VR Game Interfaces for Interaction onto 3D Display

Jeong-Dan Choi, Byung-Tae Jang

Spatial Information Technology Center, ETRI
161 Kajeong-dong, Yuseong-gu, Taejon, S. Korea
{jdchoi, jbt}@etri.re.kr

Chi-Jeong Hwang

Image Processing Lab., CNU
220 Gung-dong, Yuseong-gu, Taejon, S. Korea
cjhwang@ipl.cnu.ac.kr

Abstract
This paper describes the VR game interfaces for interaction onto 3D display. The challenge we are interested in is how to build and use a low cost 3D display that satisfies semi-immersive requirements and how to coordinate it with interactions into the immersive VR game environments. So, the objective of our display system is to produce a viewer-oriented, stereoscopic display; each of the user's eyes should see an image correctly rendered from that eye's position through the window of the displays. To accomplish this project, we concrete a projection-based multiple window rendering system on passive 3D stereo display, synchronization multi-channel system for wide field of view on different PCs, and game user interactions using real-time virtual character animations. Our preliminary results are well suited for VR Safari as game contents with surround displays running on a PC clusters.

Keywords:
3D stereo Display, Passive projection display, Real-time animation of virtual character.

1. Introduction

Virtual Reality (VR) technology fulfils a user to immersive with natural and intuitive user interaction in the virtual environments. The use of Virtual Environment (VE) is to enhance the degree of freedom of the construction under the given constraints to simulate the real worlds. This VR system is composed of the hardware and software that enables developers to create and execute VR applications with visual aspects and interactive responses.

In the visual aspects, the large physical size of each display surface allows users to interact with 3D objects, which can be critical to immersion and perception. So, we use 3D stereo display composed of multi-PCs and multi-projectors. The space in front of the display supports natural collaborations among multiple participants simultaneously viewing and entertaining visual data and enables immerse applications in which the entire FOV (field of view) for each user. And advances in large scale computing have allowed researchers to produce more data than ever, requiring in very various applications. A typical VR system was used only specific applications using high costs graphics workstations, such as military, scientific visualization, medical image analysis. But, today's PC-based graphics accelerators achieve better performance – both in cost and speed. So cluster of PC where many or all of the system have 3D accelerators is an attractive approach to building a scalable graphics system. And the multi computer displays are increasingly important for applications such as collaborative computer-aided design, interactive scientific, and entertainment and medical visualization. The high-resolution of the displays enables visualization of very detailed data sets.

Many conventional arcade games have typically used monitors as their display devices. While monitors were cost-effective but easy to use, they failed to convey a feeling of immersion to their users. We believe that multi-user VR Game display requires a different model of immersive environment, one that delivers a sense of presence among the game environments, incorporates 3D semi immersive visualization capabilities, is available all the time, is easy to use game device command and helps encourage the entertainment. So, we are interested in is how to build and use a low cost platform that satisfies these requirements and learn how to use it to bridge into the fully-immersive VR game environments. As well as, there are many growths in the user interface. These interfaces must have a high degree of clarity and allow the interaction in the form of human-like behavior through computer-aided tools.

In the followings, we introduce a preliminary example of VR safari game which is based on a projection-based multiple window rendering system on passive 3D stereo display, synchronization multi-channel system for wide field of view on different PCs, and game user interactions using real-time virtual character animations.

2. Previous Work

Numerous researchers have built low cost, PC-based CAVE-like systems. Immersive projection technology devices, typically CAVE® [7], immersive desks [7], and their derivatives are often used as superior tools for data visualization and analysis of these very large amounts of data.

A Projection-based display system, a CAVE®, commonly uses a 6 degree-of-freedom tracking device, with two or more sensors. One sensor tracks the user's head position and orientation, the others track a wand, gloves, or other control devices. The majority of currently fielded systems use electromagnetic trackers. Some drawbacks to this approach, however, are that these devices are relatively expensive and not commonplace and that they are designed specifically for visualization, not collaboration, or commercial VR Game.

The typical display device is a CRT projector and rear projection screen. In large-scale systems, there are multiple projectors and screens which are edge-aligned. To solve this problems one method is to use the hardware edge-blending [3], and the others use software methods [2].

Stereoscopic display in projection-based systems is most commonly done using liquid crystal shutter glasses, which are synchronized with the frame-sequential stereo video on the screens. But, this system or HMD (Head Mounted Display), but this system causes headache not to entertain the game contents.

Researchers in wide FOV rendering have investigated a variety of techniques to simulate the styles of dual head display. Several displays are connected one rendering system. But it failed to convey a feeling of immersion because of low-resolution. Recent work has resulted in methods for multi-channel using cluster rendering PCs [2].

3. Our Proposed System for VR safari

This section describes in detail our proposed VR Game contents, with some specific display devices graphically shown also in Figure 1.

VR safari is the virtual world of safari and on-line Role Playing Game (RPG) supported by 2 persons. Main system supports surround display with multi-channel PC Clusters. And it drives various VR interfaces with DirectX APIs that is composed of DirectInput for haptic jacket, seat and force feedback wheel, DirectMusic for 3D stereo sound, DirectPlay for network services and Direct3D for rendering of X file format.

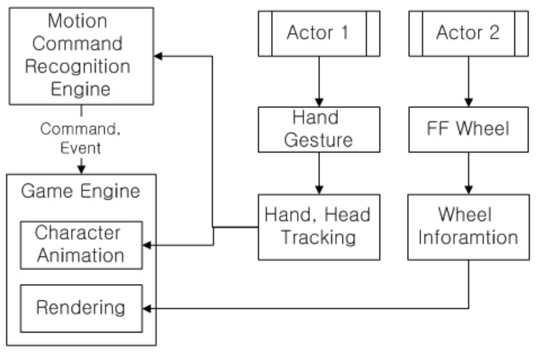


Figure 1: *Overview of the User Interface and Our System*

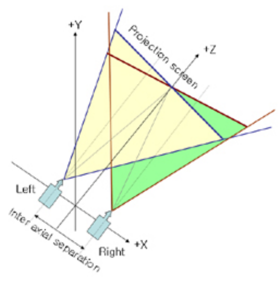
A user plays game with hand glove which trackers the hand positions and triggers the button event. The other user drives the safari off road with force feedback wheel handle. The hand gesture of player1 is captured by two CCD cameras. And then it is transmitted to Motion Command Recognition Engine. The extracted game command is transfered to Rendering and Animation Engine. And the motion of actor is generated with virtual actor by real-time onto screen. Player 1 wear see-through and polorized HMD and player2 wear polorized glasses. So, the players entertain the safari hunter with stereo wide screen.

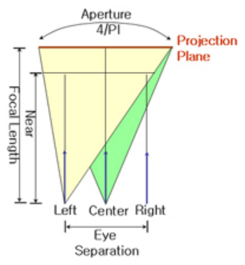
3.1. 3D Stereo Rendering Using DirectX

The following explains about creating stereo pairs using Direct3D. Regardless of whether it is an active system or a passive system, one has to create two views for both eyes.

The first thing one has to do is initialising buffers for stereo operation. The next step is to select the appropriate buffer for rendering. These procedures for a passive stereo application are given in the section 3.2. All that's left now is to render the scene with the appropriate projection to the selected buffer. A common approach is the so called "toe-in" method [4], where the camera for the left and right eye is pointed towards a single focal point and *D3DXMatrixPerspectiveFovLH( )* is used (see Figure 2(a)).

However a stereo scene using the "toe-in" method gave a while giving workable stereo pairs is not correct, it also introduces vertical parallax which is most noticeable for objects in the outer field of view. The correct method (see Figure2(b)) is to use what is sometimes known as the "parallel axis asymmetric frustum perspective projection". In this case the view vectors for each camera remain parallel and a *D3DXMatrixPerspectiveOffCenterLH()* is used to describe the perspective projection (see Figure 3).





(a) **(b)**

Figure 2: *Toe-in(a) and Correct(b) method*

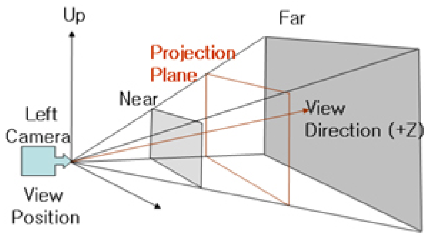


Figure 3: *Perspective view frustum*

The following is the procedure for the projection setting. First we set camera and view frustum parameters as follows.

FocalLength : distance from eye to the screen
Aperture : FOV in radians
Near : z value of the near clipping plane
Far : z value of the far clipping plane
EyeSep : ditance between two eyes. Usually set by 1/20 of *FocalLength*

HalfHeight is the half of the height of the near clipping plane and calculated by using the following.

$$HalfHeight = Near * \tan(Aperture / 2) \quad (1)$$

Next we can set the view matrix and projection matrix for the left eye. The view matrix is set by using *D3DXMatrixLookAtLH()* and its 3 parameters as follows.

LEyePt : vector(-*EyeSep* / 2, 0, 0)
LLookAtPt : vector(-*EyeSep* / 2, 0, *FocalLength*)
UpVec : vector(0, 1, 0)
D3DXMatrixLookAtLH(matView,
LEyePt, *LLookAtPt*, *UpVec*)

Referring to Figure 2(b) minimum and maximum x values of the view volume are calculated by using the following euqation.

$$Left = -Aspect * HalfHeight + EyeSep/2 * Near/FocalLength \quad (2)$$

$$Right = Aspect * HalfHeight + EyeSep/2 * Near/FocalLength \quad (3)$$

Where *Aspect* is the aspect ratio of the rendering window. The projection matrix is set as follows.

D3DXMatrixPerspectiveOffCenterLH(matProjection,
Left, *Right*, -*HalfHeight*, *HalfHeight*, *Near*, *Far*)

Finally we can set the view matrix and projection matrix for the right eye similarly using the following parameters.

REyePt : vector(*EyeSep* / 2, 0, 0)
RLookAtPt : vector(*EyeSep* / 2, 0, *FocalLength*)

$$Left = -Aspect * HalfHeight - EyeSep/2 * Near/FocalLength \quad (4)$$

$$Right = Aspect * HalfHeight - EyeSep/2 * Near/FocalLength \quad (5)$$

3.2. Passive Stereo for Surround Display Using DirectX

To configure a low cost platform we use a multi-head VGA card. So, we can build rendering system with smaller number of PCs. Each left and right image from two display adapters of a card is connected separately to LCD projectors. The simple splitter is attached onto VGA card. The splitter toggles the left and right image correctly and change the interlacing and page flip mode. And a polarizing filter mounted in the optical light path of each projector ensures that the correct information passes through its corresponding filter in the pair of passive stereo glasses. This configuration shows in Figure 4.

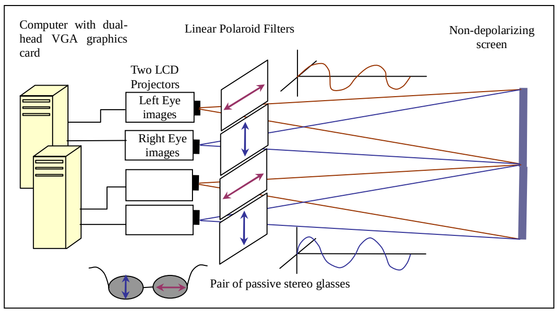


Figure 4: *Configuration of 3D Stereo Display System*

Former multi-head problems are poor API support and duplication of resources such as frame buffer. Because almost all new VGA cards support dual-head or triple-head configurations, we implemented multi-head display system with one VGA card having dual-head adapter. DirectX 8.1[5] also supports for rendering to multiple windows through the creation of additional swap chains.

All resources are shared with all the heads, and each head has exactly the same capabilities. Each head can be set to independent display modes and can be refreshed at different times. Figure 5 shows left and right images from a dual-head graphics device.

For a surround display system we have to modify the projection setting in the section 3.1 for tiling the multiple screens.

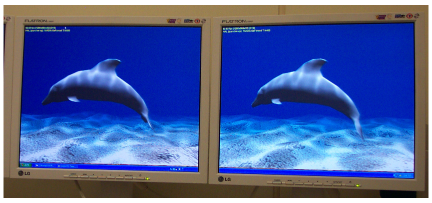


Figure 5: *Left and right images from a dual-head device*

An example left-eye projection settings for horizontally aligned 2 screens are as follows. Where *Aspect* is the aspect ratio of one screen. We can easily expand this method to arbitrary number of screens.

$$Left = -(Aspect * 2) * HalfHeight + EyeSep/2 * Near/FocalLength \quad (6)$$

$$Right = EyeSep/2 * Near/FocalLength \quad (7)$$

$$Left = EyeSep/2 * Near/FocalLength \quad (8)$$

$$Right = (Aspect * 2) * HalfHeight + EyeSep/2 * Near/FocalLength \quad (9)$$

3.3. User-Interactions with Character Animations

The tracking system reports the latest position and orientation of the head and wand trackers. From the head data, and predefined offsets, the positions of the user's eyes are calculated. Input game command is a marker-free system that consists of two synchronized CCD cameras installed front-left and front-right of a user and two clients system connected to each camera. The commands are transferred to Rendering and Animation engine.

A character represented as a skinned mesh and stored in an X file is rendered using Direct3D graphic libraries. In order to make our animation engine expansible and flexible we make it possible to load any motion data from a motion file (in Biovision's Bvh format) onto a character loaded from an X file. A Biped uses the x-axis as the bone-axis while a Bvh file uses an arbitrary axis. In order to solve this situation a normalized bone-axis (bx, by, bz) in Bvh is aligned with the y-axis first, and then the local coordinate system is circulated ($(x, y, z) \rightarrow (y, z, x)$) so that the x-axis becomes the bone-axis. The alignment can be done by a rotation about the axis R_{axis} by θ represented in (10) and (11).

$$R_{axis} = (bx, by, bz) \times (0,1,0) = \begin{vmatrix} i & j & k \\ bx & by & bz \\ 0 & 1 & 0 \end{vmatrix} = (-bz, 0, bx) \quad (10)$$

$$\theta = \cos^{-1}\left[(bx, by, bz) \cdot (0,1,0)\right] = \cos^{-1} by \quad (11)$$

The world matrix W_{ni} for rendering the i-th segment (bone) at n-th frame in a motion becomes (12).

$$W_{ni} = T_{n1} \cdot M_1 \cdot M_1^{-1} \cdots T_{n(i-1)} \cdot M_{i-1} \cdot M_{i-1}^{-1} \cdot T_{ni} \cdot M_i \quad (12)$$

Where M_i is the transformation matrix of the i-th segment representing a rotation about R_{axis} by θ, T_{ni} is a local transformation matrix holding the pose for i-th segment at n-th frame, and the index i is determined by the depth level from the root. When an additional correction matrix C is needed, a new world matrix becomes then $W_{ni} \times C$. This is the case in clavicle, upper arm, lower arm, and hand.

4. Implementation and Conclusions

We described the method of 3D stereo display construction with dual-projectors based clustered rendering system for scalable display in order to generate high-resolution images at real time frame rates. Display system using LCD dual projectors, which is infocus Proxima model 3300 ANSI lumens. And the PC cluster is Pentium 4, 2GHz CPU with NVIDIA® Geforce2 Ti 4600 models. The resolution is 1280x1024 for each left and right eye image. We skew the screen arbitrary, and then captured elevation screen data. The rendering results look so good. And also we introduce our prototype marker free motion capture system supports for real-time generation of virtual actors.

Finally, our current research includes further work on the VR game event process, projector calibration process and extension of the current ideas to handle automatic multiple projectors.

References

1. Funkhouser, F and Li, K., Large-Format Displays, *IEEE CG&A* 20, 4, (July/August 2000), 20-21
2. Brian, W, Constantine, P and Vasily, L, Scalable Rendering on PC Clusters, *IEEE CG&A* (July/August 2001), 62-70
3. Ramesh. R and Paul B., A Self Correcting Projector, TR-2000-46, (Jan 2002)
4. Course notes, #24, Stereoscopics, *ACM SIGGRAPH*(1989)
5. Zhang, Z., Flexxible camera calibration by viewing a plane from unknown orientations, *Computer Vision* (1999), 666-673
6. Microsoft Corporation. DirectX 8.1 programming reference, http://www.microsoft.com/windows
7. C.Cruz-Neira, J. Leigh and et al., Scientists in Wonderland: A Report on Visualization Applications in the CAVE Virtual Reality Environment, *IEEE 1993 Symposium on Research Frontiers in VR*, 1993.
8. Pattie Maes, Bruce Blumberg, Trevor Darrell, and Alex Pentland, "The Alive System: Wireless, Full-body Interaction with Autonomous Agents", *ACM Multimedia Systems*, No. 5, 1997, pp. 105-112.