Virtual Human Representation and Communication in VLNet

Tolga K. Capin, Hansrudi Noser, and Daniel Thalmann
Swiss Federal Institute of Technology

Igor Sunday Pandzic and Nadia Magnenat Thalmann
University of Geneva

Using virtual humans to represent participants promotes realism in networked VEs. Different message types used to animate the human body and face impose varying network requirements, as analyzed here.

The fast pace in computing, graphics, and networking technologies plus the demands of real-life applications have impelled the development of more realistic virtual environments (VEs). Realism depends on a believable appearance and simulation of the virtual world and also implies a natural representation of participants. This representation includes

- a visual embodiment of the user,
- a means of interacting with the world, and
- a means of feeling various attributes of the world using the senses.

Realism in participant representation involves two elements: believable appearance and realistic movements. This becomes even more important in multiuser networked virtual environments (NVE), since the participants' representations are used for communication.

We can define an NVE as a single environment shared by multiple participants connected from different hosts. The participants' local program typically stores the whole or a subset of the scene description, and they use their own avatars to move around the scene. Rendering takes place from their own viewpoints. 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 functions realistically, providing a direct relationship between how we control our avatars in the virtual world and how our avatars move in response to this control. Even with limited sensor information, we can construct a virtual human frame that reflects the user's activities in the virtual world. Slater and Usoh[1] indicated that using even a simple virtual body increases the sense of presence in the virtual world. (See the sidebar "Definitions and Concepts" for explanations of the terms used in this article.)

NVEs with virtual humans are 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. The practice is to bring together different, previously developed monolithic applications within one standard interface, building multiple logical or actual processes to handle separate elements of the VE. Second, at the same time, virtual human research has developed to the point of providing 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 include

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

0272-1716/97/$10.00 © 1997 IEEE

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

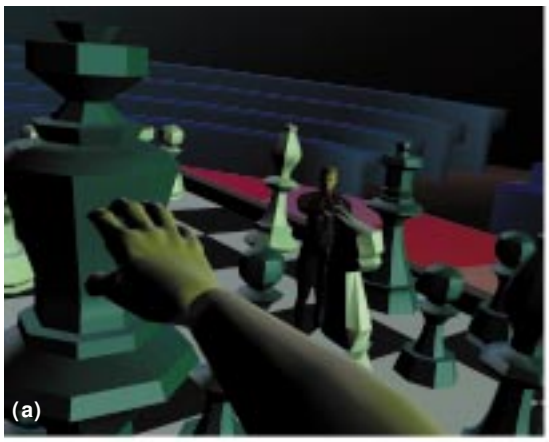
Multiprocess client architecture

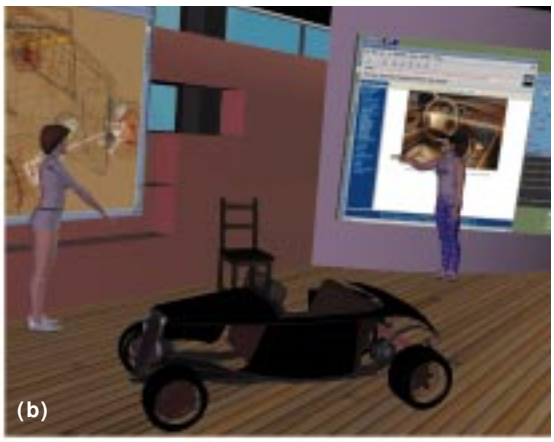
Typically, the sheer complexity of VE simulation systems imposes a modular design on the software. It is appropriate to design the runtime system as a collection of cooperating processes, each responsible for its particular task. This also allows easier portability and better performance of the overall system through the decoupling of different tasks and their execution over a network of workstations or different processors on a multiprocessor machine. We use this multiprocess approach in VLNet. Figure 2 (on the next page) shows the architecture of the VLNet client, with separation of 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 VEs' basic elements to the external programs, called drivers. The display, cull, and database processes—standard Iris

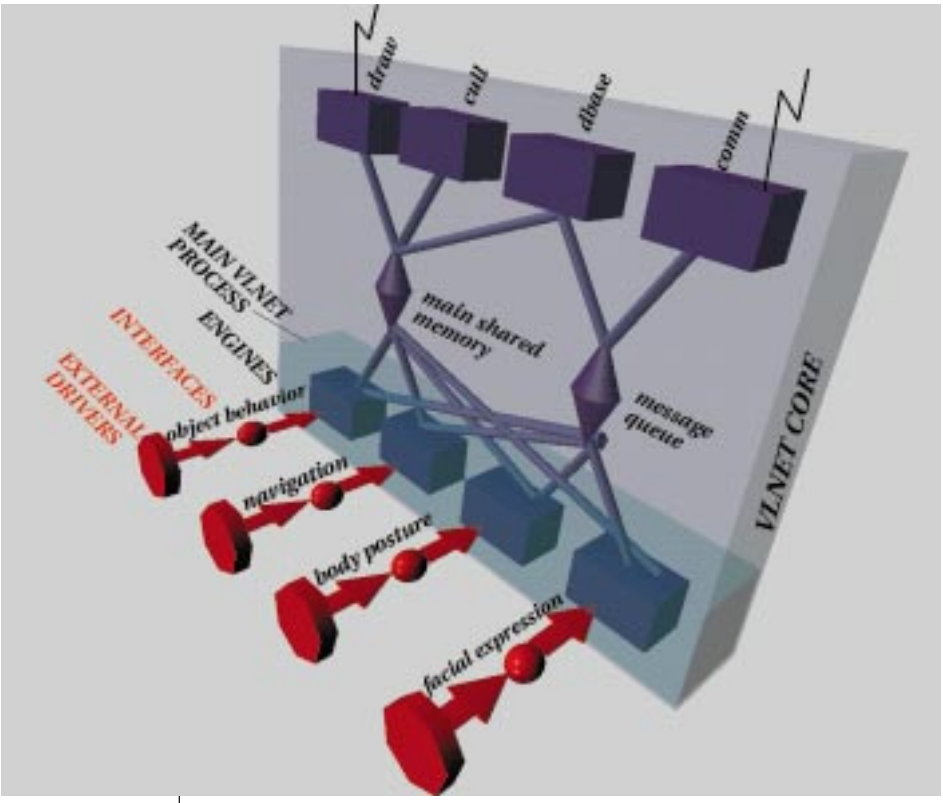
Definitions and Concepts

Agent	A software process that has autonomous behavior. It does not necessarily have to be represented by a graphics entity.
Autonomy	Quality or state of being self-governing.
Avatar	A representation of a real person in a networked virtual environment
Direct controlled virtual human	A virtual human whose body and face geometric representation is directly updated by the user or program controlling it.
User-guided virtual human	A virtual human whose control is in the form of tasks to perform (for example, go to a location, sit) and which uses its motor skills to perform this action by coordinated joint movements.
Participant	A real person who participates in the networked virtual environment and is represented by an avatar.
Virtual human	A graphics entity that is totally represented by computer and looks like a real human.





1 Example applications of networked virtual environments: (a) networked games, (b) virtual meetings, (c) teleshopping, and (d) medical education.



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

Performer processes—allow asynchronous loading and display of the scene with the main simulation.

The main process consists of four logical units, called *engines*. The engine separates one main function in the VE to an independent module and provides an orderly, controlled allocation of VE elements. Moreover, the engine manages this resource among various programs competing for the same object. The communication process receives and sends messages through the network, and uses incoming and outgoing message queues to implement asynchronous communication.

The object-behavior engine handles 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 navigation, object selection, and manipulation. The input consists of relative and absolute matrices of the body's global position and requests for selecting or releasing an object.

Similarly, the face and body representation engines are specialized for the virtual human figure. The face engine connects VLNet and external face drivers. It obtains the camera video images or face model parameters from the external face driver and places them in VLNet's internal shared memory and outgoing message queue.

The body representation engine has an external interface for body posture, including joint angles or global positioning parameters, and high-level parameters to animate the body. This engine permits defining multiple levels of control for the human body and merging the output of different external body drivers into a single final posture.

Different data dependencies exist between the external face and body process and other components of the environment simulation system. The virtual humans can have behaviors, which means they must have a method for coding and using the environment's effects on their virtual bodies. The virtual human should be equipped with visual, tactile, and auditory sensors. These sensors serve 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 act on the environment, for example by grasping and repositioning an object.

Next we discuss different human-figure motion-control methods for creating complex motion. This requires sharing information between the object and navigation engines, and external human processes using their external interfaces.

Virtual human modeling

Typically, control of body posture requires an articulated structure corresponding to the human skeleton. Structures representing body shape must be attached to the skeleton, and clothes might be wrapped around the body shape.

In VLNet, we use an articulated human body model with 75 degrees of freedom, plus an additional 30 degrees of freedom for each hand. The skeleton is represented by a 3D articulated hierarchy of joints, each with realistic maximum and minimum limits. The skeleton is encapsulated with geometrical, topological, and inertial characteristics for different limbs. The body structure has a fixed topology template for joints. Different body instances are created by scaling limbs globally as well as by applying frontal, high, and low lateral scaling or specifying spine origin ratio between lower and upper body parts.[2]

A second layer attached to the skeleton consists of blobs (metaballs) to represent muscle and skin. The main advantage lies in permitting us to cover the entire human body with a small number of blobs.

From this point we divide the body into 17 parts: head, neck, upper torso, lower torso, hip, and left and right upper arm, lower arm, hand, upper leg, lower leg, and foot. Because of their complexity, head, hands, and feet are represented with triangle meshes instead of blobs. The other parts use a cross-sectional table for deformation. This 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. The skin contour is attached to the skeleton during runtime and at each step is interpolated around the link depending on the joint angles. The deformation component creates the new

body triangle mesh from this interpolated skin contour.

Different parameter sets exist for defining virtual human postures and faces:

- *Global positioning domain parameters.* These parameters consist of the global position and orientation values of particular observable points on the body (top of head, back of neck, mid-clavicle, shoulders, elbow, wrist, hip, knee, ankle, bottom of mid-toe) in the body coordinate system.
- *Joint-angle domain parameters.* This category includes the joint angles defined above that connect different body parts.
- *Hand and finger parameters.* The hand can perform complicated motions and contains at least 15 joints, not counting the carpal part. Since 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 as a polygon mesh model with defined regions and free-form deformations modeling the muscle actions.[3] It can be controlled on several levels. On the lowest level, an extensive set of 65 minimal perceptible actions (MPAs), closely related to muscle actions, can be controlled directly. They can describe the facial expression completely. Each MPA is a basic building block facial motion parameter that allows moving a separate visual facial feature (such as raising an eyebrow or closing the eyes). On a higher level, phonemes and/or facial expressions can be controlled spatially and temporally. On the highest level, complete animation scripts can be input, defining speech and emotion over time. Algorithms exist to map texture on such a facial model.

Virtual human control

The participant should be able to animate his virtual human representation in real time, but such control is not straightforward: The complexity of virtual human representation demands tracking of many degrees of freedom. Interaction with the environment increases this difficulty even more. Therefore, virtual human control should use higher level mechanisms to animate the representation with maximal facility and minimal input. We can divide virtual humans according to the methods used to control them:

- *Directly controlled virtual humans.* The joint and face representation of the virtual human is modified directly (for example, 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. The virtual human uses its motor skills to perform actions through coordinated joint movements (for example, walking or sitting).
- *Autonomous virtual humans.* The virtual human is assumed to have an internal state built from its goals and sensor information. The participant modifies this state by defining high-level motivations and state changes (such as turning on vision behavior).

Directly controlled virtual humans

The virtual human must have a natural-looking body and be animated with respect to the participant's actual body. This corresponds to a real-time form of traditional rotoscopy.

In animation, traditional rotoscopy consists of recording motion with a specific device for each frame and using this information to generate the image by computer. The real-time rotoscopy method consists of recording input data from a VR device in real time while applying the same data at the same time to a graphics object on the screen. For example, when the animator opens her fingers 3 centimeters, the hand in the virtual scene opens exactly 3 centimeters. 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.

For more immersive interaction, a complete representation of the participant's virtual body should have the same movements as the real participant's body. You can best achieve this by using many sensors to track every degree of freedom in the real body. Molet et al.[4] wrote that a minimum of 14 sensors are required to manage a biomechanically correct posture, and Semwal et al.[5] presented a closed-form algorithm to approximate the body using up to 10 sensors.

Many 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 not tracked. This is more than a simple inverse kinematics problem, because you will generally find multiple solutions for the joint angles to reach the same position and should select the most realistic posture. The joint constraints also should be considered in setting the joint angles.

The main lowest-level approaches to this extrapolation problem are inverse kinematics using constraints,[6] closed-form solutions,[5] and table lookup solutions.[7] The inverse kinematics approach is based on an iterative algorithm, where an end-effector coordinate frame (the hand, for example) tries to reach a goal (the reach position) coordinate frame using a set of joints that control the end effector. This approach has the advantage 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 the simulation significantly, as it requires excessive computation.

The closed-form solution eliminates this problem by attaching 10 sensors 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 to determine the angle between the sensor coordinate frames. However, this approach still needs 10 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 are 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 uses 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 a 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.

Guided virtual humans

Guided virtual humans are driven by the user but do not correspond directly to user motion. They build on the concept of the *real-time direct metaphor*, a method consisting of recording input data from a VR device in real time. This method lets us produce effects of different natures but corresponding to the input data. No analysis takes place as to the real meaning of the input data.

To understand the concept, consider an example of traditional metaphor: puppet control. A puppet may be defined as a doll with jointed limbs moved by wires or strings. Glove-puppets are dolls whose body fits over your hand like a glove, with the arms and head being moved by your fingers. In both cases, human fingers drive the puppet's motion.

In VLNet, an example of virtual human guidance is guided navigation. The participant uses the input devices to update the transformation of the virtual human's eye position. This local control results from 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, from which it computes the joint angles for the whole body. The sensor information on walking can be obtained from various types of input devices, such as gesture with a DataGlove or SpaceBall, as well as other input methods.

Autonomous virtual humans

An autonomous system can give itself its proper laws, its conduct, as opposed to a heteronomous system, which is driven from the outside. Guided virtual humans are typically driven from the outside. Including autonomous virtual humans that interact with participants increases the real-time interaction with the environment and will likely increase the sense of presence for the real participants.

The autonomous virtual humans connect to the VLNet system in the same way as human participants. They also improve use of the environment by providing services such as replacing missing partners and helping in navigation. Since these virtual humans are not guided by users, they should have sufficient behaviors to act autonomously in accomplishing their tasks. This requires building behaviors for motion and appropriate mechanisms for interaction.

To have behaviors, our autonomous virtual humans must have a manner of conducting themselves. Behavior is not just reacting to the environment; it also includes the flow of information by which the environment acts on the living creature and how the creature codes and uses this information. Behavior of autonomous virtual humans arises from their perception of the environment.

Combining motions

The different motion generators output their results as new joint angles between connecting limbs. As discussed, external driver programs can be attached to the human driver engine. Normally, more than one external driver can connect to the human posture interface, and the engine resolves 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.

Combining motions requires that the human posture interface contain 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 its output on the final posture for this range.

For our initial implementation, we divided the virtual body hierarchy into eight parts: torso, neck and head, and left and right arms, legs (including feet), and hands. Only one driver can modify each part.

To control a body part, the external driver should lock the part by storing a lock identifier in the interface. At each time frame the external processes update the body parts that they have locked. This approach prevents us from using multiple processes to update the same body part and therefore is limiting. Our current development incorporates a motion combination algorithm based on weights and priorities of different external processes' output.[8]

It becomes more complicated to combine different motions if some of the external drivers contain goal-directed motion, since the final posture should satisfy the condition of reaching the goal. 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 the same as the actual hand's tracked position. Therefore, the motion combination should consider correcting the directly updated posture to minimize the difference between the goal position and the effector.

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.

Video texturing of the face

This approach continuously texture maps the video sequence of the user's face onto the face of the virtual human. The user must be in front of the camera, in such a position that the camera captures her head and shoulders and possibly the rest of her body.

A simple, fast image-analysis algorithm finds the bounding box of the user's face within the image. The algorithm requires the head-and-shoulder view and a static (though not necessarily uniform) background. Thus the algorithm primarily compares each image with

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

the background's original image. Since the background is static, any change in the image results from the user's presence, making it fairly easy to detect her position. This permits the user reasonably free movement in front of the camera without loss of the facial image.

A special facial expression driver performs the video capture and analysis. The driver compresses each facial image in the video sequence using Silicon Graphics' Compression Library and passes the compressed images to VLNet's facial representation engine, which then redirects them to the communication process. The communication process sends these images over the network to remote participants. At the receiving sites, the communication processes receive these images and pass them to their facial representation engines. The facial representation engines decompress and texture map the images onto the face of the virtual human representing the user.

Currently, for texture mapping we use a simple frontal projection, a simplified 3D head model with attenuated features. This permits less precise texture mapping. If we used a head model with all the facial features, any misalignment of the topological features in the 3D model and the features in the texture would produce visibly unnatural artifacts. To avoid this, we need the coordinates of characteristic feature points in the image. We can use these coordinates to calculate the texture coordinates so as to align the features in the image with the topology in a process called texture fitting. However, our current texture-fitting algorithm does not work in real time.

Figure 3 illustrates video texturing of the face. It shows the original images of the user and the corresponding images of the virtual human representation.

Model-based coding of facial expressions

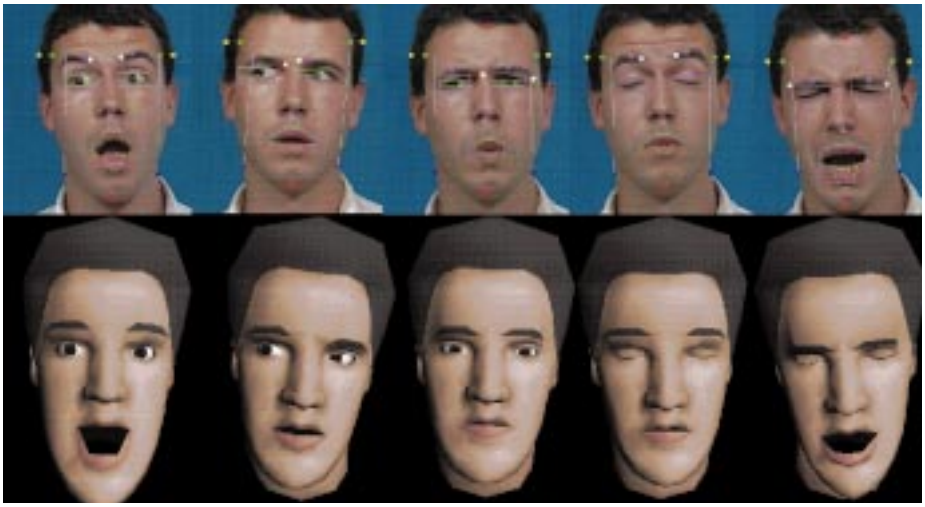
Instead of transmitting whole facial images as in the previous approach, this approach analyzes the images and extracts a set of parameters describing the facial expression. As in the previous approach, the user has to be in front of the camera that is digitizing the head-and-shoulders video images.

Accurate recognition and analysis of facial expressions from video sequences requires detailed measurements of facial features. Currently, it is computationally expensive to perform these measurements precisely. Since our primary concern has been to extract the features in real time, we have focused on recognition and analysis of only a few facial features. Our recognition method relies on the "soft mask"—a set of points on the image of the face adjusted interactively by the user. Various characteristic measures of the face are calculated at the time of initialization. Color samples of the skin, background, hair, and so forth are also registered. Recognition of the facial features relies primarily on color-sample identification and edge detection.

We use variations of these methods based on human facial characteristics to find the optimal adaptation for a particular case of each facial feature. Special care is taken to make the recognition of one frame independent of the recognition of the previous one, thus avoiding the accumulation of error. The data extracted from the previous frame is used only for the features that are relatively easy to track (such as the edges of the neck). 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

A special facial expression driver performs the analysis in our implementation. The extracted parameters are easily translated into minimal perceptible actions, which



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

are passed to the facial representation engine, then to the communication process, where they are packed into a standard VLNet message packet and transmitted. 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.

Lip-movement synthesis from speech

The user might not always find it practical to be in front of a camera (for example, if he doesn't have one, or if he wants to use an HMD). Nevertheless, we don't need to abandon facial communication. Lavagetto[9] showed that analyzing the audio signal of speech makes extracting the visual parameters of lip movement possible.

An application doing such recognition and generating MPAs for controlling the face can be connected to the VE program through the facial expression driver. The facial representation engine can then synthesize the face with the appropriate lip movements. A primitive version of such a system would just open and close the mouth during speech, letting participants know who is speaking. A more sophisticated system would actually synthesize realistic lip movement, which is an important aid for understanding speech.

Predefined expressions or animations

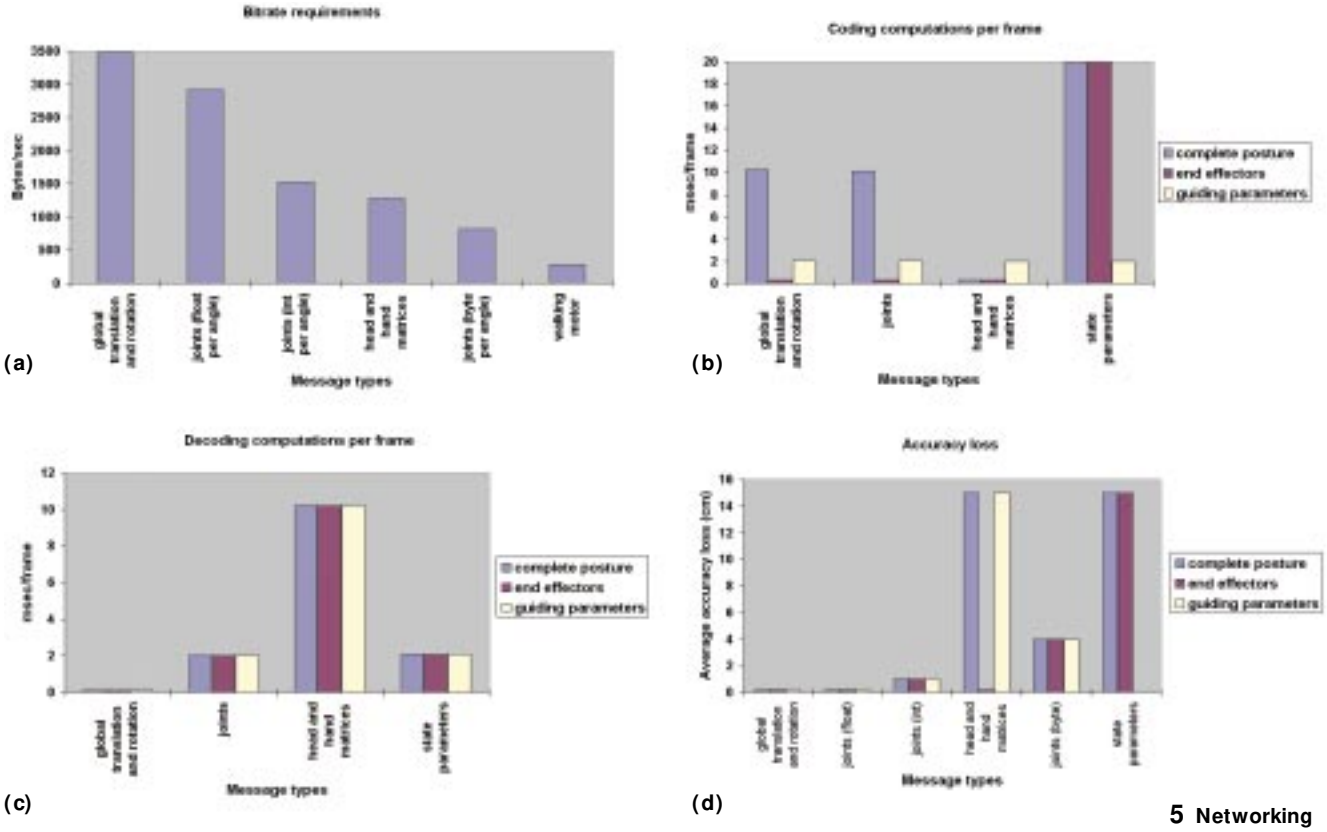
In this approach the user can simply choose between a set of predefined facial expressions or movements (animations). The choice can be done from the keyboard through a set of "smileys" similar to the ones used in e-mail messages. The facial expression driver in this case stores a set of defined expressions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

Networking

The face and articulated human body introduce a new complexity in the use of network resources because the size of a message needed to convey body posture exceeds that needed for simple, unarticulated objects. This might create significant overhead in communication, especially as the number of participants in the simulation increases. To reduce this overhead you could communicate the postures in more compact forms, accepting some loss of accuracy in the posture definition. However, this is not the only trade-off to consider 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 network analysis, we separate the discussion into body and face, as they use different control methods and 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 equal the sum of the lag of all these phases. In addition, each message type contains an accuracy loss of data—a trade-off to decrease the lag. In this section, we analyze different message types with respect to the following:

- *Coding computation at the sender site.* Evaluate the amount of computation needed to convert the input data into the message to be sent, at the sending site.
- *Bit-rate requirements.* Evaluate the bandwidth requirements for different parameters to describe the motion. We assume a minimum limit for real-time computation of 10 frames per second.
- *Decoding computation at the receiver site.* 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 typically exceeds that at the sender site because—contrary to coding, which is done only for the locally controlled figure—the messages from a potentially large number of participants have to be processed.



5 Networking body data. (a) Bit-rate requirements for each message type. (b) Coding computations for each body posture. (c) Decoding computations for each body posture. (d) Accuracy loss with respect to original input data.

- *Accuracy loss.* Evaluate the loss of accuracy of the body posture with respect to the original input data. This is typically the trade-off to consider against decreasing coding and decoding computations and transmission overhead.

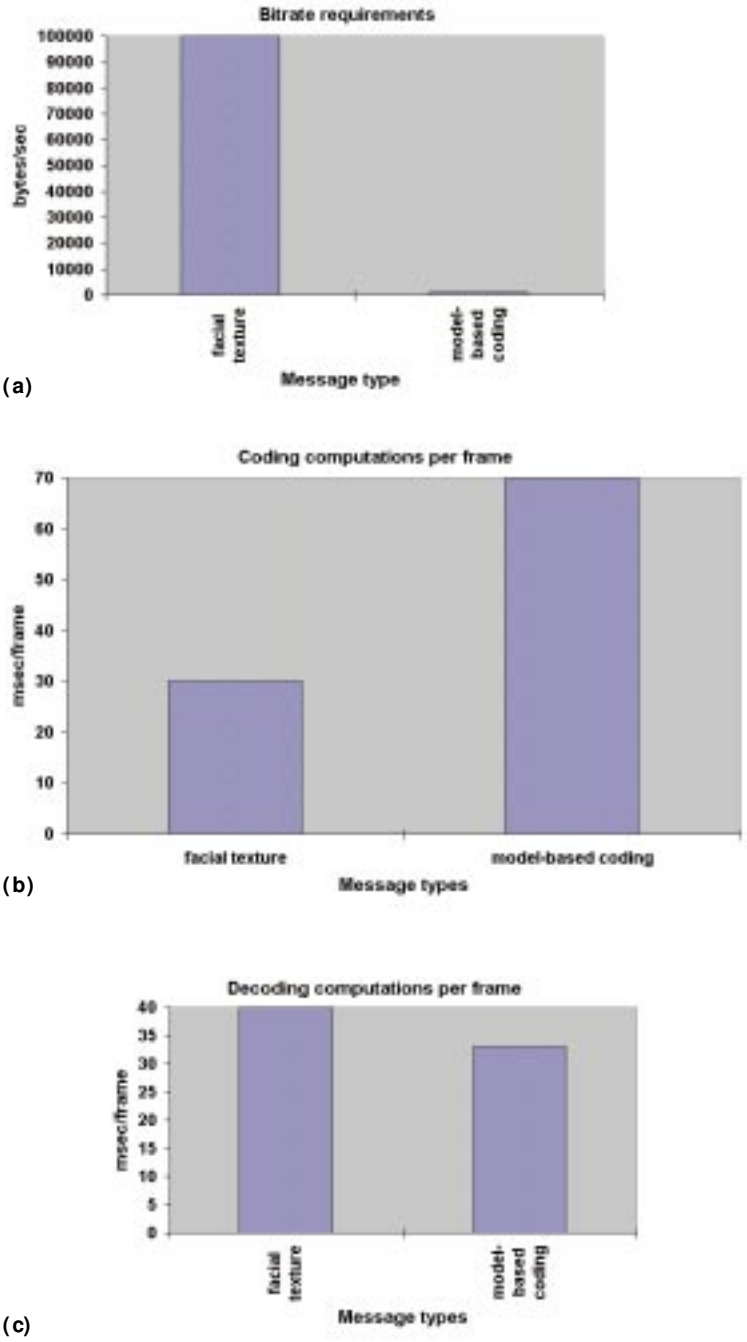
We compare these issues for various types of human-body motion control. We consider four types of message packets to convey the body posture information:

- *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, bypassing 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 4×4 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 (for predefined static postures like sitting or standing) through medium (for predefined dynamic states like walking or running) to very complex (for more complex dynamic states like searching for 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: complete body posture data is available, only head and hand end-effector data is available, and walking motion guiding data is available from an external driver.

Figure 5 shows the bit-rate requirements, coding and decoding computations per frame, and accuracy loss for different message types. Figure 5a is calculated assuming a minimum real-time speed of 10 frames per second. We see that the bit-rate requirement varies between 4 Kbytes and 240 bytes per second for one human body. Figures 5b and 5c show the coding and decoding results on an SGI Indigo2 Impact workstation with a 250-MHz processor. The results show 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 depends on the particular application's requirements. Applications needing high accuracy (in medical training, for example) will require the transfer of body part matrices or at least end-effector matrices. In a large-scale simulation with numerous users we might achieve efficiency by conveying small messages containing the state information and reduce the computational overhead by using filtering and level of detail techniques.



6 Networking face data. (a) Bit-rate requirements for each message type. (b) Coding computations for face data. (c) Decoding computations for face data.

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. This transfer balances the computational overhead of the network, compression/decompression (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 but 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. We computed the results 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.

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.

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 (see Figure 7). This application involves the critical issues discussed above: interaction of a real user with an autonomous virtual human over the network, a real-time requirement for natural ball simulation, synthetic vision for the autonomous virtual humans, and multilevel control of the synthetic human figures.

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. L-systems 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.[12] Prusinkiewicz and Lindenmayer[13] presented the Lindenmayer systems (L-systems for short) as a mathematical theory of plant development with a geometrical interpretation based on turtle geometry. The authors explained mathematical models of developmental processes and structures of plants and illustrated them with computer-generated images. Our behavioral L-system builds on the general theory about L-grammars described in their work.

An L-system is given by an axiom consisting of a string of parametric and timed symbols plus 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 animation. Basic geometric primitives include cubes, spheres, trunks, cylinders terminated at their ends by half spheres, line segments, pyramids, and imported triangulated surfaces.

We define the nongeneric environment—the ground, the tennis court, and the walls—directly in the production system's axiom. The generic parts, such 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 simple primitives like cubes and spheres to a more complicated skeletal structure to a fully deformed triangulated body surface.

Behavior control

In the tennis game simulation, automata controlled

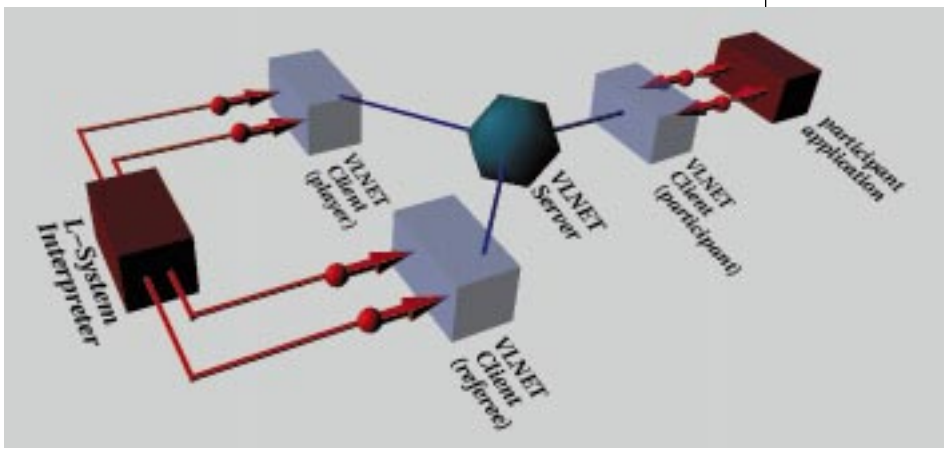
7 A snapshot from the virtual tennis game.

by a universal stack-based control system model the different behaviors of the autonomous virtual humans. Because the behaviors are strictly based on synthetic sensors—the main channels of information capture from the virtual environment—we obtain natural behavior mostly independent of the internal environment representation. Using a sensor-based concept shrinks the distinction between an autonomous virtual human and an interactive user merged into the virtual world. They can easily be exchanged, as the interactive game facility demonstrates.

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. 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 referee's decisions. Her game automata uses synthetic vision to localize the ball's and her opponent's positions, 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.

VLNet-L-system interface

The L-system interpreter shares with the participant client—through VLNet clients and the VLNet server—the important environment elements: 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. Figure 8 shows the process configuration for the game. The real participant's representation in the L-system interpreter reduces 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



8 The process configuration for the interactive tennis game.

Virtual Humans in MPEG-4

MPEG-4 addresses networked multimedia applications such as video telephony, networked virtual environments and games, distance learning, remote presentation, and other applications that involve interaction, transmission, and combination of natural audio and video streams with 2D and 3D computer graphics models. Targets of standardization include mesh-segmented video coding, compression of geometry, synchronization among audiovisual objects, multiplexing of streamed objects, and spatial-temporal integration of mixed media types.

A subgroup within MPEG-4 called SNHC (Synthetic Natural Hybrid Coding) works on efficient coding of graphics models and compressed transmission of their animation parameters specific to the model type.[1] Within this work, the group proposed a set of parameters for standard representation of the human body and the face. An initial document details the specification for face and body definition and animation parameters.[2] Definition parameters allow detailed definition of body and face shape, size, and texture. Animation parameters permit defining facial expressions and body postures. The parameters are designed to cover all (naturally) possible expressions and postures, as well as exaggerated expressions and postures (to some extent) providing cartoon-like animation. The standardization process continues, and the international standard is expected to be finished by November 1998.

Visit the MPEG-4 SNHC home page (http://www.es.com/mpeg4-snhc) for more information, to join MPEG-4 SNHC, and for details on these parameters.

References

1. P.K. Doenges et al., "MPEG-4: Audio/Video and Synthetic Graphics/Audio for Mixed Media," *Image Comm. J,* Elsevier, Amsterdam, 1997 (to appear).
2. SNHC Face/Body Ad Hoc Group, "Face and Body Definition and Animation Parameters," Document No. MPEG96/N1365, Oct. 1996, Chicago Meeting of ISO/IEC JTC1/SC29/WG11.

virtual human in the user client, getting his position at each frame from the L-system animation process.

The ball movement, modeled according to physical laws in the animation system, is represented as a particle in a force-field-based particle system. The force fields of the ground, the net, the tennis rackets, and gravity affect the ball's movement. The external object behavior interface sends its position to the VLNet client.

The architecture described here proved effective for our interactive tennis application. The transmission requirements for the virtual humans and the ball movements seemed not to be excessive, especially with a local area network. However, initial results show that remote tests over the busy Internet might cause delays, making the game difficult to play. On the other hand, our initial tests over an ATM network indicate that our system might well achieve real-time speed for play over the network at remote sites once we have optimized communication for the ATM configuration.

Conclusion

Representing the human figure in networked virtual environments is not easy. Moreover, the human figure information can put a heavy load on the computational and networking resources. You should select the optimal control, representation, and transmission forms demanded by the application and permitted by the resources available.

The tennis application allowed us to test different aspects of our architecture, specifically for including humans in networked virtual environments and for combining two complex systems. The final speed of this integration could conceivably meet the application's real-time requirement.

Virtual human representation and communication in networked virtual environments remains in an early stage. We plan further research on compression of virtual human models, as well as networking techniques to decrease communication requirements. The initial results look promising, and we hope to achieve very low bit-rate virtual human communication.

We also intend to work on human motion control for direct, guided, and autonomous virtual humans. Our new motion combination algorithm will let different external processes update the same body limbs, while satisfying the constraints imposed by the tracking information. ■

Acknowledgments

This work is partially supported by the European TEN-IBC Visinet project, which allowed us to test the VLNet system over the ATM network between Switzerland, Belgium, and the 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, and 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 the National University of Singapore, Institute of Systems Sciences, for their help in an ATM demonstration at the Telecom 95 exhibition between Singapore and Geneva.

References

1. M. Slater and M. Usoh, "Body Centered Interaction in Immersive Virtual Environments," *Artificial Life and Virtual Reality*, N. Magnenat Thalmann and D. Thalmann, eds., John Wiley and Sons, Chichester, UK, 1994, pp. 125-147.
2. R. Boulic et al., "The Humanoid Environment for Interactive Animation of Multiple Deformable Human Characters," *Computer Graphics Forum* (Proc. Eurographics 95), Blackwell Publishers, Oxford, Vol. 14, No. 3, Sept. 1995, pp. 337-348.
3. P. Kalra et al., "Simulation of Facial Muscle Actions Based on Rational Free-Form Deformations," *Computer Graphics Forum* (Proc. of Eurographics 92), Vol. 11, No. 3, Sept. 1992, pp. 59-69.
4. T. Molet, R. Boulic, and D. Thalmann, "A Real-Time Anatomical Converter for Human Motion Capture," *Proc. Eurographics Workshop on Computer Animation and Simulation*,

R. Boulic, ed., Springer, Wien, Austria, 1996, pp. 79-94.
5. S.K. Semwal, R. Hightower, and S. Stansfield, "Closed Form and Geometric Algorithms for Real-Time Control of an Avatar," *Proc. IEEE VRAIS 96*, IEEE Computer Society Press, Los Alamitos, Calif., 1996, pp. 177-184.
6. N.I. Badler, C.B. Phillips, and B.L. Webber, *Simulating Humans: Computer Graphics Animation and Control*, Oxford University Press, New York, 1993.
7. T.K. Capin et al., "Virtual Humans for Representing Participants in Immersive Virtual Environments," *Proc. FIVE 95*, Queen Mary and Westfield College Press, London, Dec. 1995, pp. 135-150.
8. L. Emering et al., "Real-Time Interactions with Virtual Agents by Human Action Identification," *Proc. ACM Conf. on Autonomous Agents 97*, ACM Press, New York, scheduled for publication in Feb. 1997.
9. 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.
10. T.A. Funkhouser, "Network Topologies for Scalable Multi-User Virtual Environments," *Proc. VRAIS 96*, IEEE Computer Society Press, Los Alamitos, Calif., Apr. 1996, pp. 222-229.
11. Capin et al., "A Dead-Reckoning Algorithm for Virtual Human Figures," *Proc. VRAIS 97*, IEEE Computer Society Press, Los Alamitos, Calif., scheduled for publication in March 1997.
12. Noser et al., "Playing Games through the Virtual Life Network," *Proc. Artificial Life 96*, Chiba, Japan, 1996, pp. 114-121.
13. P. Prusinkiewicz and A. Lindenmaer, *The Algorithmic Beauty of Plants*, Springer-Verlag, Vienna, 1990.

Tolga K. Capin *is currently a PhD candidate in computer science at Computer Graphics Laboratory, Swiss Federal Institute of Technology in Lausanne (EPFL). His research interests include networked virtual environments, human representation and animation in virtual environments, and parallel processing for computer graphics. He obtained his MS and BS from Bilkent University, Ankara, Turkey in 1993 and 1991, respectively. He is a member of Virtual Humans Architecture Group and MPEG-4 SNHC Ad-Hoc Group on Face and Body Animation.*

Igor Sunday Pandzic *is a PhD student at Miralab, University of Geneva. His research interests include collaborative virtual environments, parallel computing, computer-generated film production, and real-time facial expression analysis and synthesis. He finished his undergraduate studies in electrical engineering at the University of Zagreb in 1993. In 1994 he obtained a Masters degree in computer graphics from Swiss Federal Institute of Technology at Lausanne (EPFL). He is a member of Virtual Humans Architecture Group and MPEG-4 SNHC Ad-Hoc Group on Face and Body Animation.*

Hansrudi Noser *is a PhD student and assistant at EPFL. His research interests include behavioral animation, sound rendering, and L-systems. He received his diploma in physics from the Swiss Federal Institute of Technology in Zurich (ETHZ). After working as a researcher for the Institute of Applied Physics at the ETHZ (C-MOS process developing) and as a researcher in the department of thin film electronics of the Balzers AG, he received a diploma in computer science from the Swiss Federal Institute of Technology in Lausanne, Switzerland.*

Nadia Magnenat Thalmann *founded Miralab, a Research Lab at the University of Geneva, in 1989. She is a pioneering researcher in the field of virtual humans and has authored or co-authored more than 200 scientific papers. She has received several awards, including the 1985 Communications Award from the Government of Quebec, the 1992 Moebius Award from the European Community, and the British Computer Science Award in 1993. She received a PhD in quantum physics and computer graphics from the University of Geneva.*

Daniel Thalmann *is professor and director of the Computer Graphics Laboratory at the Swiss Federal Institute of Technology in Lausanne. His research interests are in virtual humans, computer animation, networked virtual environments, and artificial life. He is co-editor-in-chief of the* Journal of Visualization and Computer Animation *and an editorial board member of* The Visual Computer *and the* CADDM Journal of the China Engineering Society*, among others. He has organized several courses at Siggraph and is co-chair of the Eurographics Working Group on Animation and Simulation. He received his diploma in nuclear physics and a PhD in computer science from the University of Geneva.*

Readers can contact Capin at Computer Graphics Laboratory (LIG), Swiss Federal Institute of Technology (EPFL), 1015 Lausanne, Switzerland, e-mail capin@lig.di.epfl.ch.