Virtual Human Representation and Communication in VLNET Networked Virtual Environment*

Tolga K. Capin(1), Igor Sunday Pandzic(2), Hansrudi Noser(1),
Nadia Magnenat Thalmann(2), Daniel Thalmann(1)

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

(2) MIRALAB – CUI, University of Geneva
CH-1211 Geneva 4, Switzerland

The pace in computing, graphics and networking technologies together with the demand from real-life applications made it a requirement to develop more realistic virtual environments (VEs). Realism not only includes believable appearance and simulation of the virtual world, but also implies the natural representation of participants. This representation fulfills several functions:

- The visual embodiment of the user,
- The means of interaction with the world,
- The means of feeling various attributes of the world using the senses.

The realism in participant representation involves two elements: believable appearance and realistic movements. This becomes even more important in multiuser networked virtual environments (NVE), as participants' representation is used for communication. A NVE can be defined as a single environment which is shared by multiple participants connected from a different host. The local program of the participants typically store the whole or a subset of the scene description, and they use their own avatars to move around the scene and render from their own viewpoint. This avatar representation in NVEs has crucial functions in addition to those of single-user virtual environments:

- perception (to see if anyone is around)
- localization (to see where the other person is)
- identification (to recognize the person)
- visualization of others' interest focus (to see where the person's attention is directed)
- visualization of others' actions (to see what the other person is doing and what she means through gestures)
- social representation of self through decoration of the avatar (to know what the other participants' task or status is).

Using virtual human figures for avatar representation fulfills these functionalities with realism, as it provides the direct relationship between how we control our avatar in the virtual world and how our avatar moves related to this control. Even with limited sensor information, a virtual human frame that reflects the activities of the user, can be constructed in the virtual world. Slater and Usoh [1] indicate that using a virtual body, even if simple, increases the sense of presence in the virtual world.

NVEs with virtual humans is emerging from two threads of research with a bottom-up tendency. First, over the past several years, many NVE systems have been created using various types of network topologies and computer architectures [2][3]. The practice is to bring together different previously-developed monolithic applications within one standard interface; and consists of building multiple logical or actual processes that handle a separate element of the VE. Second, at the same time, virtual human research has developed to the level to provide realistic-looking virtual humans that can be animated with believable behaviors in multiple levels of control. Inserting virtual humans in the NVE is a complex task.

The main issues are:

- selecting a scalable architecture to combine these two complex systems,
- modeling the virtual human with believable appearance for interactive manipulation,
- animating it with minimal number of sensors to have maximal behavioral realism,
- investigating different methods to decrease the networking requirements for exchanging complex virtual human information.

Published in VLNET, IEEE Computer Graphics and Applications, Vol. 17, N°2, 1997, pp42-53

In this paper, we survey problems and solutions for these points, taking the VLNET (Virtual Life Network) system as a reference model. The VLNET system has been developed at MIRALab at University of Geneva, and Computer Graphics Laboratory at Swiss Federal Institute of Technology. In VLNET, we try to integrate artificial life techniques with virtual reality techniques in order to create truly virtual environments shared by real people, and with autonomous living virtual humans with their own behavior, which can perceive the environment and interact with participants. Figure 1 shows example applications of the system.

Fig 1. Example applications of Networked Virtual Environments
(networked games, virtual meetings, teleshopping, medical education)

1 Multiprocess client architecture

Typically, the virtual environment simulation systems are complex software systems, therefore a modular design imposes itself. It is appropriate to design the runtime system as a collection of cooperating processes, each of which is responsible for its particular task. This also allows easier portability and better performance of the overall system through the decoupling of different tasks and their execution over a network of workstations or different processors on a multiprocessor machine. In VLNET, we use this multiprocess approach. Figure 2 shows the architecture of the VLNET client. We separate the two types of processes: the core VLNET processes, and external driver processes.

Within the VLNET core, the main process executes the main simulation and provides services for the basic elements of VEs to the external programs, called drivers. The display, cull and database processes are standard IRIS Performer processes, and allow asynchronous loading and display of the scene with the main simulation. The main process consists of four logical units, called engines. The role of the engine is to separate one main function in the VE to an independent module, and provide an orderly and controlled allocation of VE elements. Moreover, the engine manages this resource among various programs which are competing for the same object. The communication process is responsible for receiving and sending messages through the network, and uses incoming and outgoing message queues to implement asynchronous communication.

The object behavior engine is responsible for the requests for changing or querying the object definition and behaviors in the scene, and collision detection among them. The navigation engine connects the user input to the navigation, picking and manipulation of objects. The input is in the form of relative and absolute matrices of the global position of the body, and the requests for picking or releasing an object.

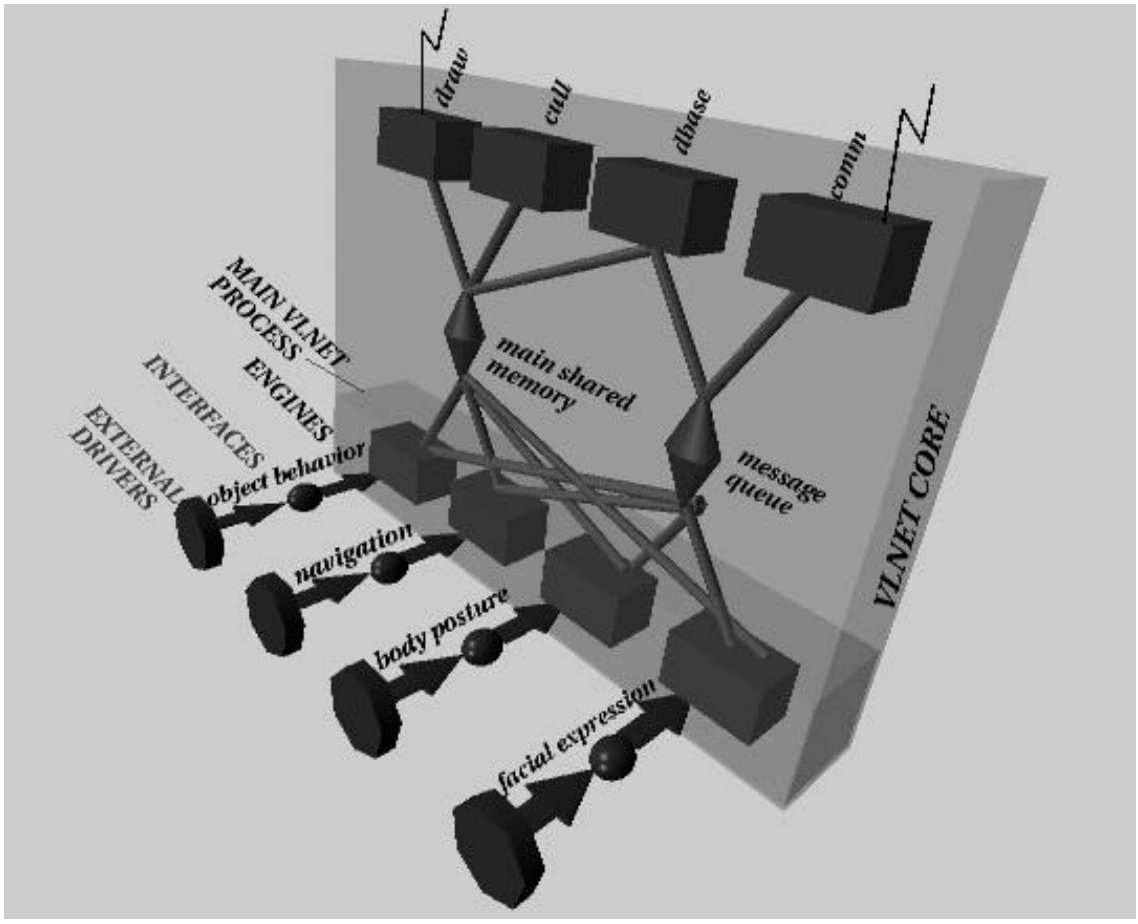


Fig. 2. Client architecture of the VLNET system and its interface with external processes

Similarly, the face and body representation engines are specialized for the virtual human figure. The face engine is responsible for bridging between VLNET and external face drivers. The engine obtains the camera video images or face model parameters discussed below from the external face driver, and places in VLNET internal shared memory and outgoing message queue.

The body representation engine has an external interface for the body posture including joint angles or global positioning parameters, and high level parameters to animate the body. The role of this engine is to provide possibilities to define multiple levels of control for the human body, and to merge the output of different external body drivers to a single final posture.

There exist different data dependencies between the external face and body process and other components of the environment simulation system. The virtual humans are able to have a behavior, which means they must have a manner of coding and using the environment's effects on the virtual body. The virtual human should be equipped with visual, tactile, and auditory sensors. These sensors are used as a basis for implementing everyday human behavior such as visually-directed locomotion, handling objects, and responding to sounds and utterances. Similarly, we want the virtual human to be able to act on the environment, for example the participant can grasp and reposition an object. In the next section, we discuss different human figure motion control methods for creating complex motion. This requires the sharing of information between the object and navigation engines, and external human processes using their external interfaces.

2 Virtual human modeling

Real-time representation and animation of virtual human figures has been a challenging and active area in computer graphics [4][5] . Typically, an articulated structure corresponding to the human skeleton is needed for the control of the body posture. Structures representing the body shape have to be attached to the skeleton, and clothes may be wrapped around the body shape.

In VLNET, we use an articulated human body model with 75 degrees of freedom, with 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. The skeleton is encapsulated with geometrical, topological, and inertial characteristics of different body limbs. The body structure has a fixed topology template of joints, and different body instances are created by scaling body limbs globally, as well as applying frontal, high and low lateral scaling, or specifying spine origin ratio between lower and upper body parts [4].

Attached to the skeleton, is a second layer that consists of blobs (metaballs) to represent muscle and skin. The method's main advantage lies in permitting us to cover the entire human body with only a small number of blobs. From this point we divide the body into 17 parts: head, neck, upper torso, lower torso, hip, left and right upper arm, lower arm, hand, upper leg, lower leg, and foot. Because of their complexity, head, hands and feet are not represented with blobs, but instead with triangle meshes. For the other parts a cross-sectional table is used for deformation. This cross-sectional table is created only once for each body by dividing each body part into a number of cross-sections and computing the outermost intersection points with the blobs. These points represent the skin contour and are stored in the body description file. During runtime the skin contour is attached to the skeleton, and at each step is interpolated around the link depending on the joint angles. From this interpolated skin contour the deformation component creates the new body triangle mesh.

There are different parameter sets for defining virtual human postures and faces:

Global Positioning Domain Parameters:
These are the global position and orientation values of particular observable points on the body, in the body coordinate system. The points are: top of head, back of neck, mid-clavicle, shoulders, elbow, wrist, hip, knee, ankle, bottom of mid-toe.

Joint Angle Domain Parameters:
These parameters comprise the joint angles defined above, connecting different body parts.

Hand and Finger Parameters:
The hand is capable of performing complicated motions and there are at least fifteen joints in the hand, not counting the carpal part. As using hand joints almost doubles the total number of degrees of freedom, we separate the hand parameters from those for other body parts.

Face Parameters
The face is generally represented differently than the other parts of the body. It is a polygon mesh model with defined regions and Free Form Deformations modeling the muscle actions [6]. It can be controlled on several levels. On the lowest level, an extensive set of *Minimal Perceptible Actions* (MPAs), closely related to muscle actions, can be directly controlled. There are 65 MPAs, and they can completely describe the facial expression. Each MPA is a basic building block facial motion parameter which allows to move a separate visual facial feature (e.g. raise eyebrow, close, upper eyelids). In a higher level, phonemes and/or facial expressions can be controlled spatially and temporally. In the highest level, complete animation scripts can be input defining speech and emotion over time. Algorithms exist to map texture on such facial model.

3 Virtual human control

The participant should animate his virtual human representation in real-time, however the human control is not straightforward: the complexity of virtual human representation needs a large number of degrees of freedom to be tracked. In addition, interaction with the environment increases this difficulty even more. Therefore, the human control should use higher level mechanisms to be able to animate the representation with maximal facility and minimal input. We can divide the virtual humans according to the methods to control them:

- Directly controlled virtual humans: the joint and face representation of the virtual human is modified directly (e.g. using sensors attached to the body) by providing the geometry directly.
- User-guided virtual humans: the external driver *guides* the virtual human by defining tasks to perform, and it uses its motor skills to perform this action by coordinated joint movements (e.g. walk, sit).
- Autonomous virtual humans: the virtual human is assumed to have an internal state which is built by its goals and sensor information from the environment, and the participant modifies this state by defining high level motivations, and state changes (e.g. turning on vision behavior).

3.1 Direct controlled virtual humans

The virtual human is required to have a natural-looking body and be animated with respect to the actual body. This corresponds to a real-time form of traditional rotoscopy. Traditional rotoscopy in animation consists of recording the motion by a specific device for each frame and using this information to generate the image by computer. Using the terminology we introduced previously [7], we call the real-time rotoscopy method a method consisting of recording input data from a VR device in real time allowing us to apply the same data at the same to a graphics object on the screen. For example, when the animator opens the fingers 3 centimeters, the hand in the virtual scene does exactly the same. In addition, playing previously recorded keyframes requires real-time input of body posture geometry. The input geometry can be given as global positioning parameters, or joint angle parameters.

A complete representation of the participant's virtual body should have the same movements as the real participant body for more immersive interaction. This can be best achieved by using a large number of sensors to track every degree of freedom in the real body. Molet et al. [8] discuss that a minimum of 14 sensors are required to manage a biomechanically correct posture, and Semwal et al. [9] present a closed-form algorithm to approximate the body using up to 10 sensors. However, many of the current VE systems use head and hand tracking. Therefore, the limited tracking information should be connected with human model information and different motion generators in order to “extrapolate” the joints of the body which are not tracked. This is more than a simple inverse kinematics problem, because there are generally multiple solutions for the joint angles to reach to the same position, and the most realistic posture should be selected. In addition, the joint constraints should be considered for setting the joint angles.

The main lowest-level approaches to this extrapolation problem are: inverse kinematics using constraints [5], closed form solutions [9], and table lookup solutions [10]. The inverse kinematics approach is based on an iterative algorithm, where an end-effector coordinate frame (for example the hand) tries to reach a goal (the reach position) coordinate frame, using a set of joints which control the end effector. The advantage of this approach is that any number of sensors can be attached to any body part, and multiple constraints can be combined through assigning weights. However, this might slow down the simulation significantly as it requires excessive computation. The closed form solution solves this problem using 10 sensors attached to the body, and solving for the joint angles analytically. The human skeleton is divided into smaller chains, and each joint angle is computed within the chain it belongs to. For example, the joint angle for the elbow is computed using the sensors attached to the upper arm and lower arm, and computing the angle between the sensor coordinate frames. However, this approach still needs ten sensors. We proposed a solution that uses previously stored experimental data. We took the arm chain as an example, and assumed that only the 6 degrees of freedom of the right hand is obtained as sensor input. The body driver controlling the arm should compute the joint angles within the right arm using this input. The arm motion makes use of experimental data obtained using sensors, and stored in a precomputed table of arm joints. This precomputed table divides the normalized volume around the body into discrete number of subvolumes, and stores a mapping from subvolumes to joint angles of the right arm. Afterwards, the normal inverse kinematics computations are performed using this posture as the starting state.

3.2 Guided virtual humans

Guided virtual humans are those which are driven by the user but which do not correspond directly to the user motion. They are based on the concept of real-time direct metaphor, a method consisting of recording input data from a VR device in real-time allowing us to produce effects of different natures but corresponding to the input data. There is no analysis of the real meaning of the input data. To understand the concept, we may take an example of traditional metaphor: the puppet control. A puppet may be defined as a doll with jointed limbs moved by wires or strings. Similarly glove-puppets are dolls of which the body can be put on the hand like a glove, the arms and head being moved by the fingers of the operator. In both cases, human fingers are used to drive the motion of the puppet.

In VLNET, an example of virtual human guidance is guided navigation. The participant uses the input devices to update the transformation of the eye position of the virtual human. This local control is used by computing the incremental change in the eye position, and estimating the rotation and velocity of the body center. The walking motor uses the instantaneous velocity of motion, to compute the walking cycle length and time, by which it computes the joint angles of the whole body. The sensor information or walking can be obtained from various types of input devices such as special gesture with DataGlove, or SpaceBall, as well as other input methods.

3.3 Autonomous virtual humans

An autonomous system is a system that is able to give to itself its proper laws, its conduct, as opposed to a heteronomous system which is driven from the outside. Guided virtual humans as introduced in the previous subsection are typically driven from the outside. Including autonomous virtual humans that interact with participants increases the real-time interaction with the environment. Therefore, it is likely to increase the sense of presence of the real participants in the environment. The autonomous virtual humans are connected to the VLNET system in the same way as human participants, and also improve the usage of the environment by providing services such as replacing missing partners, helping in navigation. As these virtual humans are not guided by the users, they should have sufficient behaviors to act autonomously to accomplish their tasks. This requires building behaviors for motion, as well as appropriate mechanisms for interaction.

Our autonomous virtual humans are able to have a behavior, which means they must have a manner of conducting themselves. Behavior is not only reacting to the environment but should also include the flow of information by which the environment acts on the living creature as well as the way the creature codes and uses this information. Behavior of autonomous virtual humans is based on their perception of the environment.

3.4 Combining Motions

The different motion generators that we discussed above, output their results as new joint angles between connecting limbs. As previously discussed, external driver programs can be attached to the human driver engine. Normally, more than one external driver should be able to connect to the human posture interface, and the task of the engine is to resolve conflicts among the external drivers. For example, while the walking motor updates the lower body, the grasping program might control the right arm branch of the body. The human posture engine should convert these motions' effects to a single virtual body posture.

Motion combination requires that human posture interface contains parameters for each external driver in addition to body control data. The external driver should be able to define its range within the body parts, and the weight of this driver’s output on the final posture for this range. For our initial implementation, we divided the virtual body hierarchy into 8 parts: torso, neck-head, left and right arms, legs (including feet), and hands. Only one driver can modify each part. In order to control the body part, the external driver should lock this part. This is done by storing a lock identifier in the interface, and at each time frame, the external processes update the body parts that they have locked. This approach prevents us to use multiple processes to update the same body part, and therefore it is limiting. In the current development, we are incorporating a motion combination algorithm based on weights and priorities to the output of different external processes [11].

It becomes more complicated to combine different motions if some of the external drivers contain goal directed motion, as the final posture should satisfy the condition that the goal is reached. For example, when the hand is tracked by an external posture driver while another external driver plays a previously recorded keyframe, the hand position in the posture should be in the tracked position of the actual hand. Therefore, the motion combination should consider correcting the direct-updated posture, so that the difference between the goal position and the effector is minimal.

4 Facial communication

We discuss four methods of integrating facial expressions in an NVE: video-texturing of the face, model-based coding of facial expressions, lip movement synthesis from speech and predefined expressions or animations.

4.1 Video-texturing of the face

In this approach the video sequence of the user's face is continuously texture mapped on the face of the virtual human. The user must be in front of the camera, in such a position that the camera captures his head and shoulders, possibly together with the rest of the body. A simple and fast image analysis algorithm is used to find the bounding box of the user's face within the image. The algorithm requires that head & shoulder view is provided and that the background is static (though not necessarily uniform). Thus the algorithm primarily consists of comparing each image with the original image of the background. Since the background is static, any change in the image is caused by the presence of the user, so it is fairly easy to detect his/her position. This allows the user a reasonably free movement in front of

the camera without the facial image being lost. The video capture and analysis is performed by a special Facial Expression Driver.

Each facial image in the video sequence is compressed by the Driver using SGI Compression Library and the compressed images are passed to the Facial Representation Engine of VLNET, then redirected to the Communication Process. On the receiving end, the images are received by the Communication process, decompressed by the Data Base process and texture-mapped on the face of the virtual human representing the user. Currently we use a simple frontal projection for texture mapping. A simplified head model with attenuated features is used. This allows for less precise texture mapping. If the head model with all the facial features is used, any misalignment of the topological features in the 3D model and the features in the texture produces quite unnatural artifacts. The only way to avoid this is to have the coordinates of characteristic feature points in the image which can be used to calculate the texture coordinates in such a way that the features in the image are aligned with the topology. This is called texture fitting. However, currently our texture fitting algorithm does not work in real time. Figure 3 illustrates the video texturing of the face, showing the original images of the user and the corresponding images of the Virtual Human representation.

Fig. 3. Continuous real-time texture mapping of the actual face on the virtual human's face.

4.2 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**.** 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. Accurate recognition and analysis of facial expressions from video sequence requires detailed measurements of facial features. Currently, it is computationally expensive to perform these measurements precisely. As our primary concern has been to extract the features in real time, we have focused our attention on recognition and analysis of only a few facial features. The recognition method relies on the "soft mask", which is a set of points adjusted interactively by the user on the image of the face. Using the mask, various characteristic measures of the face are calculated at the time of initialization. Color samples of the skin, background, hair etc., are also registered. Recognition of the facial features is primarily based on color sample identification and edge detection. Based on the characteristics of human face, variations of these methods are used in order to find the optimal adaptation for the particular case of each facial feature. Special care is taken to make the recognition of one frame independent from the recognition of the previous one in order to avoid the accumulation of error. The data extracted from the previous frame is used only for the features that are relatively easy to track (e.g. the neck edges).

The set of extracted parameters includes:
- vertical head rotation (nod)
- horizontal head rotation (turn)
- head inclination (roll)
- aperture of the eyes
- horizontal position of the iris
- eyebrow elevation
- distance between the eyebrows (eyebrow squeeze)
- jaw rotation
- mouth aperture
- mouth stretch/squeeze

In our implementation, the analysis is performed by a special Facial Expression Driver. The extracted parameters are easily translated into Minimal Perceptible Actions, which are passed to the Facial Representation Engine, then to the Communication process, where they are packed into a standard VLNET message packet and transmitted. On the receiving end, the Facial Representation Engine receives messages containing facial expressions described by MPAs and performs the facial animation accordingly. Figure 4 illustrates this method with a sequence of original images of the user (with overlaid recognition indicators) and the corresponding images of the synthesized face.

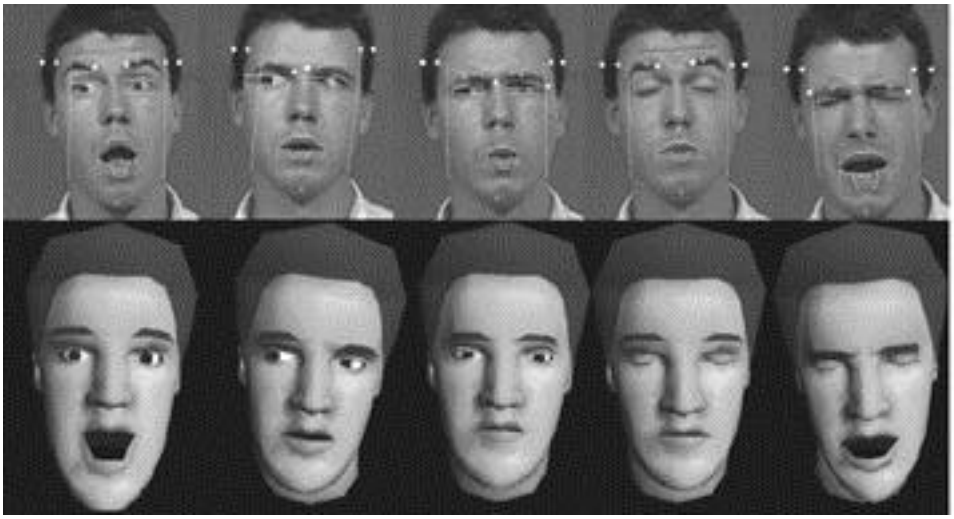


Fig. 4. Model-based coding of the face. The remote virtual human representation replicates the real participant's facial expressions, using the communicated parameters.

4.3 Lip movement synthesis from speech

It might not always be practical for the user to be in front of the camera (e.g. if he doesn't have one, or if he wants to use an HMD). Nevertheless, the facial communication does not have to be abandoned. Lavagetto [12] shows that it is possible to extract visual parameters of the lip movement by analyzing the audio signal of the speech. An application doing such recognition and generating MPAs for the control of the face can be connected with the VE program as the Facial Expression Driver, and the Facial Representation Engine will be able to synthesize the face with the appropriate lip movement. A very primitive version of such system would just open and close the mouth when there is any speech, allowing the participants to know who is speaking. A more sophisticated system would be able to actually synthesize a realistic lip movement which is an important aid for speech understanding.

4.4 Predefined expressions or animations

In this approach the user can simply choose between a set of predefined facial expressions or movements (animations). The choice can be done from the keyboard through a set of "smileys" similar to the ones used in e-mail messages. The Facial Expression Driver in this case stores a set of defined expressions and animations and just feeds them to the Facial Representation Engine as the user selects them.

5 Networking

The articulated structure of the human body together with the face introduces a new complexity in the usage of the network resources because the size of a message needed to convey the body posture is greater than the one needed for simple, non-articulated objects. This might create a significant overhead in communication, especially as the number of participants in the simulation increases. In order to reduce this overhead it is possible to communicate the body postures in more compact forms, accepting some loss of accuracy in the posture definition. This is not the only trade-off to be considered when choosing the optimal approach. Conversions between different forms of posture definition require potentially expensive computations which might induce more overhead in computation than was reduced in communication. The choice will also depend on the quality and quantity of raw data available from the input devices, the posture accuracy required by the application and the projected number of participants in the simulation.

For the networking analysis, we separate the discussion into body and faces, as they use different control methods and as they use different channels for communication. In any case, we can decompose the communication into three phases: coding, transmission, and decoding of the data. The transmission lag for a message will be the sum of the lag of all these phases. In addition, each message type contains an accuracy loss of data which is a trade-off to decrease the lag. In this section, we analyze different message types with respect to the following aspects:

- *coding computation at the sender site:* we evaluate the amount of computation needed in order to convert the input data into the message to be sent, at the sending site;

- *bitrate requirements:* we evaluate the bandwidth requirements for different parameters to describe the motion. We assume a minimum limit for real-time computation as 10 frames/second.

- *decoding computation at the receiver site:* we evaluate the amount of computation needed to interpret the message and obtain the body posture(s) for display at the receiving site. The weight of this computation on the simulation is typically more than the one at the sender site because the messages from a potentially large number of participants have to be processed contrary to coding which is done only for the locally controlled figure.

- *accuracy loss:* We evaluate the loss of accuracy of the body posture with respect to the original input data. This is typically the trade-off to be considered against decreasing coding/decoding computations and transmission overhead.

We compare these issues in Figure 5 for various types of human body motion control. We consider four types of message packets that can be used to convey the body posture information:

- *global positioning parameters*: The global positioning of 17 body parts can be sent as 3 rotation and 3 translation values. This data can be used directly to display the body by-passing the conversion to joint angles.

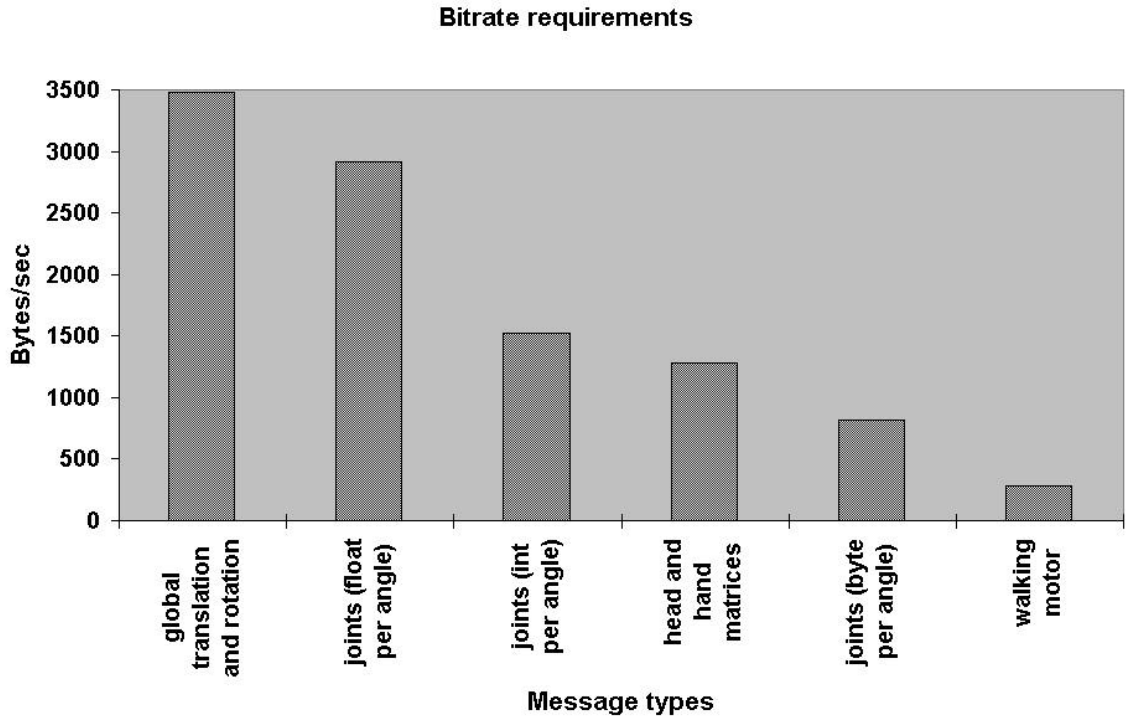
- *joint angles*. These values are the degrees of freedom comprising the body, each represented by a floating point value. We evaluate three possibilities for the message type: the actual floating point representation of the angle, and 2-byte integer and 1-byte angle information, discretized between 0 and 360. This data has to be transformed into global positioning for display.

- *end-effector matrices.* A 4x4 floating point matrix is used to determine the position of the end effectors, in this example the head and the right hand. An inverse kinematics with two end-effectors (head and hand) has to be applied to the received message to obtain the final posture.

- *state information.* Only the high level state information is conveyed which makes the messages small. Moreover, the messages are sent only when state changes. The computation complexity involved to produce the posture(s) from the state information can range from quite simple (in the case of predefined static postures like sitting, standing) through medium (in the case of predefined dynamic states like walking or running) to very complex (in the case of more complex dynamic states like searching an object). In this evaluation, we take the medium-level walking action as an example.

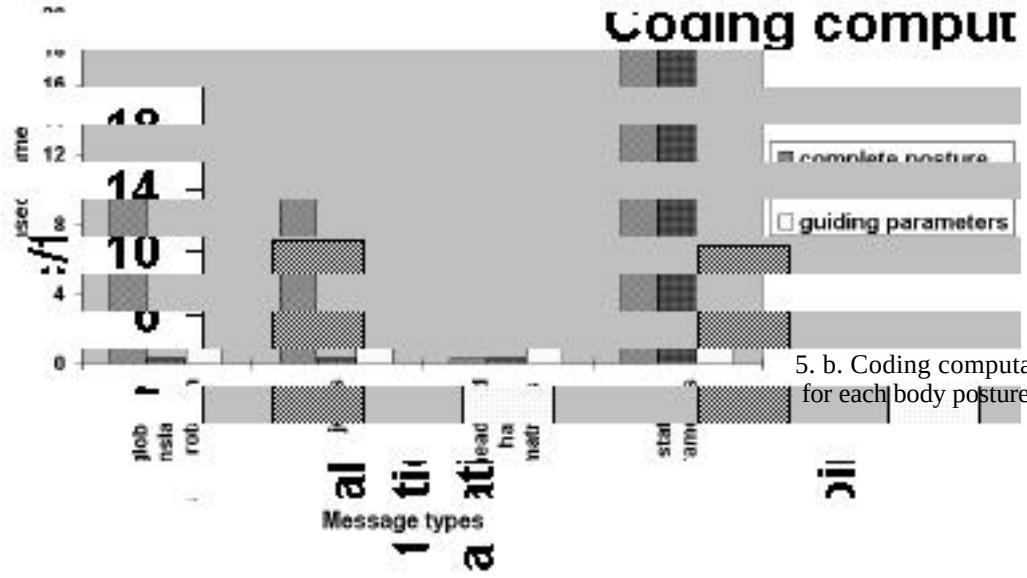
We analyze three typical situations with respect to different real-time control data: i) complete body posture data is available, ii) only head and hand end-effector data is available, iii) walking motion guiding data is available from an external driver.

Figure 5 shows the bitrate requirements, coding and decoding computations per frame, and accuracy loss for different message types. Figure 5a is calculated with the assumption of a minimum real-time speed of 10 frames/second. We see that the bitrate requirement varies between 4 Kbytes/sec and 240 bytes/second for one human body. Figures 5b and 5c show the coding and decoding results on an SGI Indigo2 Impact workstation with 250 MHz processor. The results show that there are a wide range of possibilities to define and transmit human figure information with respect to computation and bandwidth requirements. The choice of the control and message type will depend on the particular application requirements. Where high accuracy is needed (e.g. medical training applications) the transfer of body part matrices or at least end effector matrices will be required; in the large-scale simulation with numerous users it might be efficient to convey small messages containing the state information and use filtering and level of detail techniques to reduce the computational overhead. We chose the joint angles transfer as the optimal solution covering a wide range of cases since it offers fair or good results on all criteria, balancing the network and compression/decompression computational overhead (the traversal of the human hierarchy to convert joint values to transformation matrices of body parts, to be rendered on display), and accuracy loss. Using the state parameters for high level motions decreases the network bandwidth, however it requires a fast decoding process at the receiving site. The example walking motor showed the possibility of decreasing bandwidth requirements for sending motion data. Figure 5d shows the accuracy loss of posture data with varying message types and input methods. The results were computed by averaging the Euclidean distance between the corresponding body parts in the initial body posture at the sender's site and the decoded posture at the receiver's site.



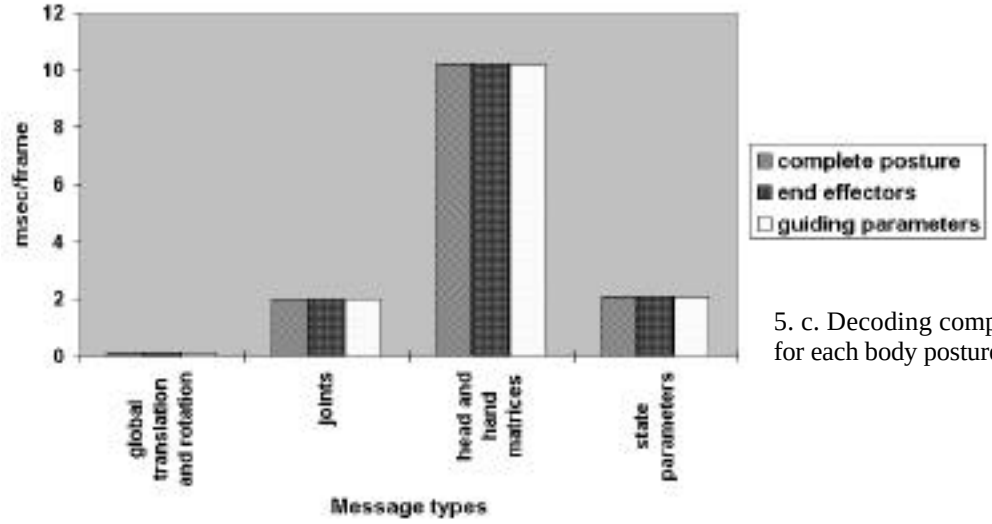
a. Bitrate requirements for each message type

Fig. 5: Networking body data



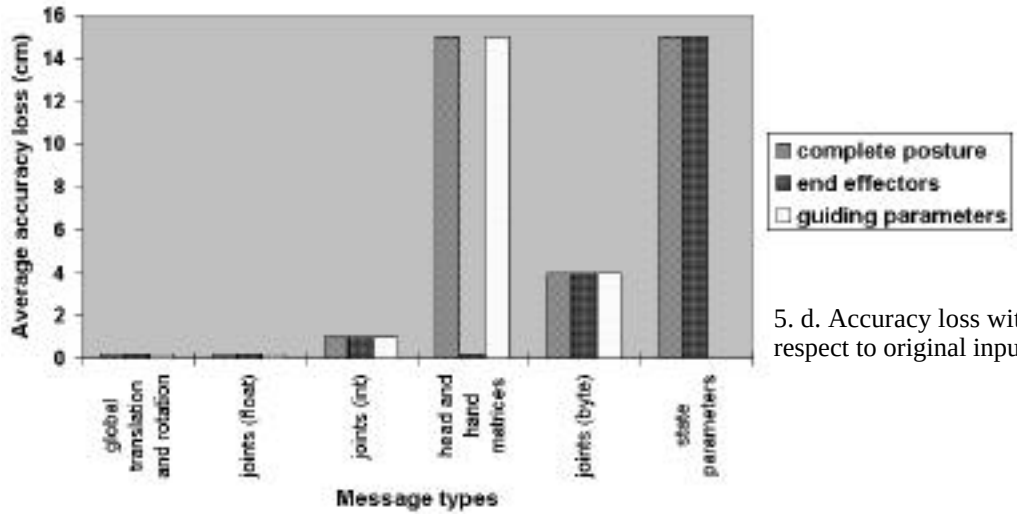
5. b. Coding computations for each body posture

Decoding computations per frame



5. c. Decoding computation for each body posture

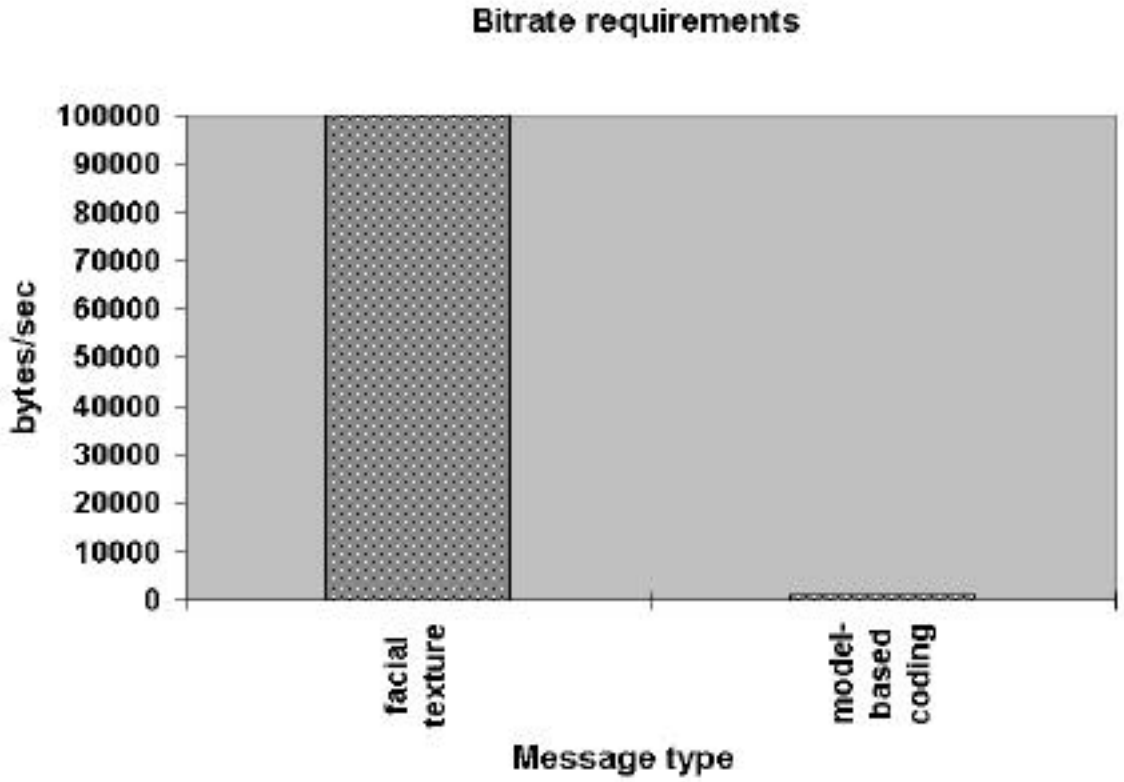
Accuracy loss



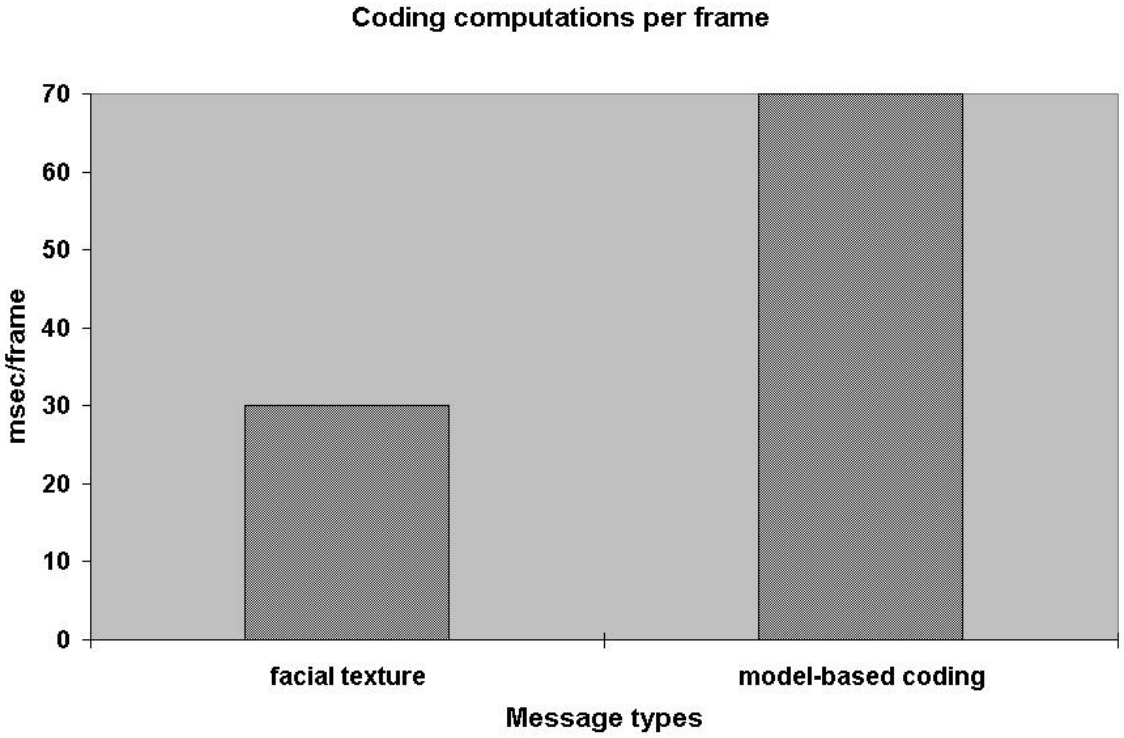
5. d. Accuracy loss with respect to original input data

Fig. 5: Networking body data

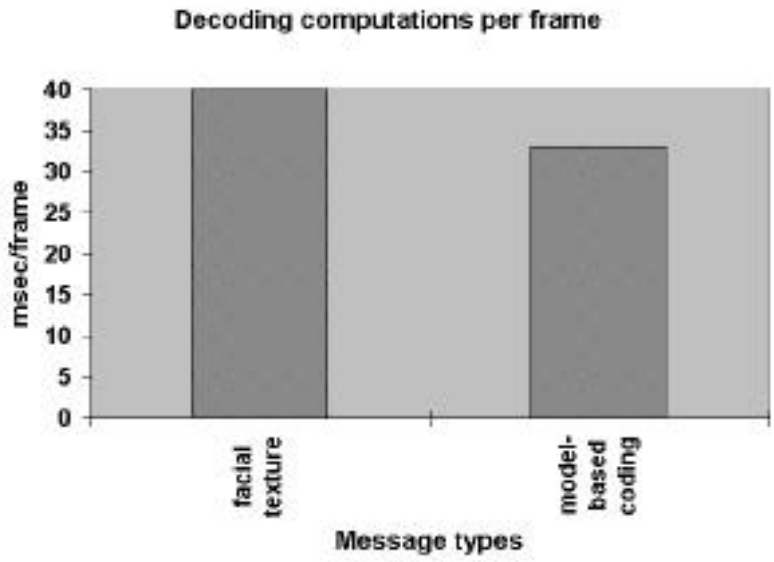
Similarly, Figure 6 shows the bandwidth requirements and experimental results for coding and decoding of the face. Figure 6a shows that using model-based coding drastically decreases the network overhead, while Figures 6b and 6c demonstrate that the overheads of coding and decoding are insignificant.



6. a. Bitrate requirements for each message type



6. b. Coding computations for face data



6. c. Decoding computations for face data
Fig. 6: Networking face data

As the number of participants increases, the transmission and decoding overheads will be excessive, and the speed might decrease significantly. Therefore, methods should be investigated to decrease this overhead. An approach is not to send the information to a site at all if there is no or little interaction, using filtering techniques [13]. In addition, the *dead-reckoning* techniques may be applied to extrapolate the human information from the last received information of body and face, and the last speed. The initial results on human body dead reckoning have shown that up to 50% of the traffic can be decreased by applying simple predictive filtering on the joint angles [14].

6 Virtual Tennis: an Example Application

As an example application, we selected a virtual tennis game with an autonomous virtual human player, and an autonomous referee. This application was chosen because it involves the critical issues discussed above: interaction of a real user with an autonomous virtual human over the network, real-time requirement for natural ball simulation, synthetic vision necessary for the autonomous virtual humans, multilevel control of the synthetic human figures.

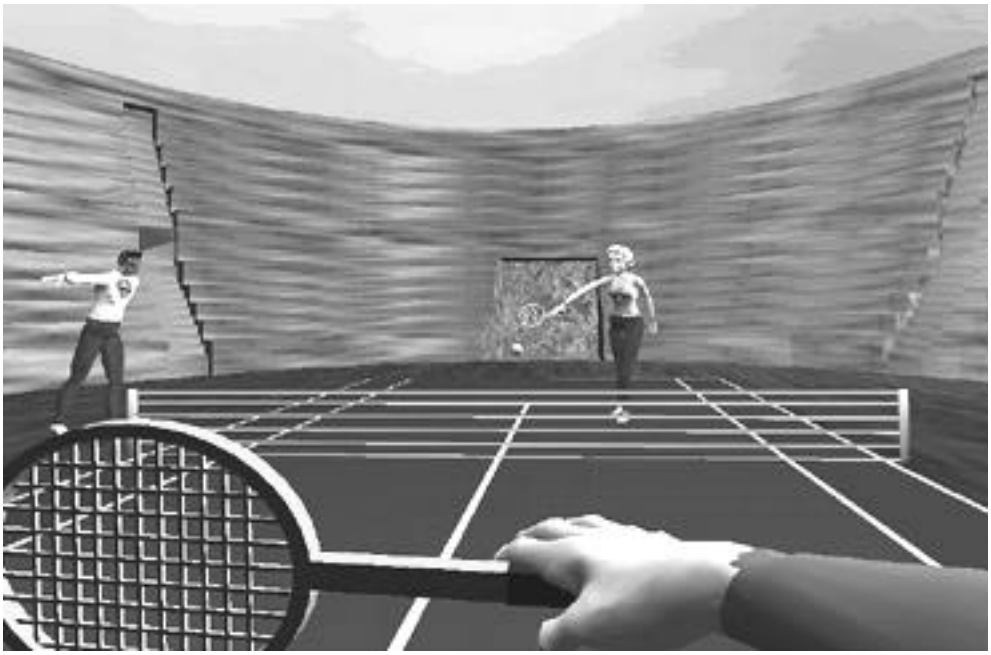


Fig. 7. A snapshot from the tennis game

In the next subsection we present some features of the example application with its autonomous virtual humans. Then we show more detailed how it was used together with the VLNET system to enable the networked interactive tennis game.

6.1 Application: L-system interpreter

We modeled a synthetic sensor based tennis match simulation for autonomous players and an autonomous referee, implemented in an L-system based animation system. The environment of the autonomous virtual humans is modeled and animated by L-systems which are timed production systems designed to model the development and behavior of static objects, plant like objects and autonomous creatures. They are based on timed, parameterized, stochastic, conditional environmentally sensitive and context dependent production systems, force fields, synthetic vision and audition. We published parts of this system in Noser et al. [15]. Prusinkiewicz and Lindenmayer [16] present the Lindenmayer systems - or L-systems for short- as a mathematical theory of plant development with a geometrical interpretation based on turtle geometry. The authors explain mathematical models of developmental processes and structures of plants and illustrate them with computer-generated images. Our behavioral L-system is based on the general theory about L-Grammars described in the above mentioned work.

An L-system is given by an axiom being a string of parametric and timed symbols, and some production rules specifying how to replace corresponding symbols in the axiom during the evolution of time. The L-system interpreter associates to its symbols basic geometric primitives, turtle control symbols or special control operations necessary for an animation. Basic geometric primitives are cubes, spheres, trunks, cylinders terminated at their ends by half spheres, line segments, pyramids and imported triangulated surfaces. We define the non generic environment as the ground, the tennis court, walls directly in the axiom of the production system. The generic parts as growing plants are defined by production rules having only their germ in the axiom. The virtual actors are also represented by a special symbol. Their geometric representation can vary from some simple primitives like some cubes and spheres, over a more complicated skeleton structure to a fully deformed triangulated body surface.

6.2 Behavior control

In the tennis game simulation the different behaviors of the autonomous virtual humans are modeled by automata controlled by a universal stack based control system. As the behaviors are severely based on synthetic sensors being the main channels of information capture from the virtual environment we obtain a natural behavior which is mostly independent of the internal environment representation. By using a sensor based concept, the distinction between an autonomous virtual human and an interactive user merged into the virtual world becomes small and they can easily be exchanged as demonstrated with the interactive game facility. The autonomous referee, represented by Virtual Elvis, judges the game by following the ball with his vision system. He updates the state of the match when he "hears" a ball collision event (ball - ground, ball - net, ball - racket) according to what he sees and his knowledge of a tennis match, and he communicates his decisions and the state of the game by "spoken words" (sound events). The autonomous player, represented by virtual Marilyn, can also hear sound events and obeys the decisions of the referee. Her game automata uses synthetic vision to localize the ball's and her opponent's position and adaptively estimates the future ball-racket impact point and position. She uses her partner's position to fix her game strategy and to plan her stroke and her path to the future impact point.

6.3 VLNET - L-system interface

The L-system interpreter shares with the participant client through VLNET clients and the VLNET server the important environment elements as the tennis court, the tennis ball, an autonomous referee, the autonomous player and the participant. Each virtual human has its own VLNET client, and the L-system interpreter program controls the referee and the autonomous player through the external interfaces of the VLNET system. The representation of the real participant in the L-system interpreter is reduced to a simple racket whose position is communicated through the network and obtained from the local VLNET client's object behavior interface at each frame. The racket and head positions and orientations of the autonomous virtual human player are communicated at each frame to the participant's VLNET client where they are mapped to an articulated, guided virtual human. This guided human is animated through inverse kinematics using the racket position. The referee is also represented by a guided virtual human in the user client getting his position at each frame from the L-system animation process.

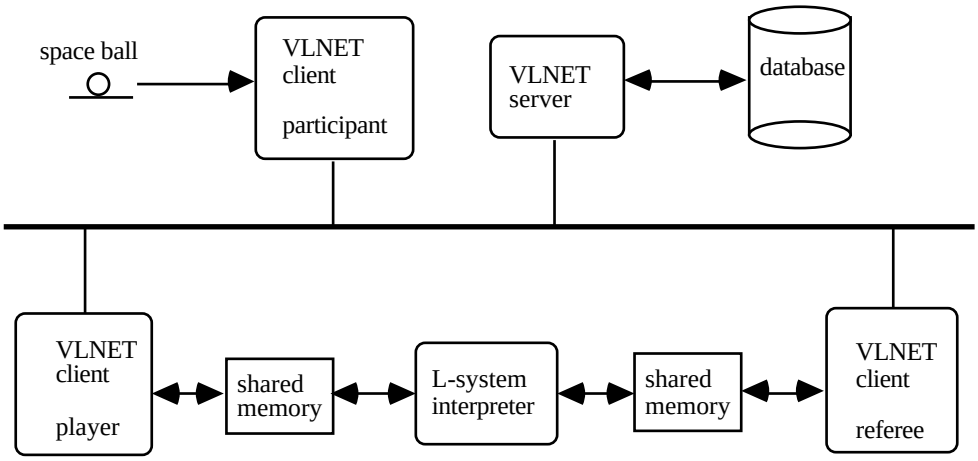


Fig. 8. The process configuration for the interactive tennis game.

The ball movement is modeled according to physical laws in the animation system. It is represented as a particle in a force field based particle system. The force fields of the ground, the net, the tennis rackets and the gravitation affect the balls movement. Its position is input to the VLNET client by the external object behavior interface.

The described architecture proved to be effective for the interactive tennis application. The transmission requirements for the virtual humans and the ball movements seemed not to be excessive, especially with local area network. However, initial results show that remote tests over busy Internet might cause delay that makes the game difficult to play. However, our initial tests over the ATM network could achieve real-time speed for playing over remote sites.

7 Conclusion

The human figure representation in networked virtual environments is not an easy task. First, we presented an easy architecture how they can be included in a complex NVE system. Next, we showed the different levels of controlling the human body, and compared them. Then, we presented different possibilities to send human information, and the load they put on the sender, network and receiver; as well as accuracy loss. The human figure information can put a load on the computational and networking resources, and the best control, representation and transmission form should be selected depending on the application and the resources.

The tennis application was an example to test different aspects of our architecture for including humans in networked virtual environments and combining two complex systems. The final speed of this integration could answer to the real-time requirement of the application.

Virtual human representation and communication in networked virtual environments is in an early stage. We will further our research on compression of virtual human models, and networking techniques to decrease communication requirements. The initial results are promising, and we hope to achieve very low bitrate virtual human communication.

Moreover, we will work on human motion control for direct, guided and autonomous virtual humans control. The new motion combination algorithm will allow different external processes to update the same body limbs, while satisfying the constraints imposed by the tracking information.

Acknowledgements

This work is partially supported by European TEN-IBC VISINET project, which allowed us to test the VLNET system over the ATM network between Switzerland, Belgium and UK . The development was also partly sponsored by the SPP program and the Federal Office for Education and Science in the framework of the European ACTS Project COVEN.

We are grateful to Patrick Keller for his remarkable help in designing the VLNET scenes and producing images. We would also like to thank Mireille Clavien for some of the designs, Eric Chauvineau for deformable body implementation, Ronan Boulic for his walking model. The ATM tests would be impossible without the VISINET project partners, especially Wim Lamotte and Nic Chilton. We also would like to thank National University of Singapore, Institute of Systems Sciences, for their help in ATM demonstration at TELECOM'95 exhibition between Singapore and Geneva.

References:

[1] M. Slater, M. Usoh, "Body Centered Interaction in Immersive Virtual Environments", *in Artificial Life and Virtual Reality*, N. Magnenat Thalmann, D. Thalmann, eds., John Wiley and Sons, 1994, pp 125-147.

[2] Rich Gossweiler, Robert J. Laferriere, Michael L. Keller, Pausch, "An Introductory Tutorial for Developing Multiuser Virtual Environments", *Presence: Teleoperators and Virtual Environments*, MIT Press, Vol. 3, No. 4, 1994. pp. 255-264.

[3] M. R. Stytz, "Distributed Virtual Environments", *IEEE Computer Graphics and Applications*, Vol. 16, No. 3, 1996, pp 19-31.

[4] R. Boulic et al. "The HUMANOID environment for Interactive Animation of Multiple Deformable Human Characters", *Computer Graphics Forum (Proceedings of Eurographics'95)*, Blackwell Publishers, Oxford, Vol. 14, No.3, 1995. pp. 337-348.

[5] N. I. Badler, C. B. Phillips, B. L. Webber, *Simulating Humans: Computer Graphics Animation and Control*, Oxford University Press, New York, 1993.

[6] P. Kalra, A. Mangili, N. Magnenat Thalmann, D. Thalmann, "Simulation of Facial Muscle Actions Based on Rational Free Form Deformations", *Computer Graphics Forum (Proceedings of Eurographics'92)*, Blackwell Publishers, Oxford, Vol. 11, No. 3, 1992, pp.59-69.

[7] D. Thalmann, "Using Virtual Reality Techniques in the Animation Process", in *Virtual Reality Systems*, R. A. Earnshaw, M. A. Gigante, H. Jones, ed., Academic Press, London, 1993, pp 143-160.

[8] T. Molet, R. Boulic, D. Thalmann, "A Real-Time Anatomical Converter for Human Motion Capture", *Proc. Eurographics Workshop on Computer Animation and Simulation*, R. Boulic ed., Springer, Wien, 1996, pp.79-94.

[9] S. K. Semwal, R. Hightower, S. Stansfield, "Closed Form and Geometric Algorithms for Real-Time Control of an Avatar", *Proc. IEEE VRAIS'96*, IEEE Computer Society Press, 1996, pp. 177-184.

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

[11] L. Emering, R. Boulic, S. Balcisoy, D. Thalmann, "Real-Time Interactions with Virtual Agents by Human Action Identification", *Proc. ACM Conference on Autonomous Agents'97*, ACM Press, 1997.

[12] F.Lavagetto, "Converting Speech into Lip Movements: A Multimedia Telephone for Hard of Hearing People", *IEEE Trans. on Rehabilitation Engineering*, Vol.3, No.1, 1995, pp.90-102.

[13] T. A. Funkhouser, "Network Topologies for Scalable Multi-User Virtual Environments", *Proc. VRAIS'96*, IEEE Computer Society Press, 1996, pp.222-229.

[14] Capin et al., "A Dead-Reckoning Algorithm for Virtual Human Figures", *Proc. VRAIS'97*, IEEE Computer Society Press, 1997.

[15] Noser et al., "Playing Games through the Virtual Life Network", *Proc. Artificial Life'96*, Chiba, Japan, 1996, pp.114-121.

[16] P. Prusinkiewicz, A. Lindenmaer, *The Algorithmic Beauty of Plants*, Springer-Verlag, Wien, 1990.