Virtual Human Representation and Communication in Networked Virtual Environments

Tolga K. Capin¹, Igor Sunday Pandzie², Hansrudi Noser¹
Nadia Magnenat Thalmann², Daniel Thalmann¹

¹Computer Graphics Laboratory(EPFL)
²MIRALAB, University of Geneva

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
Using virtual human figures for avatar representation fulfills these functionalities with the greatest 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 [Slater94] 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. ([Gossweiler94][DVE96] 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 processes, modeling the virtual human with believable appearance for interactive manipulation, animating it with minimal number of sensors to have maximal behavioral realism, and investigating different methods to decrease the networking requirements for exchanging complex virtual human information. 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 actors with their own behavior, which can perceive the environment and interact with participants. Figure XX shows example applications of the system.

**Figure: Example applications of Networked Virtual Environments**

**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 XXX shows the architecture of the VLNET client. The client consists of two types of processes: the core VLNET processes, and external driver processes.

**Figure: Client architecture of the VLNET system and its relationship with external 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 Performer processes, allowing 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; and to manage 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.

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 merging 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 actor 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.

**Virtual Human Modeling**

Real-time representation and animation of virtual human figures has been a challenging and
active area in computer graphics [BOULIC95][BADLER93]. 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 74 degrees of freedom without the
hands, 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 [BOULIC95].

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. Possible choices 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 [Kalra92]. It can be controlled on several levels. On the lowest level, an extensive set of Minimal Perceptible Actions (MPAs), closely related to muscle actions and similar to FACS Action Units, can be directly controlled. There are 65 MPAs, and they can completely describe the facial expression. 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 facial model.

*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 with minimal input. We can divide the virtual humans according to the methods to control them:

- Directly controlled actors: 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 actors: the external driver << guides >> the virtual actor by defining tasks to perform, and the actor uses its motor skills to perform this action by coordinated joint movements (e.g. walk, sit).

- Autonomous actors: The virtual actor 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).

Figure describing or demonstrating virtual human control. More info in caption.

Direct controlled actors

The virtual actor 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 [Thalmann93], 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 angles between limbs.

A complete representation of the participant actor's 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. [Molet96] discuss that a minimum of 14 sensors are required to manage a biomechanically correct posture, and Semwal et al. [Semwal96] 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 [Badler93], closed form solutions [Semwal96], and table lookup solutions [Capin95]. 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 arm motor should compute the joint angles within the right arm using this input. The arm motor 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. Figure XXXX shows examples of body controlled by this technique, with the tracker input on the hand and the head.

**Guided actors**

Guided actors are actors 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 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 actor guidance is guided navigation. The participant uses the input devices to update the transformation of the eye position of the virtual actor. 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 necessary joint angles. The motor is based on the walking model, guided by the user interactively or automatically generated by a trajectory. 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.

**Autonomous actors**

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 actors as introduced in the previous subsection are typically driven from the outside. Including autonomous actors that interact with participants increases the real-time interaction with the environment. Therefore we believe that it contributes to the sense of presence in the environment. The autonomous actors are connected to the VLNET system in the same way as human participants, and also enhance
the usability of the environment by providing services such as replacing missing partners, helping in navigation. As these virtual actors 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 actors 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 actors is based on their perception of the environment.

**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. 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.

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. For this problem, Boulic and Thalmann [Boulic92] introduced the coach-trainee method. This method is based on taking the direct updated posture are the reference posture, and applying a correction to this posture so that the end-effector is reached. We plan to use this method in the motion combination method.

**Reference to Agentlib**

**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

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. Obviously, the color images of 120 x 80 pixels, even compressed, do not fit in the standard VLNET message packets used by the Communication process. Therefore special data channels are open for this video communication.

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 XX illustrates the video texturing of the face, showing the original images of the user and the corresponding images of the Virtual Human representation.

**Figure:** Continuous real-time texture mapping of the actual face on the virtual face.

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), making the risk of error accumulation low. A reliability test is performed and the data is reinitialized if necessary. This makes the recognition very robust. 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

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 XX illustrates this method with a sequence of original images of the user (with overlaid recognition indicators) and the corresponding images of the synthesized face.

**Figure: Model-based coding of the face. The remote virtual actor representation replicates the real participant's facial gestures, using the communicated parameters.**

**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 [Lavagetto95] 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. An extremely 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.

**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.

**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 much larger then 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. In this section, we assume that the participants create messages in real-time with a speed of 10 frames/second.

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: compression, transmission, and decompression of the data. The lag of the overall transmission will be the sum of the lag of all these phases, hence we analyze the various aspects of communication:

- **compression computation at the sender site (CS):** 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. MORE

- **decompression computation at the receiver site (CR):** 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 much heavier than the one at the sender site because the messages from a potentially large number of participants have to be processed.

- **accuracy loss (AL):** We evaluate the loss of accuracy of the body posture with respect to the original input data.

We try to compare these issues in Figure ????. We consider four types of message packets that can be used to convey the body posture information:
- **global positioning parameters**: A 4x4 floating point matrix is used for each of the body segments, defining precisely its position. The rotation, scaling, and translation parameters of this matrix can be sent for each of 17 body parts. This data can be used directly to display the body.

- **joint angles**. These values are the degrees of freedom comprising the body, each represented by a floating point value. We evaluated 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 (usually head and hand). Complex motor functions have to be used in order to generate the body postures from this data.

- **state information**. Only the high level state information is conveyed which makes the messages extremely 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 complex (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). Here, as examples, we took a simple medium complex action (walking), and a complex action.

We analyze two situations with respect to the 3 input data possibilities, as discussed in the previous section: direct control, guiding through walking and grasping, and autonomous action.

The figures show that there are a wide range of possibilities to define and transmit human figure information. The choice of the control and message type will depend on the particular application requirements. Where high accuracy is needed (e.g. medical 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, as it balances 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).

**Similar possibilities apply for the face.**

**THE FACE COMPARISONS**

As the number of participants increases, the compression, transmission and decompression tasks will be excessive, and the speed might decrease significantly. Therefore, methods should be investigated to decrease this lag. An approach is not to send the information to a site at all if there is no or little interaction, using filtering techniques [Funkhouser96]. In addition, the dead-reckoning techniques may be applied to extrapolate the human information from the last received information. The initial results on human body dead reckoning have shown that up to 50% of the traffic can be decreased by simply applying predictive filtering to the joint angles [Capin97].

**Figure**: Bitrate requirements (more information in caption)
Virtual Tennis: an Example Application

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

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 actors 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 [Noser96]. Prusinkiewicz and Lindenmayer [Prusinkiewicz] 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 beautiful computer-generated images. This book is highly recommended for everybody who wants to get familiar with basic notions related to L-systems and their application in computer graphics. Our behavioral L-system [Noser96] 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 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.
Behavior control

In the tennis game simulation the different behaviors of the actors 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 a virtual actor 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 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 can also hear sound events and obeys the decisions of the referee. The player's 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.

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 actor has its own VLNET client process. The L-system interpreter communicates through humanoid shared memories of the VLNET processes. The representation of the participant in the L-system interpreter is reduced to a simple racket whose position is communicated through the net at each frame. The racket and head positions and orientations of the autonomous actor are communicated at each frame to the user client where they are mapped to an articulated, guided actor. This guided actor is animated through inverse cinematic according to the racket position. The referee is also represented by a guided articulated actor in the user client getting its position at each frame from the L-system animation process.

Figure: 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. The ball simulation is performed within the L-System interpreter contained in the referee, and its representation is updated through the shared memory of the database process, and communicated to other participants.

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 process 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.

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.

As discussed, we will work on human motion control for direct, guided and autonomous actors 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. We are grateful to Patrick Keller for his remarkable help in producing the VLNET scenes and images. We would also like to thank Mireille Clavien for the body model design, 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:


