DISTRIBUTED AUDIO-VIDEO SHARING BY COPY-AND-TRANSFER OPERATION FOR NETWORK 3D GAMES

Hirotatsu Sakamoto   Yoshihiro Okada   Eisuke Itoh   Masafumi Yamashita
Graduate School of Information Science and Electrical Engineering, Kyushu University
6-10-1 Hakozaki, Higashi-ku, Fukuoka, 812-8581, Japan
Phone: +81-92-642-3872, Fax: +81-92-583-7632
hirotatu@swlab.csce.kyushu-u.ac.jp   okada@i.kyushu-u.ac.jp   itou@cc.kyushu-u.ac.jp   mak@csce.kyushu-u.ac.jp

KEYWORDS

Network 3D games, Audio-video communication, Copy-and-transfer, Component ware, Distributed virtual environments

ABSTRACT

This paper treats distributed audio-video sharing mechanisms for the development of network 3D games. Especially the authors propose the concept of the *copy-and-transfer* operation. This concept is that making a copy of a visible, manually operable software component and transferring it to another computer enable users to share it. If a facility that manages audio or video data is realized as such a component, even end-users can easily and rapidly build audio-video communication environments through the *copy-and-transfer* operation. This paper explains realization mechanisms of the *copy-and-transfer* operation and describes its availability by showing network 3D game examples.

1 INTRODUCTION

Advances of computer hardware technologies make it possible to create 3D images in real-time, so that 3D graphics software has become in great demand. For this reason, we have been studying 3D graphics software development systems and using *IntelligentBox* [Okada and Tanaka 1995] as our research system. *IntelligentBox* is a component ware, which provides various software components as visible, manually operable 3D objects called *boxes*. *IntelligentBox* also provides a dynamic data linkage mechanism called *slot-connection* that enables users to construct 3D graphics applications by combining existing *boxes* through direct manipulations on a computer screen. Okada *et al.* described that *IntelligentBox* is available for the development of interactive 3D games and also network 3D games [Okada *et al.* 2000]. For network 3D games, face-to-face communication by audio/video media enables players to feel their enemy's emotion and it enhances enjoy-ability during playing a game. Especially for group battle games, audio-video communication is necessary to effectively and strategically play games.

Then this paper treates a distributed audio-video sharing mechanism. Especially we describe a new concept of the *copy-and-transfer* operation. This *copy-and-transfer* operation is similar to the *copy-and-paste*/*cut-and-paste* operations, which are standard GUI operations based on using a mouse-device. With the *copy-and-transfer* operation, even end-users come to easily and rapidly build audio-video communication environments. *IntelligentBox* provided a video managing facility as a *VideoBox*. However this version of *VideoBox* was not available through network. So we improved it in order to support video communication via network [Sakamoto *et al.* 2001]. We have also been developing *boxes* to manage audio media through network. Furthermore a particular *box* called *RoomBox* exists for providing a shared 3D space [Okada and Tanaka 1998]. Using these *boxes*, i.e., *RoomBox*, *VideoBox* and audio managing *boxes*, which support audio-video communication, only through *copy-and-transfer* operations, it will be possible to build network 3D games. In this paper, we clarify availability of this *copy-and-transfer* operation by discussing development costs and performances with showing possible, practical application examples.

Related works

Our research purpose is to establish software architecture that makes it easier to develop 3D graphics applications. Related works are 3D softoware development systems including DIVE [Hagsand 1998], MASSIVE [Greenhalgh and Benford 1995], MERL [Anderson *et al.* 1995][Barrus *et al.* 1996], dVS [Ghee 1995], MR Toolkit [Shaw *et al.* 1993]. DIVE

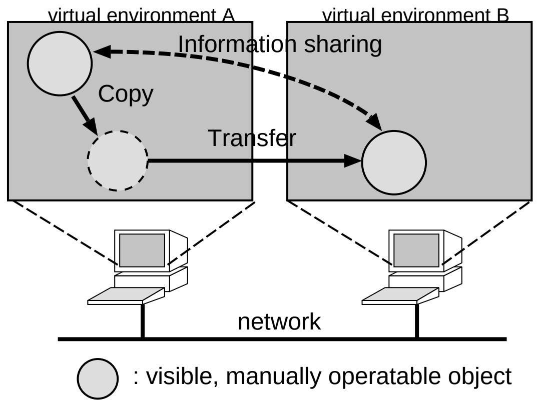

Figure 1: The concept of *copy-and-transfer.*

is a virtual reality construction toolkit system. It has network communication facilities by audio and text media. MASSIVE is a remote conference system based on sharing a 3D virtual space. It has a text-to-audio communication facility, but not a video communication facility. MERL is a development system for collaborative virtual environments. It allows us to communicate by audio and text media. dVS is a commercial product as a virtual reality construction toolkit. It allows us to develop distributed collaborative applications, but does not provide audio-video communication facilities. MR Toolkit is a virtual reality construction toolkit system at a library level. Although these are very powerful systems, it is not easy to use their essential mechanisms when developing distributed 3D graphics applications. Our research system *IntelligentBox* provides various 3D software components represented as visible, manually operable, and reusable objects. Furthermore the *IntelligentBox* system provides a dynamic data linkage mechanism. These features make it easier even for end-users to develop 3D graphics applications including network 3D games. This is the main difference between *IntelligentBox* and the others.

The remainder of this paper is organized as follows: Section 2 describes the *copy-and-transfer* concept. Section 3 explains essential mechanisms of *IntelligentBox* and *RoomBox*. In section 4, we explain audio-video data sharing mechanisms. Section 5 discusses development costs and performances with showing possible, practical 3D game examples. Finally, we conclude this paper in section 6.

2 COPY-AND-TRANSFER

Object-oriented programming becomes very common because of its reusability of software componets and its availability of bottom-up programming manner. However, conventional object-oriented programming is not enough for end-users, who do not have any programming knowledge, since it requests users to write text-based programs. We think that software components should be represented as visible, manually operable objects because such software components allow even end-users to develop software only by combining them interactively on a computer screen. Furthermore, for such visible, manually operable components, it is possible to execute the *copy-and-transfer* operation as shown in Figure 1. In this figure, the user using the left computer can make a copy of a certain component, and transfer it to the right computer. This operation is done only with manual operation using a mouse-device on a computer screen. If the copy and its original software component keep the same states, two users using each computer can share the same data as its states. Therefore the *copy-and-transfer* operation of audio-video managing components allows users to share audio-video data via network.

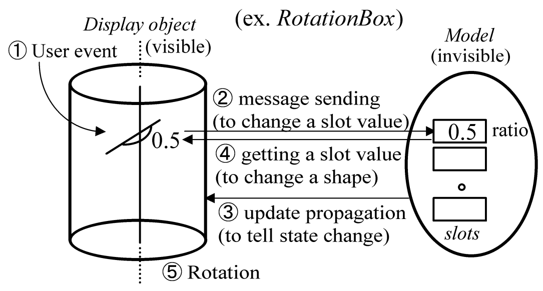

Figure 2: An *MD* structure of a *box* and its internal messages.

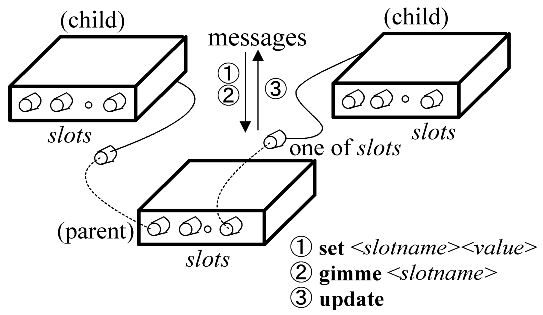

Figure 3: Standard messages between *boxes.*

3 ESSENTIAL MECHANISMS OF INTELLIGENTBOX

The following essential mechanisms are inherited from *IntelligentPad* [Tanaka 1996], which is a 2D synthetic media system, to *IntelligentBox*, since *IntelligentBox* is an extension of *IntelligentPad* to 3D graphics applications.

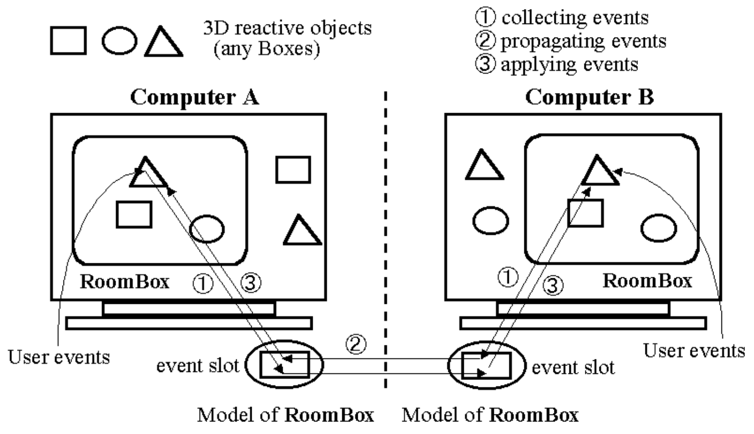

Figure 4: Message flow between two *RoomBoxes* for network collaboration.

3.1 Basic structure of box

As shown in Figure 2, each *box* consists of two objects, a *model* and a *display object*. This structure is called an *MD* (*Model-Display object*) structure. A *model* holds state values of a *box*. They are stored in variables called *slots*. A *display object* defines how the *box* appears on a computer screen and defines how the *box* reacts to user operations.

Figure 2 is an example of a *RotationBox*. A *RotationBox* holds a rotation angle as its *slot* value. Through direct manipulations on the *box*, this *slot* value changes(①②). Furthermore, its visual image simultaneously changes according to the *slot* value change(③④). Then the *box* reacts to the user manipulations according to its function(⑤).

3.2 Message-sending protocol for slot connections

Figure 3 illustrates a data linkage concept between *boxes*. Each *box* has multiple *slots*. Its one *slot* can be connected to one of the *slots* of other *boxes*. This connection is called a *slot connection*. The *slot connection* is carried out by three messages.

A *set* message(①) writes a child *box slot* value into its parent *box slot*. A *gimme* message(②) reads a parent *box slot* value and sets it into its child *box slot*. An *update* message(③) is issued from a parent *box* to all of its child *boxes* to tell them that the parent *box slot* value has changed. In this way, these three messages connect a child *box slot* and its parent *box slot*, and combine their two functionalities.

3.3 A shared-copy and a distributed model-sharing

The *MD* structure allows more than one *box* to share the same common *model*. This mechanism is called *model-sharing* and the operation that generates a copy of a *display object* sharing a common *model* is called *shared-copy*. A *box* generated by the *shared-copy* operation shares all *slot* values. After one of *model-shared boxes* is transferred to another computer via network, a new corresponding *box* is generated in that computer and the *box* has the same *slot* values and keep them always by messages via network to conserve consistency of *slot* values. This means distributed *model-sharing*.

3.4 RoomBox for collaborative virtual environments

This section briefly describes an idea of a shared 3D space and a functionality of a *RoomBox*. As shown in Figure 4, the *RoomBox* has a *slot* named 'event' which holds a current user-operation event operated on its descendant *boxes*. Some specific user-operation events generated in a *RoomBox* are always stored in this *slot* until the next event is generated. As mentioned above, *IntelligentBox* provides a distributed *model-sharing* mechanism. By this mechanism, multiple *RoomBoxes* can share user-operation events with each other. Here, descendant *boxes* of a *RoomBox* are treated as collaborative operable 3D objects.

In Figure 4, there are two *RoomBox models* existing separately on a different computer. These *models* are kept in the same state by messages passed via network. This linkage is built easily and rapidly by making a *shared-copy* of a *RoomBox* on one computer and by transferring it to the other computer. When a user

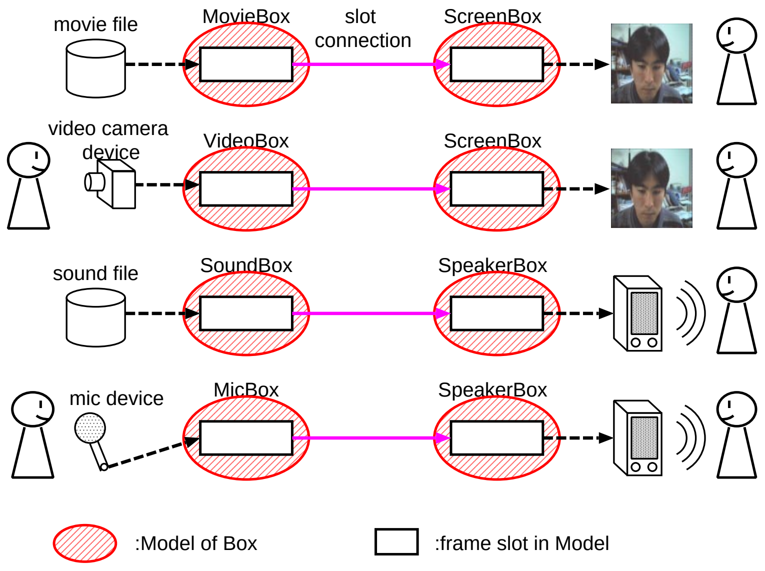

Figure 5: *Boxes* managing audio-video data and their pairs.

operates one *box* in the *RoomBox* on the computer A, his operation event is sent to the *RoomBox model* and subsequently set in its event *slot*. Furthermore, this event is sent to the other *RoomBox model* existing on the computer B by a message. After these processes are completed, the operation event is applied to the corresponding *box* on the computer B. In this way, by using distributed *RoomBoxes*, user-operation events are shared among several computers.

4 DISTRIBUTED AUDIO-VIDEO SHARING MECHANISMS

As previously mentioned, using *RoomBoxes*, it is possible to build collaborative virtual environments easily and rapidly. As well if *boxes* that manage audio-video data exist, it is possible to build audio-video communication environments easily and rapidly by their *copy-and-transfer* operations. Then we designed and implemented six *boxes* managing audio-video data as follows. Actually four *boxes* are used for audio-video communication, i.e., the pair of *VideoBox* and *ScreenBox* for video communication, and the pair of *MicBox* and *SpeakerBox* for audio communication.

1. *MovieBox* reads movie data from a movie file.
2. *VideoBox* gets movie data from a video camera device.
3. *ScreenBox* displays movie data onto its surface as texture images.
4. *SoundBox* reads audio data from a sound file.
5. *MicBox* gets audio data from a mic device.
6. *SpeakerBox* outputs audio data to a speaker device.

Figure 5 illustrates the usage of the six *boxes*. We can implement audio-video facilities as just one software component like Microsoft MediaPlayer^{TM}. However we designed them as six components separetely because of the following. We think that software components should be as simple as possible and should have the same metaphor as existing things in the real world. Such software components are very easier for end-users to deal with and have high reusability.

4.1 Boxes managing video data

MovieBox, *VideoBox*, and *ScreenBox* manage video data. *MovieBox* and *VideoBox* are used for reading video data, and *ScreenBox* are used for displaying video data. The texture-mapping technique is used to display a binary 2D image in a 3D virtual space. Strictly speaking, a texture image is mapped on the surface of a 3D object, i.e., a *box*. These *boxes* have a ‘frame’ *slot* in its *model*. A texture image is loaded and stored in this *slot*. Periodical updates of the *slot* content allow us to see an animation. *MovieBox* and *VideoBox* also have a ‘TRIGGER’ *slot*. Whenever ‘TRIGGER’ is accessed, the next frame will be loaded. Actually a *TimerBox* is used to access the ‘TRIGGER’ periodically by a *slot-connection*. A *TimerBox* holds a timer value, which periodically increases every user-specified interval time, in its ‘time’ *slot*. Then this *box* is used as a timer to notify *MovieBox* and *VideoBox* of its timing to get new frame of video data.

Figure 6 and Figure 7 illustrate distributed video data shaing mechanisms using *VideoBox* and *ScreenBox*. Figure 6 is the case without using *RoomBox*.

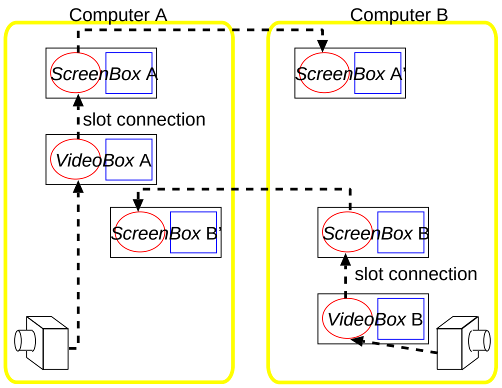

Figure 6: Distributed video data sharing mechanisms without *RoomBox*.

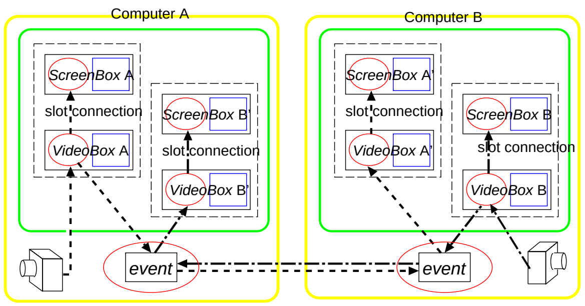

Figure 7: Distributed video data sharing mechanisms with *RoomBox*.

We can build this structure by means of the *copy-and-transfer* operation of *ScreenBox*. This is the simplest way to share audio-video data between two computers. For network 3D games, we have to build collaborative virtual environments using *RoomBoxes*.

Figure 7 is the case with using *RoomBox*. We can also build this structure by means of the *copy-and-transfer* operation of *RoomBox*, which contains a composite *box* of *VideoBox* and *ScreenBox*. Actually a *RoomBox* contains its child *boxes*, which are collaborative operable 3D objects. In this way, even end-users can build collaborative virtual environments including audio-video communication for network 3D games.

4.2 Boxes managing audio data

SoundBox, *MicBox*, and *SpeakerBox* manage audio data. *SoundBox* and *MicBox* are used for reading audio data, and *SpeakerBox* for playing audio data. These *boxes* have the common structure for managing audio-data. Actually these *boxes* have a dedicated buffer called "audio buffer," a 'frame' *slot*, and an 'always' *slot*. The audio buffer is used for storing temporal audio data sent from an audio device or read from an audio file. A part of the audio buffer is mapped to a 'frame' *slot*. The size of this *slot* is fixed. The timing of updating this *slot* value is controlled by an 'always' *slot*, which has a boolean value. If an 'always' *slot* value is true, the update timing is asynchronous. Strictly speaking, in the case of *SoundBox* and *MicBox*, audio data sent from a source is once written to audio buffer and its part is sent to a 'frame' *slot* asynchronously. In the case of *SpeakerBox*, audio data stored in its 'frame' *slot* is written to a part of an audio buffer asynchronously and is sent to speaker device. Actually a *ToggleSwitchBox* is used to change the 'always' *slot* value since a *ToggleSwitchBox* has a boolean value in its 'State' *slot* and it can be used as a switch.

5 DISCUSSION

This section discusses possible, a practical application example, their development costs and performances.

5.1 A practical application example

Figure 8 shows two screen images of a network game, which is a tank battle game actually we have already developed. In this example, there are two players each using a different computer. The left figure is a screen image of one computer and the right figure is that of the other one. As for the left figure, the upper right small view is a camera view of a *CameraBox* attached to the tank controlled by this computer's player. Each player can control his own tank with looking at the each camera view. Furthermore the two upper left small images are two players faces displayed using *VideoBoxes*. As this example case, face-to-face communication is important for enhancement of enjoy-ability during playing a game. Especially for a group battle game, audio-video communication is necessary to effectively and strategically play a game. Then our proposed *copy-and-transfer* operation is significant since it allows even end-users to construct network 3D games.

5.2 Development costs

We give an outline of the simplest way to create an application with video communication using *VideoBox*, *RoomBox* and *TimerBox*:

1. Compose a composite *box* from a *TimerBox* and a *VideoBox*.
2. Connect the 'time' *slot* of the *TimerBox* and the 'TRIGGER' *slot* of the *VideoBox* by a *slot-connection* through a menu selection.
3. Define the composite *box* as a descendant of a *RoomBox*.
4. Make a copy of the *RoomBox* and transfer it to another computer.

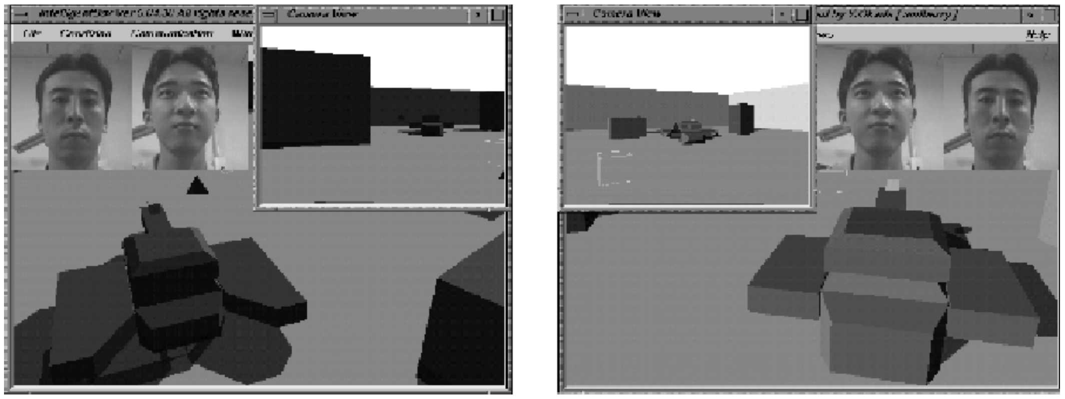

Figure 8: A distributed tank battle game.

In this way, construction process for video communication, based on the *copy-and-transfer* operation, is very simple and easy for even end-users. This process is done immediately only through mouse-device operations.

As mentioned in the paper[Okada *et al.* 2000], using *IntelligentBox*, it is possible to build 3D games without writing any text-based programs only through direct manipulations on a computer screen. The network game example without audio-video communication presented in this paper was also developed in several hours only through direct manipulations on a computer screen.

5.3 Performances

We experimented to evaluate the performance of video communication. So we executed a *copy-and-transfer* operation to each *VideoBox* on two computers and created video communication environment. The environment we experimented is as follows:

- computer A
  - CPU: Pentium III 800MHz
  - Memory: 256MB
  - Network: 100Base-TX
  - OS: Windows 98
  - Graphics Card: Geforce
- computer B
  - CPU: Pentium III 450MHz
  - Memory: 256MB
  - Network: 100Base-TX
  - OS: Windows NT
  - Graphics Card: Cobalt

Parameters are as follows:

- uni-direction or bi-direction
- frame rate(frame/sec)
- resolution(the numbers of vertical pixel $\times$ the numbers of horizontal pixel)
- color depth(byte/pixel)

Table 1 shows frame rates in some cases. For instance, the left lower values, i.e., 2.5 and 0.2, mean the frame rate of computer A to computer B and the frame rate of computer B to computer A respectively in the case that communication is bi-direction, frame resolustion is 128×128 pixels, and color depth is four byte/pixel. The frame rates of computer A to computer B are larger than the frame rates of computer B to computer A because of the difference of CPU peformances of computer A and computer B. In any case, the table does not show performances enough for practical use since we didn't use any data compression technique nor particular communication protocol. So communication performce can be improved if these techniques are used. These are left as a future work.

6 CONCLUDING REMARKS

This paper presented distributed audio-video sharing mechanisms for the development of network 3D games. Especially the new concept, i.e., the *copy-and-transfer* operation, was proposed and its realization mechanisms were explained. If a software component is represented as a visible, manually operable object, the *copy-and-transfer* operation on such an object becomes possible. Then if a facility that manages audio-video data is realized as such a visible, manually operable object, even end-users can easily and rapidly build audio-video communication environments through the *copy-and-transfer* operation. This paper clarified the availability of the *copy-and-transfer* operation by discussing development costs and performances of applications with showing network game examples.

resolution (pixel) depth (byte/pixel)		128 x 128 4	128 x 128 1	64 x 64 4	64 x 64 1
stand-alone	A	18.0	18.0	18.0	18.0
	B	6.6	6.6	6.1	6.1
uni-direct	A -> B	2.6	18.1	10.3	18.1
	B -> A	0.3	0.9	0.9	3.4
bi-direct	A -> B	2.5	17.0	10.3	18.0
	B -> A	0.2	0.8	0.9	3.1

frame rate (frame/sec)

Table 1: Frame rate of *VideoBox*.

ACKNOWLEDGEMENTS

We would like to thank all members of our laboratory for their advices and suggestions. This work is partially supported by foundation for Fusion Of Science and Technology(FOST) of Japan.

References

[Anderson *et al.* 1995] Anderson, B., D., Barrus, W. J., et al., 1995. "Building Multiuser Interactive Multimedia Environments at MERL." IEEE Multimedia, Vol 2, No. 4, 77-82.

[Barrus *et al.* 1996] Barrus, W., J., Waters, C., R. and Anderson, B., D., 1996. "Locals and Beacons: Efficient and Precise Support For Large Multi-User Virtual Environments." Proc. of IEEE Virtual Reality Annual Int. Symp. (VRAIS-96).

[Greenhalgh and Benford 1995] Greenhalgh, C. and Benford, S., 1995. "MASSIVE: A Collaborative Virtual Environment for Teleconferencing." ACM Transations on Computer-Human Interaction, Vol. 2, No. 3, 239-261.

[Hagsand 1998] Hagsand, O., 1996. "Interactive Multiuser VEs in the DIVE System." IEEE Multimedia, Vol. 3, No. 1, 30-39.

[Okada *et al.* 2000] Okada, Y., Itoh, E. and Hirokawa, S., 2000. "IntelligentBox: Its Aspects as a Rapid Construction System for Interactive 3D Games." Proc. of First International Conference on Intelligent Games and Simulation (GAME-ON2000), SCS Publication, 22-26.

[Okada and Tanaka 1998] Okada, Y. and Tanaka, Y., 1998. "Collaborative Environments in IntelligentBox for Distributed 3D Graphics Applications." The Visual Computer (CGS special issue), Vol. 14, No. 4, 140-152.

[Okada and Tanaka 1995] Okada, Y. and Tanaka, Y., 1995. "IntelligentBox:A Constructive Visual Software Development System for Interactive 3D Graphic Applications." Proc. of Computer Animation '95, IEEE Computer Society Press, 114-125.

[Sakamoto *et al.* 2001] Sakamoto, H., Okada, Y., Shimokawa, T. and Ushijima, K., 2001. "Component Based Video Communication Tool for Collaborative Virtual Environment." Proc. of 15th International Conference on Information Networking, 375-380.

[Shaw *et al.* 1993] Shaw, D., Green, M., Liang, J. and Sun, Y., 1993. "Decoupled Simulation in Virtual Reality with the MR Toolkit." ACM Trans. on Information Systems, Vol. 11, No. 3, 287-317.

[Ghee 1995] Ghee, S., 1995. "dVS – a Distributed VR System Infrastructure." In SIGGRAPH '95 CourseNotes.

[Tanaka 1996] Tanaka, Y., 1996. "Meme Media and a World Wide Meme Pool." Proc. of ACM Multimedia'96, 175-186.