENGINE
- correct body matrix (global motion constraints)
- update view
- update objects based on hand matrix and pick info

**Figure 2:** VLNET modules involved in navigation and corresponding data flow

It can be observed how the problems involved with navigation are split between modules of VLNET. The device support is handled by the Navigation Driver. The Body Posture Driver handles the mapping of actions on the embodiment and the body posture constraints. The basic navigation and object manipulation, as well as global motion constraints, are handled by the Navigation and Object Manipulation Engine.

This specialization of modules results in higher flexibility and efficiency.

## 4.1 The navigation data

The data involved in navigation includes the following:

• the body matrix

• the hand matrix

• the head matrix

• pick info

The body matrix determines the global position of the user's body origin in world coordinates. The hand matrix is relative to the body origin and defines the position of the end effector used for grabbing objects, usually the hand. The head matrix is also defined with respect to the body origin and determines the position and orientation of the user's head, i.e. the user's view into the world. The pick info contains the pick and unpick flags used to control the grabbing of objects.

## 4.2 The roles of modules

The basic role of the Navigation Driver is to support a particular input device and navigation paradigm. New input devices/paradigms can be added by programming new drivers - a simple interface API is provided for that purpose.

The general function of the Body Posture Driver in VLNET is to determine the body postures and pass them to the Body Representation Engine in order to put the body in the correct posture. The body postures are generated by walking and arm motors, generating appropriate motions [Boulic 90, Pandzic 96]. In the context of navigation, the function of this driver is to implement body posture constraints. It can be replaced by a user designed driver but within this paper we will base the discussion on the standard driver provided in VLNET.

The Navigation and Object Manipulation Engine is the part of VLNET Core responsible for updating the view based on the incoming data, and for implementing basic object manipulation. It also implements the global motion constraints.

## 4.3 The data flow

The Navigation Driver reads from the input device and sets the matrices and pick info accordingly. In case of an incremental device, it uses the feedback of constraint-corrected matrices from the previous frame in order to prevent accumulation of error (at the beginning of a session the initial matrices are set by the Navigation Engine).

The Navigation Engine implements the global motion constraints using world global orientation and collision detection and corrects the body matrix to keep the body in correct uprigt orientation and keep it from colliding with obstacles.

Based on the corrected global motion expressed by the corrected body matrix, as well as hand and head positions, the Body Posture Driver generates the body posture reflecting the walking motion and arm movement. The resulting posture is procceded in terms of joints to the Body Representation Engine for body deformation. At the same time, the resulting posture determines the constraints on the head and hand matrices based on which the new, corrected matrices are generated passed to the Navigation Engine.

The Navigation Engine updates the view matrix used by the rendering pipeline based on the body and head matrices. If object grabbing is requested by the pick info, it tries to grab an object in the vicinity of the user's hand and then moves the object accordingly.

## 5. Implementation

The VLNET system is implemented on Silicon Graphics platforms using SGI Performer [Rohlf94] library. All interprocess communication passes through shared memory segments. The logical data flow between the Navigation Driver, Body Posture Driver and The Navigation Engine illustrated in figure 2 actually passes through a single shared memoy segment, shared by all three processes, that holds the complete navigation data.

## 6. Results

We have implemented several navigation drivers for different devices and paradigms. The more classical examples include a GUI-based mouse driver, the SpaceBall driver and the driver supporting head and head trackers with a data glove.

A full body tracking driver suppors control of the whole body using a larger number of magnetic trackers (12 trackers in current implementation) [Molet96] as illustrated in figure 3.

A more original experimental driver exists, using image processing techniques to track facial features of the user in front of the camera, letting the user navigate using his face [Pandzic 94].

The possibility to use the driver mechanism to implement autonomous Virtual Actors is particularly interesting. Instedad of supporting hardware devices to get input from a real human, in this kind of applications the drivers use AI algorithms to make Virtual Humans navigate and interact autonomously. An example of this approach is our experimental tennis game [Noser96] where the user plays against an autonomous virtual player while the game is refereed by another autonomous Virtual Human (figure 4).

**Figure 3:** Navigation and posture control using magnetic trackers

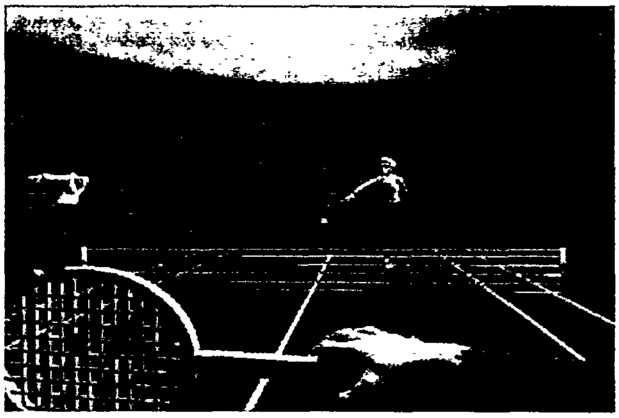
**Figure 4:** Tennis with autonomous Virtual Humans

To implement support for any of these navigation devices and paradigms it was only necessary to program new drivers, without changing the system itself. The navigation interface consisting of a simple API enabled coleagues and students without the knowledge of the complete system to implement new navigation paradigms quickly.

## 7. Conclusions and future work

We have analyzed the problems involved with navigation in a broader sense within the context of Collaborative Virtual Environments involving human-like embodiments. We have presented solutions to the analyzed problems within the framework of the Virtual Life Network system, showing how the functionalities are split between different modules of VLNET, achieving great flexibility, which is also confirmed by different navigation modes implemented and presented in the results section.

We intend to explore new and more convenient paradigms for navigation in Virtual Environments, e.g. the hand-centered paradigm with gaze, posture and finally the whole body motion follow the movement of the hand controlled by the user.

## 8. 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 Elwin Lee, Eric Chauvineau, Hansrudi Noser, Marlene Poizat, Laurence Suhner and Jean-Claude Moussaly.

## 9. 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.

[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.

[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*, March-April 1997.

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

[Noser96] 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] I.S. Pandzic, T.K. Capin, N. Magnenat Thalmann, D. Thalmann, "Motor functions in the VLNET Body-Centered Networked Virtual Environment", *Proc. of 3rd Eurographics Workshop on Virtual Environments*, Monte Carlo, 1996.

[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)

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

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