A Taxonomy of Networked Virtual Environments

Tolga K. Capin, Daniel Thalmann
Computer Graphics Laboratory (LIG)
Swiss Federal Institute of Technology (EPFL)
1015 Lausanne, Switzerland
{capin, thalmann}@lig.di.epfl.ch

Abstract

There have been several networked virtual environments (NVEs) described in the literature. Each NVE system considers a separate part of the problem and a reference frame for listing all components and comparing different NVEs missing. We summarize the issues to consider while developing complete NVEs, and we compare the most characteristic solutions in the literature for each issue.

1. Introduction

In order to discuss the issues to be considered in NVE development and analysis, we can divide our taxonomy into three elements:

- *Preconditions* are the facts that NVE system designers have minimum control of. These typically include the target application requirements, constraints of the network, computing and interface equipment.
- *Design decisions* include the tools and parameters that NVE system developers can control, such as the protocol for communication and the client architecture.
- *Further issues* include the further tools and techniques that NVE system designers can exploit in order to run the NVE system more efficiently, and to increase the quality of the feedback to the user.

2. Preconditions of NVEs

2.1 Target Applications

There have been a number of pilot or working applications constructed in the literature. Although the theoretical approach of the early systems tried to provide a general-purpose virtual environment for any type of application (Bricken and Coco 1993), it was later realized that these systems lack efficiency, one of the principal requirements for NVE systems. Recent efforts concentrated on characterizing applications for different dimensions. Slater et al. (1995) proposed three such dimensions: 1) the number of participants and entities simultaneously involved in the same world. 2) the complexity of the objects and their behaviours, ranging from static data to those responding to participant interaction, and on to dynamically changing objects without user intervention. 3) the degree of interaction among participants, ranging from low (users can see each other), through medium (users can be involved in complex activities), to high (synchronized activities to achieve a common task). Each of the applications will need different network requirements and interentity synchronization; this will affect the development of the NVE system.

2.2 Underlying Network

The network characteristics underlying the NVE system can have a wide range. Funkhouser (1996) classified the network characteristics of wide area networks into three types: 1) *Connection*: this allows two workstations to send data unidirectionally over a connection-oriented link. An example is the modem link using a standard telephone line, with two-way, connection-oriented, unicast data transport with low latency and bandwidth. 2) *Unicast*: this allows a message to be sent to each other entity on the network for distributing messages. An Internet connection without the multicast capabilities is an example. This distribution of one update message requires $O(N)$ separate communications. 3) *Multicast*: a subset of workstations can communicate with each other

using connectionless messages. The underlying network should support the multicast operation. The MBONE (Multicast Backbone implemented over the Internet) is an example. This requires one update message for distributing a message.

For generality, heterogeneous networks can be constructed using a combination of these different network types.

2.3 Projected Hosts

The projected number of connecting hosts is an important factor in developing efficient NVE systems. Although the number of connecting hosts is expected to increase with the concurrent developments in network and CPU technology, we can postulate that some applications will require smaller numbers of participating hosts than others. These applications will typically require better display quality and representation than large-scale NVEs (e.g. teleconferencing). In addition, it is also important to consider the properties of the connecting hosts. The processing power of the workstation should be sufficient to cope with the messages received, and to perform the environment simulation and processing of remote entities without degrading the performance and quality of the simulation.

A number of research groups studied medium- to large-scale virtual environments. Among them, NPSNET was reported to be successfully simulated with 300 entities, theoretically shown to grow to thousands (Macedonia et al. 1995).

2.4 Input Devices and Rendering

Many different types of input device are in current use. In their number of tracked degrees of freedom, they range from the ubiquitous mouse to a large number of magnetic trackers attached to the body. Similarly, the display systems can be as simple as a desktop display, or a more complicated set of helmet-mounted displays.

3. Design Decisions for NVEs

The design decisions of NVEs are the parameters under the control of NVE system developers. Different than the preconditions of NVEs, these are the elements that can be changed during software development, in order to have an efficient simulation of the virtual world.

3.1 Host Program Architecture

An NVE is defined as a single environment shared by multiple participants connecting from different hosts. Each host typically stores the whole or a subset of the scene description, and lets the participant use their own avatar to move around the NVE. Additionally, the local program simulates the behaviour of a set of entities in the world, and also handles the real-world sensing using a set of input devices.

NVE software shares similar design goals as the other types of software: generality, usability, portability, understandability and efficiency. In addition, there are other characteristic goals of NVEs: rapid development of applications, modularity, decoupling of main VE tasks from the application, immersion and embodiment of participant.

Toolkit-Based Architectures

Toolkit-based architectures provide tools and libraries for creation and interaction with virtual environments. They provide: controlling the objects in the environment, moving the body representation, changing viewpoint, object relationships, display, management and synchronization of resources, networking multiple participants, obtaining statistics. Example toolkits are WorldToolKit, MR Toolkit, and other 3D graphics toolkits such as Performer, Java3D. We will review the advantage and disadvantage of each toolkit.

Integrated Software-Based Architectures

Integrated software-based architectures, in contrast to toolkit approaches, provide a complete system that implements basic VE tasks. This is similar to thinking of implementing the NVE system as a distributed database access problem, therefore the details of the main NVE tasks and synchronization of data are transparent to the application developer. The new applications are developed either by using the previously developed components, or by replacing them with new programs. VEOS, dVS, NPSNET, DIVE, MASSIVE, SPLINE, BrickNet, VISTEL are some of the systems.

3.2 Data and Task Distribution

Each virtual object and each participant embodiment within the NVE affects the performance of the different parts of the simulation: network, CPU, graphics. These overheads easily become significant with increasing number of participants and complexity of environments. The main solution to this problem is to let each participant's host computer process only the part of the environment it is interested in. Many researchers propose methods for dividing and sharing the VE data and processing at each host computer. Distribution may depend on the application, client architecture, network topology and other design decisions. Generally, the data distribution is not transparent; the application developer, designer of the applications and the participants should be aware of the data distribution, so it is neither possible nor necessary to make the distribution fully transparent. In this section and the following sections, we survey the various methods for *sharing* and *distributing* a virtual world over multiple participants, i.e. their host computers.

Sharing of the virtual world typically involves sharing the geometry, and transmitting object updates. However, some systems such as BrickNet allow applications to share the behaviours too.

Partitioning of the virtual world is necessary to decrease the CPU, network and graphics overheads of the simulation. Additional advantages of distribution are minimized communication between network tail links, and localized reliability problems caused when a host or link goes down (Macedonia et al. 1994). Macedonia et al. (1995) also suggest that there are three possible approaches for partitioning the scene: spatial, temporal, functional. Various research groups have proposed methods for spatial NVE partitioning.

Four schemes are possible (Capin 1999): *separate servers* (that separate the world into independent worlds), *uniform geometrical structure* (that divide the world uniformly), *free geometrical structure* (that divide the world based on users choices), *user-centred dynamic structure* (that divide the world based on interactions between users).

3.3 Network Topology

We have discussed possible network characteristics: unicast, multicast, broadcast. It is necessary also to consider network topologies for connecting the clients in the NVE. The choice of network topology depends on the network characteristics and applications, among other preconditions. Network topologies into two categories: *peer-to-peer* and *client–server*. It is also possible to use a hybrid combination of these topologies (Funkhouser 1996) (Capin 1999).

3.4 Avatar Representation

The participant is an important element in the integrated NVE system, so the embodiment should be represented realistically and efficiently. The embodiment has functions for self-representation and for representation to other participants in the same world. Here are some functions for *self-representation*: 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 functions for *interaction with others* are: perception, localization, identification, visualization of others' interest focus and actions, and social representation through decoration of the avatar.

Most of the common systems use simple avatars to represent participants. Among the most complex representations, the VLNET system from EPFL and Miralab uses a wide range of animation techniques from magnetic trackers to autonomous control, SPLINE system from MERL Labs uses procedural animation, VISTEL from ATR labs uses computer vision techniques.

3.5 Protocol

Developers of NVE systems should consider all the factors described up to now, in order to define the architecture of the system, the topology for communication, avatar representation, and the classification of entities in the VE. Once the NVE system designer makes these decisions, the system should implement a protocol to be able to communicate efficiently. This mainly depends on how the participants and virtual objects are represented, as well as the

preconditions and design decisions of the NVE system. The protocol will ideally contain session management, state and event information, and interaction among objects. Until now there has been a lack of efficient standard protocols for managing NVE systems. The Distributed Interactive Simulation (DIS) system provides a special-purpose protocol for military applications. On the other hand, a general-purpose reliable multicast protocol is used to synchronize distributed versions of the NVE world data. There is also a new standardization effort within the ISO SC29 MPEG-4 body to standardize bitstream syntax and semantics for computer graphics applications and a special ad hoc group has been formed for multiuser worlds.

4. Further Improvements

The fidelities in NVEs can be further improved by introducing the following additional features in the virtual environment:

- *Default parameters for different worlds*: The NVE system can provide default behaviours of objects in order to minimize the world design complexity. For example, the gravity property can be part of the world classification, and the application designer of the virtual world does not need to specify gravity behaviour for each entity.
- *Collisions*: Collisions significantly improve the quality of the virtual world simulation.
- *Access rights*: Virtual environments with large numbers of participants create the problem of accessing different objects and parts of the virtual world. This can be moderated by introducing access rights that participants need to be given to enter parts of the world, similar to access rights for files in operating systems.
- *Picking resolution*: By default, most of the current NVEs provide a coarse resolution to compute intersection tests for picking. This can be improved by introducing fine object manipulation techniques.
- *Force feedback*: Haptics is an important factor that increases the natural interaction within the virtual environment.

Furthermore, the NVE system developers can exploit various techniques to decrease the communication and computation overhead for tasks connected with participant-to-participant communication; and there are other techniques to increase the quality and response rate of the simulation. Here are some of them: 1) *Level of detail*: for representing the geometrical data in different resolutions. 2) *Motion level of detail*: for sending the state update messages to remote hosts in different rates, depending on their distance. 3) *Filtering messages*: so that only interested hosts receive the state changes of an entity. 4) *Dead reckoning*: a technique to decrease the network requirements, based on extrapolation of future states of entities. 5) *Compression of messages*: sending messages with a loss of information, for using less bandwidth. 6) *Synchronization*: so that each host machine has similar states of the virtual world.

5. Conclusion

We have presented a taxonomy for NVE systems. We believe that this taxonomy is a useful tool to analyze and compare various NVE systems.

References

Bricken W., Coco G. (1993) *The VEOS Project*, Technical Report R-93-3, Human Interface Technology Laboratory, University of Washington.

Capin T.K., Pandzic I.S., Thalmann D., Magnenat Thalmann N. (1997a) A Dead-Reckoning Algorithm for Virtual Human Figures, *Proc. VRAIS '97*, pp. 161–169.

Capin T.K., Pandzic I.S., Noser H., Magnenat Thalmann N., Thalmann D. (1997b) Virtual Human Representation and Communication in VLNET Networked Virtual Environments, *IEEE Computer Graphics and Applications*, Vol.17, No.2, pp. 42–53.

Capin T.K., Pandzic I.S., Thalmann D., Magnenat Thalmann N. *Avatars in Networked Virtual Environments*, John Wiley, June 1999.

Funkhouser T.A. (1996) Network Topologies for Scaleable Multi-User Virtual Environments, *Proc. VRAIS '96*, pp. 222–228.

Macedonia M.R., Zyda M.J., Pratt D.R., Barham P.T., Zestwitz (1994) NPSNET: A Network Software Architecture for Large-Scale Virtual Environments, *Presence, MIT Press*, Vol.3, No.4, pp. 265–287.