*ICAT 2001*
*December 5-7, Tokyo, JAPAN*

# Real-Image-Based Distributed Virtual Reality System with Java3D

Seung-Woo Keum, Jong-Il Park, Youjip Won, Yong-Jin Park

swkeum@mr.hanyang.ac.kr, jipark@hanyang.ac.kr, yjwon@ece.hanyang.ac.kr, park@hyuee.hanyang.ac.kr

Division of Electrical and Computer Engineering,
Hanyang University, Haengdang-dong, Sungdong-gu, Seoul, 133-791, Korea

**Abstract**

In this paper, we propose a distributed VR system based on transmission of real-time video. One can join the virtual environment as a form of real video as well as an avatar. The background of the participant is removed with the chroma-key technique so that a more immersive virtual environment can be constructed. A server-client platform capable of handling time-based-media is implemented on a network environment with Java Media Framework and Java3D. A non-contacting interaction scheme is also proposed for the system. One can freely navigate the environment by a vision-based interaction in a personal space.

**Key words**: Distributed Virtual Reality, Real Image, Java3D, Computer Vision, Human-Computer Interaction.

## 1. Introduction

Recently, virtual reality is emerging as a practical approach for realistic human communications with the rapid progress in all aspects of VR technology during the last 20 years. It now provides pretty good interaction environments between human and computer far beyond the conventional desktop computer systems. High-speed networks become available with the advance of network technologies. The topic of this paper is on realizing a highly realistic 3D distributed virtual environment(DVE) on a high-speed network.

One of the main contributions of the paper lies in the implementation of a real-image-based DVE. The proposed 3D VR system can handle 2-types of users. One can participate in the DVE as a form of either a conventional avatar or a real video. Backgrounds of the real video are effectively eliminated to enhance the sense of reality. A user can select his/her own type of participation based on the system capability. Users with capture board and camera may select a video avatar while others select a conventional CG-based avatar.

Another contribution of the paper is a vision-based interaction system where users can freely navigate the virtual environment without a keyboard or control devices. A personal-space viewer is designed to control the system and to navigate the virtual environment.

There are some systems that use 2D image in 3D VR system. Mcintyre et al.[1] developed a Video Actor framework where they integrate 2D video actors into a 3D AR(Augmented Reality) narrative system. The system puts 2D image into a 3D scene along with other 3D objects. However, their compositing does not cover explicit keying technique. The Renderware[2] uses a gif-format picture in 3D VR systems. However, transmission issues over networks are not considered.

We will overview our system in Section 2. In Section 3, time-base media handling is described as well as our 2 kinds of java classes. Vision-based user interaction is described in Section 4.

## 2. Overview of DVE

The virtual reality system is designed with Java3D(Fig.2), and the media is transmitted with Java Media Framework(JMF) [3].

Our purpose in this paper is to construct a virtual world capable of handling time-based media, so that a user can join with his real image instead of an avatar. Users may feel more immersive and also comfortable when looking at real people. And the background of a user's video is removed and set transparent so that only the participant's body can be naturally shown in the 3D virtual world. To be more interactive, a simple vision-based interaction is set up, so that users can navigate the virtual world without conventional input devices.

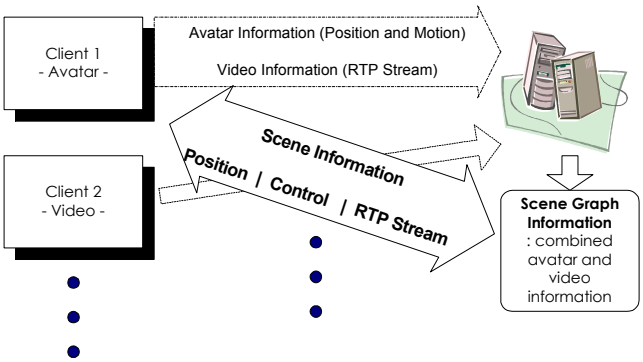


**Fig. 1 Concept of the server-client model**

In setting up a real-image object into 3D scene, we have two choices : IBR on 3D models[4] on high-end system and real video itself as texture on low-end. The first one may be more flexible, but it is difficult to create and animate in a networked VR. As mentioned in introduction, using 2D image in 3D world is a common technique in 3D graphics, and by virtue of the Java Media Framework, we can encode and transmit a reasonably sized video stream on network. Our prototype uses 2D video by texture mapping, and it's quality is set to 320x240, 15fps and H.263 encoded for network transmission.

### 2.1 Server-Client Design

The current version of the DVE is a server-client implementation. It is basically based on IP multicasting. Clients send data correspond to the user type, and the server multicasts it to all users. Since this system deals with time-based media, synchronization is very important. Here, the synchronization is done at the server side. The data from all clients are multiplexed at the server and transmitted to each client. When a client receives the multiplexed data stream, it will demultiplex the data into objects, and construct a 3D scene by using them. The data from each user-type is as follows:

- *Conventional Avatar: Control and location information of avatar.*
- *Video Avatar: motion picture (time-based media) and location information*.

We will focus on transmission of video avatar in this paper.

### 2.2 Video Stream Processing

The live video from the participant with video avatar should satisfy following specifications.

1. Good for network transmission : the video is encoded in H.263 in real-time with JMF.
2. Background elimination : It means that the background should be made transparent. Therefore, we first remove background by a generalized chroma-keying[5], and fill it with specific color(black here).
3. Vision-based user interaction : a custom effect class (MenuEffect) will detect the position of hand and fire corresponding event.

Thus, the video should do two jobs : chroma-keying and menu activation. We set up 2 JMF processors for these purposes, and implement ChromaEffect and MenuEffect for each job, respectively (Fig.3) .

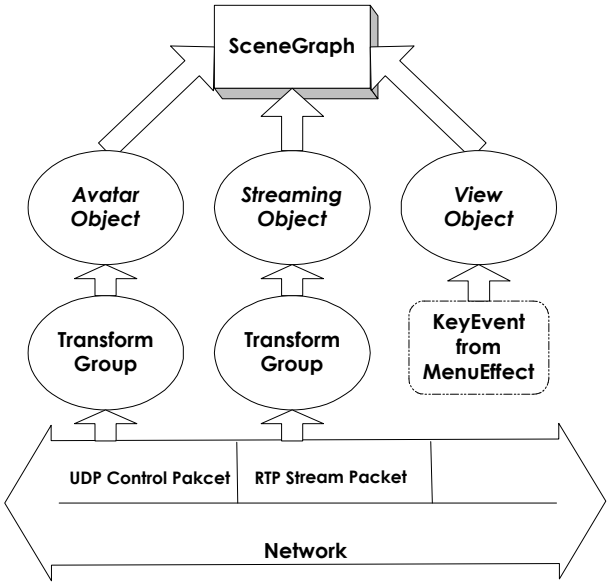


**Fig. 2 Virtual world with Java3D**

## 3 Handling Time-Based Media

Time-based media handling is done in 3 steps. First, the sender client generates media streams with chroma-keyed data. Then, the server multiplexes all of the client's data into one stream and retransmits them to each client. Finally, each client receives and demultiplexes the stream and renders it as a 3D scene.

When sending a live video of a participant, we use a chroma-key technique to eliminate backgrounds from the 2D image and then render it on a transparent 3D object with billboard property. To do this job, we build two kinds of Java class: TransmitVideo and ObjectBG.

The TransmitVideo class at the sender eliminates backgrounds using a chroma-key technique and sends it to the server as a video stream with JMF. As seen in Fig.3, this class has two Effect plug-ins. Class ChromaEffect is for chroma-keying. Chroma-keying can be done at either TransmitVideo or ObjectBG. We put it in TransmitVideo, because of the following reasons : (1) if the chroma-key is done in ObjectBG, computation burden at receiver client will be heavier as the number of users increases. (2) When ObjectBG class gets the stream, the background can be removed easily by setting transparency if color is black. In this way, we can save computation on receiver client. (3) Since there is no video format that supports transparency, we set the background color as black(color can be changed). The bit-saving by using constant background instead of adding transparency channel or using original background is significant.

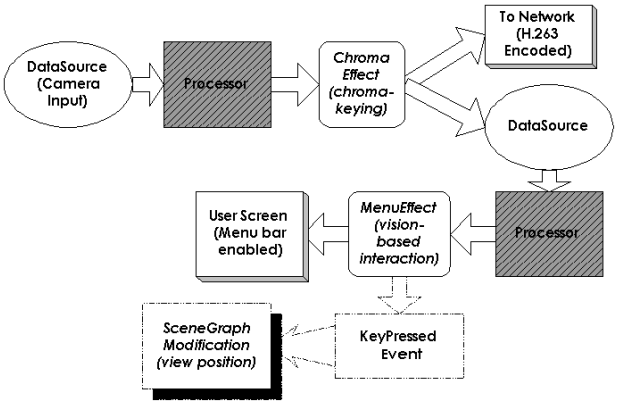


**Fig. 3 Class TransmitVideo and Effects** : this class has two output (shadowed box), one is chroma-keyed data for network transmission and the other for user interaction on user screen. Each has its own Effect plug-in. KeyPressed event is processed in KeyNavagation-Behavior in Java3D.

The ObjectBG class at the receiver is capable of receiving time-based media, and renders it on a 3Dobject with background-removal. By texture-mapping the 2D video on a transparent 3D object, we put background-removed 2D video in the 3D virtual world. As shown in Fig 6, the video input is textured in virtual world perfectly with a reasonable time delay due to encoding and network latency.

## 4. User Interaction

For a participant with real video, it is not easy to use a mouse or keyboards since he/she may not be so close to the computer. Moreover, users want to interact with computers without using cumbersome devices. Thus, a passive recognition system is requested. Our system employs a simple computer-vision technique. A separate window called *personal-space viewer* shows his/her body for a natural interaction(Fig.5). The image is mirrored one of the user image in Fig.6, which is seen from other clients. The mirror-type interaction scheme is very comfortable for human.

The interaction is based on position of user's hand. When user put his hand on top-left corner of his view, the menu will be activated. Currently, menu consists of navigation events. When navigation menu is activated, he can control his movement by positioning his hand. This is done by custom effect (MenuEffect in Fig.3). The MenuEffect class uses chroma-keyed video as an input, and detects the hand in the video.

As shown in Fig. 5 and Fig. 6, our system renders the video stream in virtual world well, and personal space viewer works correctly. But since H.263 encoding is done by software, it consumes too much computation of the system. When a client was not sending its camera input to the network, full 15fps stream was successfully decoded and textured in virtual world, but when video transmit start, 15fps was hardly accomplished.

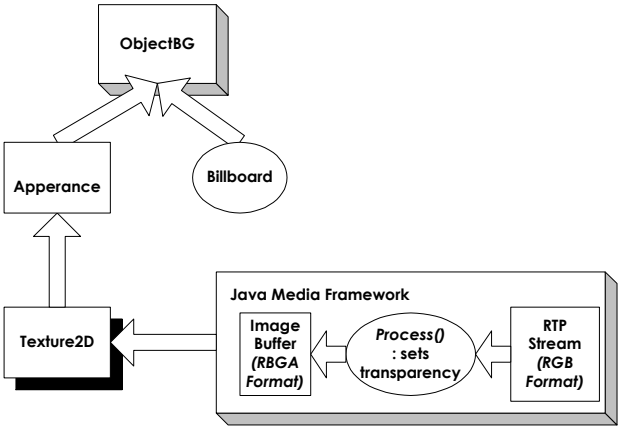


**Fig. 4 Class ObjectBG** : since chroma-key is done on class TransmitVideo, process() is just setting the transparency level.

## 5. Concluding Remarks

We introduced a prototype of a virtual world system with video avatar. It is capable of handling 320x240 sized 15fps media stream in H.263 format in real-time, and video is rendered as background-removed by chroma-keying. But since camera capture and H.263 encoding is heavy, the overall system performance was a little short of video-rate application.

There are several things to be explored in the future. The latency problem should be investigated intensively. Moreover, although the transmission is based on a server-client model at the current implementation, it should be changed to a peer-to-peer system to accommodate a large number of participants. Furthermore, we are going to add one more type of users where users can enjoy looking at an arbitrary view generated by image-based rendering[4].


**Fig.5 Personal Space Viewer** : User can navigate virtual world by positioning his/her hand on each colored box at the top of the image**.**

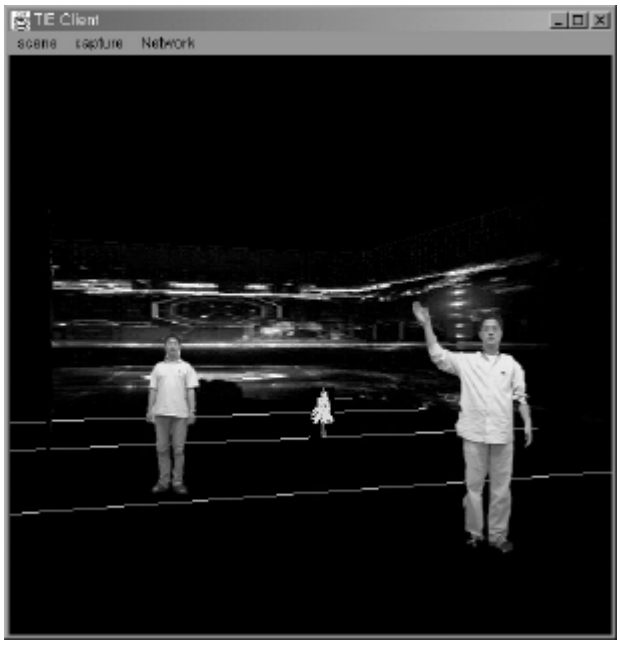


**Fig. 6 Client Program :** 3 Users are in virtual space with video avatar. Note that space around user is transparent. They can navigate either PSV or keyboard.

**Acknowledgement**

This work was supported by grant No. R01-1999-00232 from the Korea Science & Engineering Foundation

## 5. References

1. Macintyre et., al, "Ghosts in the machine : Integrating 2D video actors into a 3D AR system", *Proc. 2nd ISMR*, pp83-80, March 2001.
2. Renderware, http://www.renderware.com
3. Java Media Framework http://java.sun.com
4. Jong-Il Park and S. Inoue, "Arbitrary view generation using multiple cameras". *Proc. IEEE ICIP '97*, vol.1, pp.149-153, 1997.
5. Hyuk-Rae Park and Jong-Il Park, "Real-time vision-based virtual advertisement system", to appear in *Proc. 2001 Annual Conference of KOSBE*, Nov. 2001 (Korean)