An Adaptive Event Passing Model for Shared Virtual Environment

ChanYong PARK, JeongDan CHOI, JinSung CHOI, ChanSoo LEE, DongHyun KIM

VR Lab., Human Computer Interface Department,
Systems Engineering Research Institute
1 Ueun-dong Yusong-gu, Taejun, Korea, 305-333
e-mail: {cypark, jdchoi, jin1025, chanse, dhkim}@seri.re.kr

Abstract

The goal of this research is the development of a shared virtual environment (SVE) and a reduction of network traffic. To do this, We develop an authoring tool for shared virtual environment, a SVE player(Client) and a SVE server. The user can build the SVE with the SVE authoring tool build and participate in the SVE with SVE the player. When many players participate in the SVE, to reduce network traffic, we propose a event filtering function and event filtering threshold. The event occurred in the SVE player is applied to event filtering function.

Key words: Virtual Reality, Shared virtual environment, Multi-participant virtual environment, Event Filtering Function

1. Introduction

There are many virtual reality(VR) applications, but most of them have been developed to supply only a single user on a stand-alone system. With increasing availability of the Internet, many people are interested in distributed virtual reality system. The way of information representation change to the like of real world in 3D virtual space. If there is any way to get the goods regardless of his own position and time, a user is willing to choose the means. The Virtual Reality Modeling Language (VRML) [1][2] is the answer for this desire. Though the VRML is not perfect to cover every requests, it enables a user to describe and build 3D space on the Internet[3]. The representation of 3D space is very attractive in marketing and advertizing since this representation method is very intuitive and similiar to real world. Therefore they need a means to construct the 3D virtual space. Currently available methods are the graphic library and the commercial authoring tools.

With the increasing performance of modern PCs, the acceptance and distribution of 3D applications has become more common. The best known and probably most spectacular development in this area is that of the shared virtual environment(SVE)[4][5][6]. A Many people participated in the same SVE with own avatar. The user controls the avatar and the movement of avatar sends to other clients. If many people(avatars) are participated in the SVE, a network traffic increases rapidly. In this paper, we propose event filtering function and filtering threshold for decrease network traffic. The filtering threshold is changed dynamically by network speed. This means that we adapt filtering threshold to network speed.

This paper consists of five chapters. The second chapter discusses the related works. The third chapter discusses the overall structure of this system and suggests event weight function. The forth chapter discusses performance results and last chapter gives results and future works.

2. Related Works

In the DVE, all clients on network access the complex scene database representing the shared virtual world, since the clients will be updating the data, then the principal role of the database is to maintain a consistent copy of the data. Changes at the client side need to be propagated to the database, and used to update the scene in the database in a consistent manner. Those of higher image quality often lack smoothness of animation or sometimes can not even be animated.

The methodologies for time critical rendering in the DVE can be classified as follows:

· culling of the database
· occlusion filtering of outgoing data stream

The first step is making the scene hierarchically on an object or image space by a potentially visible set, and the second step is culling all sub parts of hierarchy culling the parent on the fly. The object space algorithms for hierarchical structuring uses bounding volume, bounding box [7], and deformed affine matrix [8], and slabs [9]. The second step is culling. It includes view frustum culling and occlusion culling. View frustum culling removes only parts out of view frustum. But the occlusion culling removes the occluded parts by objects, additionally. Image space algorithms for visibility culling have been presented in hierarchical Occlusion Maps [10]. Recently, Zhang has presented hierarchical occlusion maps on complex models with high depth complexity. The culling algorithm uses an object space bounding volume hierarchy and a hierarchy of image space occlusion maps. Compared with methods above, this algorithm is found to be generally applicable to all models and obtains significant speed ups for interactive walkthroughs on conservative graphics system. Greene [11] has proposed a hierarchical Z-buffer algorithm combining spatial and temporal coherence. It uses two data structures: an octree for object space and a Z-pyramid for image space. And in 1996, Greene has presented a hierarchical tiling algorithm using coverage masks. It uses an image hierarchy named a "coverage pyramid" for visibility operations. Traversing polygons from front to back order, it can process densely occluded scenes efficiently and is well suited to anti-alising by oversampling and filtering.

Object space algorithms for occulsion culling in general polygonal models have been presented in linear Critical Surfaces and Shadow Frusta. These algorithms dynamically compute a subset of the objects as occluders and user them to cull away portions of the model. In particular, Coorg [12] computes an arrangement correspondig to a linearized portion of an aspect graph and track the viewpoint within it to check for occlusion. Hudson [13] uses shadow frusta and fast interference tests for occlusion culling. Moreover, most of the work done on static visibility does not easily extend to dynamic environments. Sudarsky [14] has adjusted the structure with dynamic objects and output sensitive visibility algorithm which minimizes the time required to update the hierarchical data structure for a dynamic objects and minimize the number of dynamic objects for which the structure has to be updated.

The second method includes the filtering[15]and space subdivision. For further reduction of data, useless information can be filtered out of the update stream. One filtering technique is to calculate the distance that a user could "see". An object beyond that distance need not send its status to that user's avatar. The technique bears some relation to what in computer graphics research is known as level-of-detail, in which the number of polygons representing an object varies with the viewer's position. One filtering technique of this kind is based on distance, if the ratio of an object's size to its distance from the user is less than some predetermined threshold value, then its actions are not currently relevant to the user. Another technique is occlusion filtering. It is built on the fact that people ordinarily cannot see or clearly hear through solid objects (Wall, etc.). Some other objects may be obscured by fixed features in the environment (building, wall, etc.), or by the presence of certain environmental effects (fog, clouds, etc.) in the modeling software. Therefore, the update messages for occluded object would not be sent to the user.

Another form of relevance filtering uses spatial subdivision. The virtual world can be divided into zones and zone-to-zone visibility can be preprocessed. If the server computer analysis the boundary of the cells or regions, it will not send any updated messages for objects contained entirely in one region to another region's user. The filtering is to lessen the amount of data flowing through a distributed VE.

3. Shared Virtual Environment System

This chapter describes our experiment system[16], client -server network model and suggests adaptive event passing model.

3.1 The SVE authoring tool and the SVE player

This chapter describes the entire structure of our experiments system. The system consists of three parts. One is the SVE authoring tool and another is the SVE player and the other is SVE server.

The SVE authoring tool constructs 3D virtual environment supporting multi-participant. The Fig.1 shows the construction process of the SVE. The process of the SVE construction has three steps; First, making SVE template, second, arrange virtual objects and third, registering the SVE to the SVE server. The SVE template is basic structure of shared virtual environment. The SVE authoring tool imports virtual objects(VOs) which are made by 3D modeler and represent real object. The user who build the SVE is able to rotate, translate and scale VOs and is able to change properties - material and texture. The SVE authoring tool builds the SVE with the SVE template and VOs. Finally, the SVE is stored and registered in the SVE server. The SVE is linked to web page.

The Fig.2 shows the structure of Web server, the SVE server and the SVE player. The user connects web server with web browser and the SVE server with the SVE player respectively. The SVE player connects the SVE server with TCP/IP.

3.2 Network model

The SVE System developed in this paper is based on distributed client-server architecture. The synchronization in the SVE means avatar synchronization in the multi-player virtual environment. In this chapter, we discuss only the synchronization of avatar's movement (Fig.3). When avatar moves in the SVE, the SVE player sends avatar's position and rotation information to the SVE server. We define avatar's position and rotation information as a event. All event generated from an avatar does not always send to SVE server. The event is filtered by a event weight function. The event weight function describes next chapter precisely. The SVE server multicasts the event to the client that participates in the SVE.

A client which send event to SVE server receives own event. The client calculates a network speed(NS) between the client and the SVE server. The network speed is use for adaptive event filtering.

3.3 Event Weight function

This paper proposes adaptive event passing model which is caused avatar's movement in the shared virtual environment. If all event of avatar's movement is transferred to server, the shared virtual environment could maintain the consistency of many avatars, but network traffic may increase, therefore the number of client decrease in the same shared virtual environment. In this system, we suggests event weight function which filters avatar's event. The event weight function sums up five parameters. These parameter values are calculated from previous event, which is transferred to server to current event. The maximum value of parameters is 1.0 and minimum value is 0. The parameters are

- Time: avatar's stillness time
- Movement : avatar's move length
- Rotation : avatar's rotation
- Density: density of avatars
- Awareness: Awareness to other avatars

The Time parameter is how long time to avatar is still. It's proportion to avatar's stillness time. The more avatar stills, the more time parameter increases. We define difference of time as ΔT.

The network speed is critical parameters. But the clients that are connected in the shared virtual environment has different network speed. If parameters of event weight function sets with slowest network speed between client and server, the overall system speed may be slow down. Therefore we uses network speed to Time parameter for adaptive event filtering function.

$$\Delta T = TCurrentTime - TPreviousEventTime \quad (1)$$

The Time parameter calculates like this,

$$Time = \begin{cases} 100 & (\Delta T \geq NS) \\ \dfrac{\Delta T}{NS} & (\Delta T < NS) \end{cases} \quad (2)$$

The Movement parameter is sum up distance of avatar's movement. This is proportion to avatar's distance, that is the more avatar moves, the more movement parameter increase. We define sum up movement of avatar as Δ M. The Movement parameter calculates like this,

$$Movement = \begin{cases} 100 & (\Delta M \geq 5m) \\ \dfrac{\Delta M}{5} & (\Delta M < 5m) \end{cases} \quad (3)$$

The Rotation parameter is value of avatar's rotation. It's proportion to rotation value. The more avatar rotates, the more the rotation parameter increases. We define difference of avatar's rotation as ΔR. The rotation parameter calculates like this.

$$Rotation = \begin{cases} 100 & (\Delta R \geq \dfrac{\pi}{6}) \\ \dfrac{\Delta R * 6}{\pi} & (\Delta R < \dfrac{\pi}{6}) \end{cases} \quad (4)$$

The Density parameter is density of avatars. If avatar's event occurs in the high density area, The Density parameter increase. The Density parameter calculates

$$density = \frac{\#\text{ avatars which include square } (5m * 5m)}{\#\text{avatars which participated in virtual environment}} \quad (5)$$

The Awareness parameter is whether avatar is included by other avatars. In this system, a floor is the x-z plane in the virtual environment and we calculate only 2D-view frustum.

$$Awareness = \frac{\#\text{ view frustum which include the avatar}}{\#\text{avatars which participated in virtual environment}} \quad (6)$$

The event weight function composed of sum of five parameters with coefficient.

$$Event_{Weight} = \alpha 1 * Time + \alpha 2 * Movement + \alpha 3 * Rotation + \alpha 4 * density + \alpha 5 * Awareness \quad (7)$$

The coefficients are,

$\alpha_1 = 0.3$
$\alpha_2 = 0.2$
$\alpha_3 = 0.1$
$\alpha_4 = 0.1$
$\alpha_5 = 0.3$
$(\alpha 1 + \alpha 2 + \alpha 3 + \alpha 4 + \alpha 5 + \alpha 6 = 1.0)$

The maximum number of $Event_{Weight}$ is 1.0 and minimum number of $Event_{Weight}$ is 0. If $Event_{Weight}$ is larger than some threshold, Avatar'event send to SVE server through network.

The Threshold is determined by network speed. In the experiment, we determine the meaning of threshold.

4. Results and Performance

4.1 Results

We developed our system on the PC platform. In PC environment, the 3D graphic performance is lack of providing the real time process for avatar's movement and so on. Therefore, our PC system equipped with 3D graphic accelerator. We adopt Open InventorTM[17] as 3D graphics library.

Using the SVE authoring tool, we implement virtual shop on the Internet. Before the SVE authoring, we have modeled the SVE template of shop and many virtual objects. After authoring the shop, we upload this shop in the SVE server and links web page. To enter the SVE (virtual shop), user opens the web page with a web browser and selects virtual shop. After the SVE player launches, the user can participate in the SVE. As the result of implementation, we show a screen shots. Fig.4 is the SVE with multi-players.

4.2 Performance experiments

We experiments event weight function based on many different environment.

First, We experiments relationship between the number of event and the threshold of event weight function. The purpose of this experiment determines the optimal event weight function threshold in the very fast network environment. The system environment is that

- The number of avatar : 3
- The size of virtual environment : 10 m * 16 m
- The time of experiments: 1 min (Total number o f events: about 2000)
- The Configuration of Network: directly connected.
- The Network speed : <10ms (very fast)
- Threshold value: 0.0

The result is Table 1. In this experiments, we knows that critical threshold value is about 0.35 since, event weight function increase rapidly. This means that the 0.35 is optimal filtering threshold. If threshold of event weight function is 0.35, 1028 event are filtered in the event filter.

Second, We experiment adaptive threshold of event weight function based on the network traffic. The system environment is that

- The number of avatar: 3
- The size of virtual environment: 10 m * 16 m
- The time of experiments: 1 min (Total number o f events: about 1500)
- Threshold value: 0.0

The result is Table 2. The network speed is about 0.5 ~ 1sec. In the experiments, we knows that optimal event filtering threshold is 0.3. If threshold of event weight function is 0.3, 1013 event are filtered.

5. Conclusion

We develops the SVE authoring tool, the SVE player(Client) and the SVE server. For participating many clients, we propose the event filtering function. The network speed between the SVE player and the SVE server is the part of Time parameter in event filtering function. In the experiments, event filtering threshold is determined.

We plane to design more sophisticated event weight function. Currently, we ignore avatar's motion such as walking, sit down. We will include avatar's motion in the event weight function. We are going to experiments more different situation, too.

References

[1] http://www.vrml.com/

[2] Wolfgang Broll, David England. 1995, Bringing Worlds Together: Adding Multi-user support to VRML, *Symposium on VRML*, 1995

[3] J. Hartman, J. Wernecke: *The VRML 2.0 Handb ook, Building Moving Worlds on the web*, Addis on Wesley, 1996

[4] Shaw, C., Green, M., Liang, J., and Sun, Y. De coupled Simulation in Virtual Reality with The MR Toolkit. *In ACM Transaction of Information Systems*, Vol, 11, No.3, (July 1993), pp. 287-31 7

[5] Wolfgang Broll, Tanja Koop, VRML: Today and Tomorrow., *Computers & Graphics*, Vol. 20, N o. 3. 1996

[6] Carlsson, C. and Hagsand, O. 1993. "DIVE - A platform for multi user virtual environments," *C omputer and Graphics*. Vol. 17, No. 6, pp. 663-339

[7] J. H. Clark, "Hierarchical Geometric Models for Visible Surface Algorithm," *Communication of A CM, 19(10),* pp. 547-554, Oct. 1976.

[8] S.M. Rubin and T. Whitted, "A Three Dimensio nal Representation for Fast Rendering of Comple x Scenes," *Computer Graphics(Proc. of ACM SI GGRAPH '80),* pp. 110-116, 1980.

[9] T.L. Kay and J.T. Kajiya, "Ray Tracing Comple x Scenes," *Computer Graphics(Proc. of ACM SI GGRAPH'86),* pp. 269-278, 1986.

[10] H. Zhang and D. Manocha, "Visibility Culling using Hierarchical Occlusion Maps", *SIGGRAPH' 97*, pp. 77-88, 1997.

[11] N. Greene, M. Kass, and G. Miller, "Hierarchic al Z-Buffer Visibility," *Computer Graphics (Proc. of ACM SIGGRAPH '93)*, pp. 231-238, 1993.

[12] S. Coorg and S. Teller, " Real-Time Occlusion culling for Modes with Large Occluders", *ACM Symposium on Interactive 3D Graphics*, 1997.

[13] T. Hudson, D. Manocha, "Accelerated Occlusio n Culling using Shadow Frusta", *ACM Symposiu m on Computational Geometry*, 1997.

[14] O. Sudarsky and C. Gotsman, "Output-Sensitive Visibility Algorithms for Dynamic Scenes with Applications to Virtual Reality", *EUROGRAPHIC S'96*, pp. 249 - 258, 1996.

[15] B. Roehle, "Channeling the data flood", ", *IEE E Spectrum*, pp.32-38, 1997.

[16] Chanyong Park, 1997, A Development of Auth oring tool for shared virtual environemnt, *Procee dings of the Thirteenth Symposium on Human Int erface*, 1997, pp419-422.

[17] J. Wernecke: The Inventor Mentor, Addison We sley, 1994

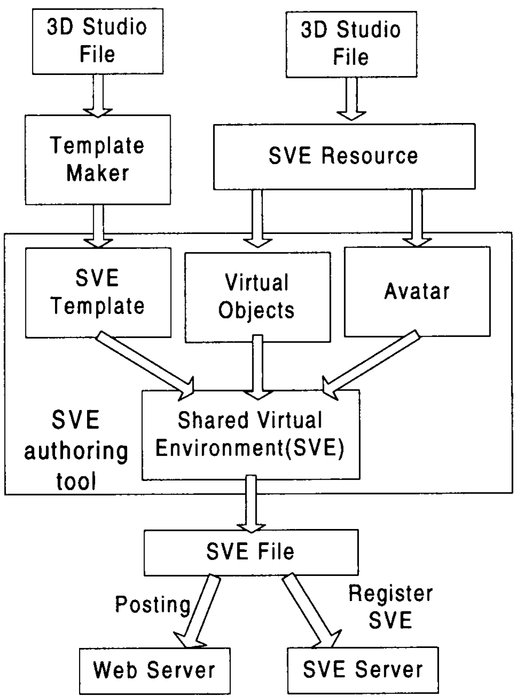


Fig.1 The SVE Authoring Tool

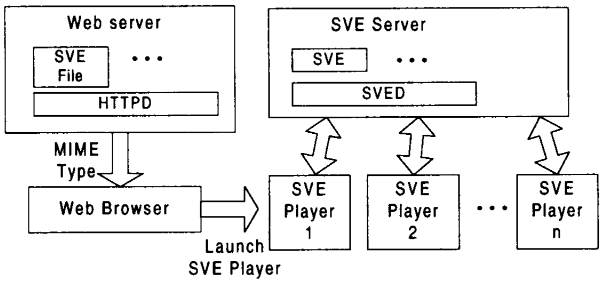
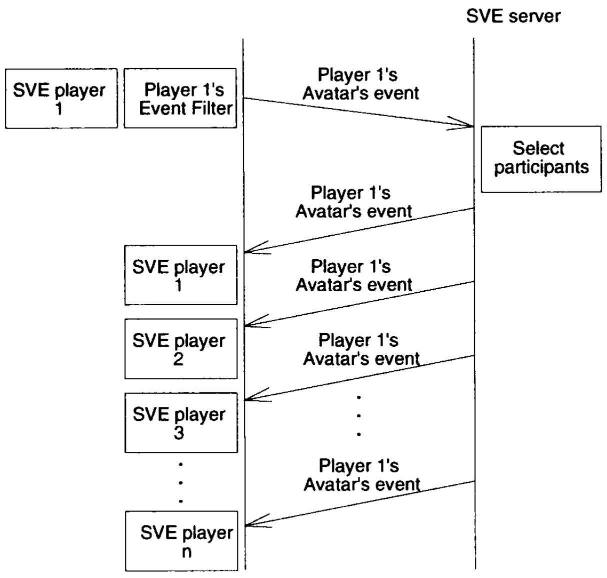


Fig.2 The SVE and the SVE player



SVE server
SVE player 1
Player 1's Event Filter
Player 1's Avatar's event
Select participants
Player 1's Avatar's event
SVE player 1
Player 1's Avatar's event
SVE player 2
Player 1's Avatar's event
SVE player 3
Player 1's Avatar's event
SVE player n

Fig.3 Synchronization of avatar position

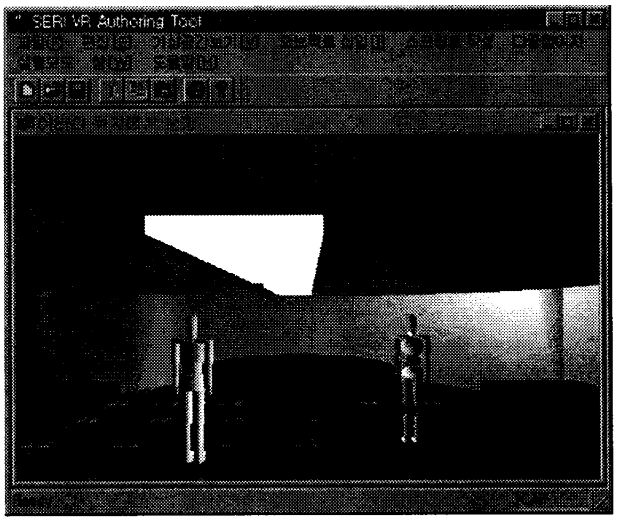


Fig. 4 The SVE player connected to the SVE server

Table 1. The event weight function in fast network

Value of Event Weight Function	The number of event
0 ~ 0.1	93
0.1 ~ 0.2	434
0.2 ~ 0.3	671
0.3 ~ 0.4	464
0.4 ~ 0.5	133
0.5 ~ 0.6	90
0.6 ~ 0.7	68
0.7 ~ 0.8	33
0.8 ~ 0.9	6
0.9 ~ 1.0	8

Table 2. The event weight function in slow network

Value of Event Weight Function	The number of event
0 ~ 0.1	270
0.1 ~ 0.2	442
0.2 ~ 0.3	422
0.3 ~ 0.4	148
0.4 ~ 0.5	112
0.5 ~ 0.6	60
0.6 ~ 0.7	28
0.7 ~ 0.8	13
0.8 ~ 0.9	4
0.9 ~ 1.0	1