International Journal of Signal Processing, Image Processing and Pattern Recognition
Vol. 10, No. 7 (2017), pp. 9-20
http//dx.doi.org/10.14257/ijsip.2017.10.7.02

Preventing Rendering Speed Degradation for Helmet Mounted Display Simulation

Ju Ho Lim[1], Chung Jae Lee[1], Kyong Hoon Kim[1] and Ki-Il Kim[2*]

[1]*Department of Informatics, Gyeongsang National University, Jinju, Korea*
[2*]*Department of Computer Science and Engineering, Chungnam National University, Daejeon, Korea*
[*] *kikim@cnu.ac.kr*

Abstract

Recently, the research on Helmet Mounted Display (HMD) that displays aircraft status information on a pilot's helmet is active. Accordingly, there is a growing interest in simulations for verifying new technologies related to HMD. It is important to maintain Frame Per Second (FPS) because HMD simulation processes data with large sizes such as terrain. However, there is a problem of performance degradation in a rapidly changing field of view movement. Previous studies have used Level of Detail (LOD) technology to improve the performance by changing the resolution of the model, but it has a problem of low realism. To solve this problem, we propose HMD simulation software that improves performance and realism. The proposed software applies and adjusts the existing LOD technology and motion blur technique in real time according to the movement of the field of view. Through experiments, we confirmed that the performance of the proposed software is improved by about 70%.

Keywords*: Helmet Mounted Display, Real-time rendering, Motion blur*

1. Introduction

Recently, the development of the aircraft is verified by applying the mathematical model of the aircraft to the simulator for performance verification at the design stage. This process analyzes the problems occurring in the development process, facilitates the modification, and shortens development time and development cost. As a result, many aviation equipment developers are testing the performance of the equipment they develop through simulations when developing their equipment, which in turn reduces development costs.

Among these, the aircraft HMD is hardware mounted on the pilot's helmet, and is a video display device that briefly displays the status information of various aircraft displayed on the Head Up Display (HUD) in the helmet of the pilot. It displays various information such as the navigation information required when operating the aircraft, as well as the radar information used when engaging the bandit and the armed information on the aircraft being operated. Such HMD changes the position of the symbol displayed on the screen according to the visual field movement of the pilot wearing the HMD. Such visual field motion information is referred to as LOS (Line Of Sight). This LOS means a distance that can reach the electromagnetic wave linearity. However, the LOS during flight operations is important information that can detect the accuracy of the HMD motion and the change in the view point of the pilot in flight.

Therefore, in this paper, we propose a simulation software system for the HMD for the performance verification of the system during the design stage of the HMD development process. The proposed HMD simulation software renders terrain data using satellite image and Digital Terrain Elevation Data (DTED) to provide 3D virtual reality based on LOS data of HMD, it receives status information of the aircraft generated from commercial flight simulation software and operates according to this data. It also displays the HMD symbol on the screen in accordance with the received data. In addition, since the real-time rendering software of the HMD simulation system reduces the Frame Per Second (FPS) when the pilot's field of view changes rapidly, it includes a rendering optimization mechanism to address this problem.

In this paper, we use two techniques to improve the performance of real-time rendering system. The first technique is the motion blur effect to improve the realism, and the second technique is the LOD (Level Of Detail) for improving the rendering performance. Among the two techniques, we use the LOD technique to reduce the number of data and texture size, and apply the motion blur effect to reduce the realism when the LOD technique is applied to the terrain data. HMD simulation software using the proposed mechanism was implemented using an open source based OSG, OpenSceneGraph, library, and measured rendering performance using 10m class satellite image, Yecheon airport 100km radius and 30m elevation data[8-11].

The rest of this paper is organized as follows. Section 2 introduces the existing system of 3D rendering of virtual reality and LOD and motion blur used for optimization mechanism. In Section 3, we describe proposed system architecture and mechanism. In Section 4, experimental results and conclusions are presented.

2. Related Work

To render 3D virtual reality, real-time rendering system requires object data. The rendering data may be a single object or a large amount of terrain data. The rendering data is structured as follows. It also has a texture for image mapping in the triangle mesh. These data are computed based on the rendering pipeline of the graphic library, and the computation results are displayed on the screen. The result displayed on the screen of the 3D real - time rendering system is called virtual reality. Recently, there have been many studies on 3D virtual reality using HMD.

Nozomi Sugiura propose a system that enables dynamic 3D interaction with real and virtual objects using an optical see-through head-mounted display and an RGB-D camera. The proposed system uses a collision sensing physics engine to enable the user to wear the device and interact with virtual objects [1]. As another HMD study, Aryabrata Basu has made it possible for many users to experience various kinds of HMD equipment for virtual reality. After that, they analyzed the effect of the 3D rendering technology applied to each HMD device on the quality of experience based on the result value. To do this, they measured the HR and EDA, which are objective physiological metrics, after using various kinds of HMD products. They analyzed the differences between the users in the VR and non-VR environments based on the test questionnaire that the user created after using the HMD device and they confirmed the effect of the QoE(Quality of Experience) on the users in the VR environment[2].

The static LOD technique divides data according to object precision. Since this technique changes the details of the object according to the distance from the camera, it is troublesome to construct an object model having different details depending on the distance. Also, it cannot be applied to a large amount of data like terrain data because memory leak is severe.

The technique to solve this problem is dynamic LOD. Dynamic LOD is a technique that changes the detail of an object according to the distance between a viewpoint and an object in the real-time rendering process. This technique has less memory leak than static LOD. Therefore, it is suitable to the real-time rendering system for processing large-capacity terrain data. This dynamic LOD technique is also used to study merge and split meshes of objects in real time.

Filip Biljecki measured the vertices of objects in the real-time rendering process for the LOD level selection, and proposed a framework for dynamic LOD based on this information [3]. In addition, Wang Ying implemented a fixed quadtree-based dynamic LOD to improve the FPS in the process of rendering large-scale terrain data[4]. This study improves existing dynamic LOD implementation algorithm using quadtree and eliminates T-type facture problem of quadtree. In another example, Mark Duchaineau utilizes Queue to perform segmentation and merging of triangle meshes in the LOD level selection process along the distance, and they propose a method of segmenting and merging triangle meshes using information from previous frames[5]. In addition, Guanbo Bao proposed a hybrid LOD scheme that applies static LOD for trees (objects) and dynamic LOD for terrain to render large-scale forests and terrain[6].

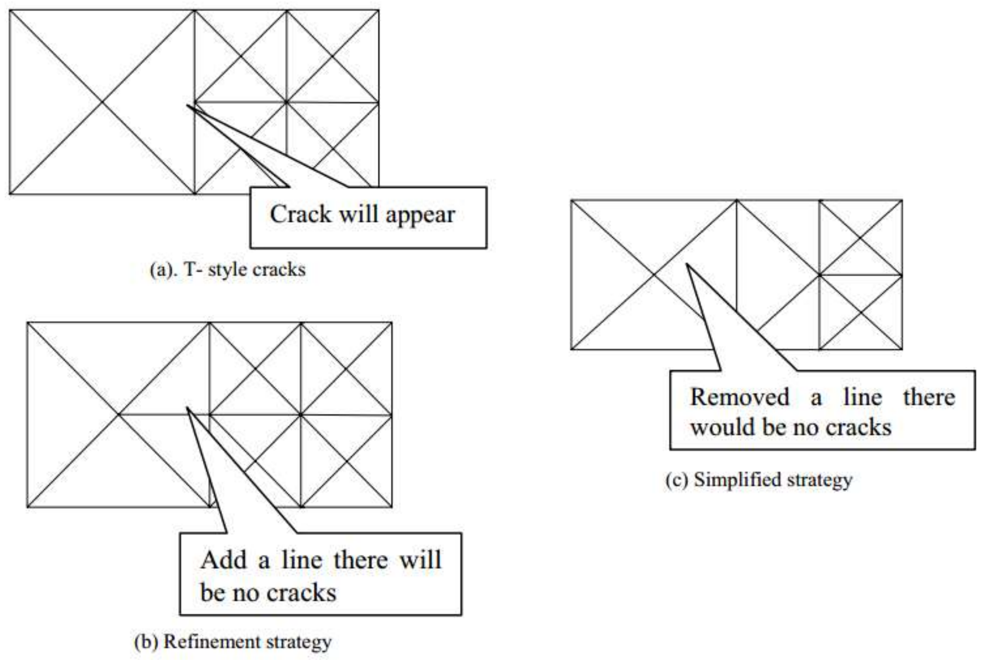


Figure 1. T-type Facture

The above-mentioned LOD-related studies propose a method for merging and dividing triangular meshes effectively and improving the speed in selecting the LOD level. However, since the real-time rendering system for HMD simulation changes the field-of-view change rate rapidly, the data in the view-frustum continuously changes when the general dynamic LOD technique is applied, thereby reducing the FPS. In addition, the T-type facture phenomenon occurs very frequently in the process of changing the LOD of the terrain data, so that the reality is degraded as shown in Figure 1.

To solve this problem, a real-time rendering system for HMD simulation requires a mechanism suitable for a real-time rendering system in which the visual field changes quickly and continuously. Therefore, in this paper, we use the method of adjusting the LOD scale in the level selection process of the dynamic LOD technique. However, since this method cannot solve the T-type facture effect, we apply the motion blur effect at the same time.

Motion blur is a technology that gives a residual image effect according to the speed of a moving object in a real-time rendering image, and is used as a technique to improve the realism in a real-time rendering system. Matthias M. Wloka implemented a motion blur function in a real-time rendering system utilizing hardware-assisted rendering of translucent polygons, which are commonly available on workstations today, and achieved performance degradation.[7] However, since the motion blur technique requires additional operations for the afterimage effect, there is a disadvantage that the rendering speed is reduced.

Therefore, in this paper, we propose HMD simulation software that improves FPS by using LOD scale adjustment mechanism and simultaneously improves realism by using motion blur technique.

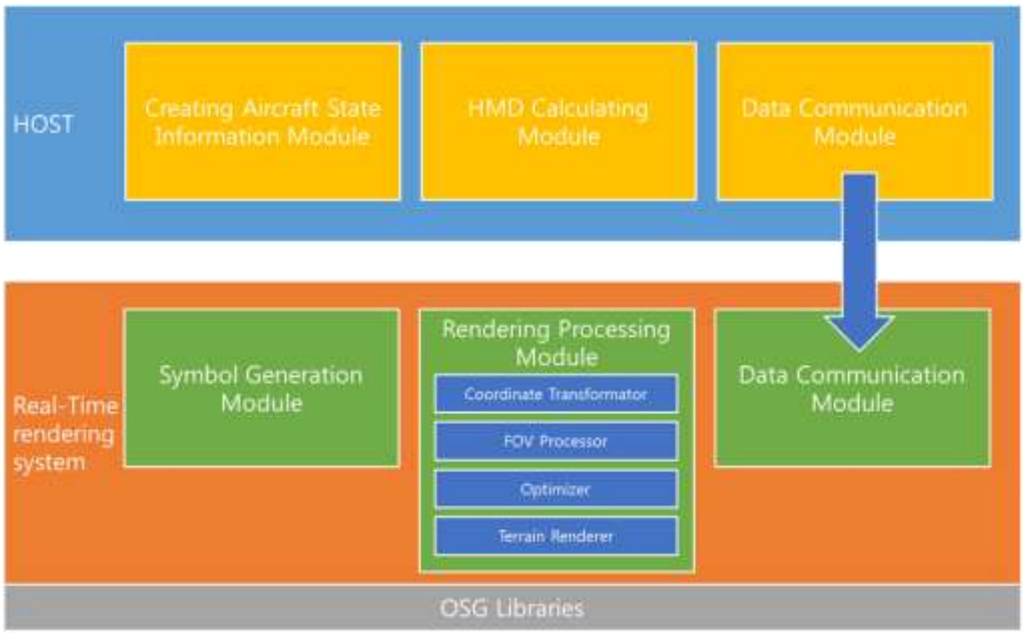

HOST
Creating Aircraft State Information Module
HMD Calculating Module
Data Communication Module
Real-Time rendering system
Symbol Generation Module
Rendering Processing Module
Coordinate Transformator
FOV Processor
Optimizer
Terrain Renderer
Data Communication Module
OSG Libraries

Figure 2. System Architecture

3. Proposed Architecture and Mechanism

In this section, we describe the overall system architecture and mechanism.

3.1. Overall System Architecture

The system architecture illustrated in Figure 2 is divided into two parts. First, the HOST consists of Creating Aircraft State Information Module that processes information about an aircraft and information about symbols on a display, HMD Calculating Module that processes information on LOS(Line of Sight), and Data Communication Module that transmits data to the real-time rendering system. The real-time rendering system consists of Data Communication Module that receives data from the HOST, Symbol Generation Module that generates and displays symbols of HMD, and Rendering Processing Module that processes the simulation rendering with the received flight information and HMD information.

The rendering processing module is composed of Coordinate Transformator which converts latitude/longitude coordinate information into OSG format coordinates, FOV Processor which processes the terrain data with LOS information, Optimizer which processes the rendering algorithm which improves the performance reduced by fast FOV movement, and Terrain Renderer that displays the rendered data. HOST and real-time rendering system communicate by UDP method and send/receive data.

3.2. Optimizer

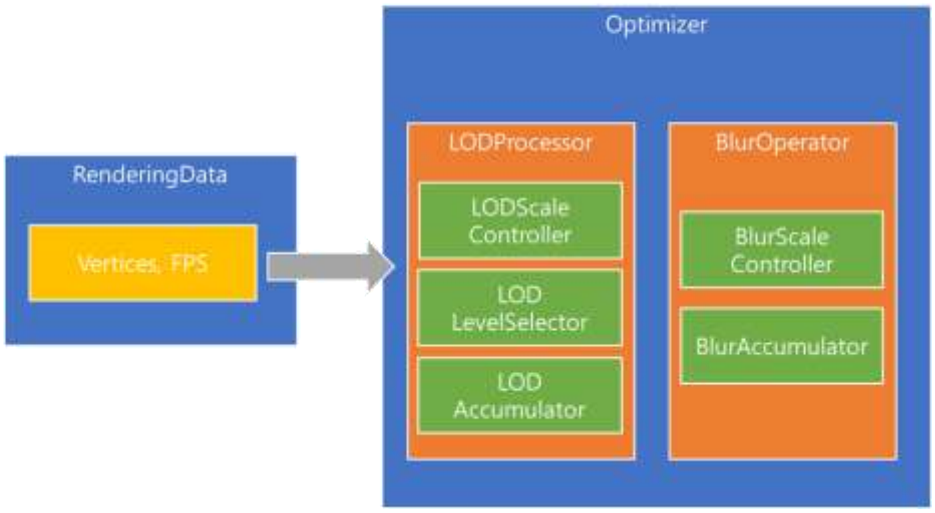


Figure 3. Optimizer Architecture

Figure 3 shows an Optimizer that handles algorithms that improve rendering performance. Optimizer receives and process information about vertices and FPS, and consists largely of LODProcessor and BlurOperator. LODProcessor consists of LOD effect processing modules to improve the performance of mesh data in View-Frustum, and BlurOperator consists of modules for setting the rendering screen camera and applying blur effect. LODScale Controller of LODProcessor controls the entire LOD range of the mesh data in View-Frustum. LODLevelSelector sets the distance between LOD levels. LODAccumulator computes and redefines LOD of the mesh data according to LOD scale and level. BlurScaleController processes the duration and scale of the afterimage effect and BlurAccumulator applies the afterimage effect to the data in the rendering screen.

3.2. Proposed Mechanism

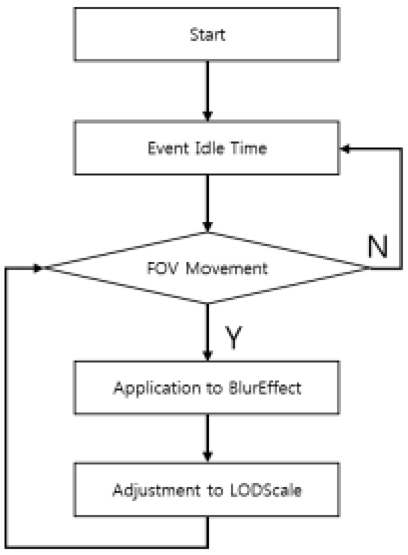


Figure 4. Proposed Mechanism

Figure 4 shows the overall mechanism for improving rendering performance. Event Idle Time is waiting for the movement of the field of view. FOV Movement determines the movement of the field of view by the amount of change of the position (Pitch, Heading), which is the state information of the aircraft or pilot receiving from the HOST. After that, the blur effect is applied according to the change amount of the position vales and the movement of the field of view speed. LODScale is a step that adjusts LOD step according to the movement of the field of view and prevents overloading of operations.

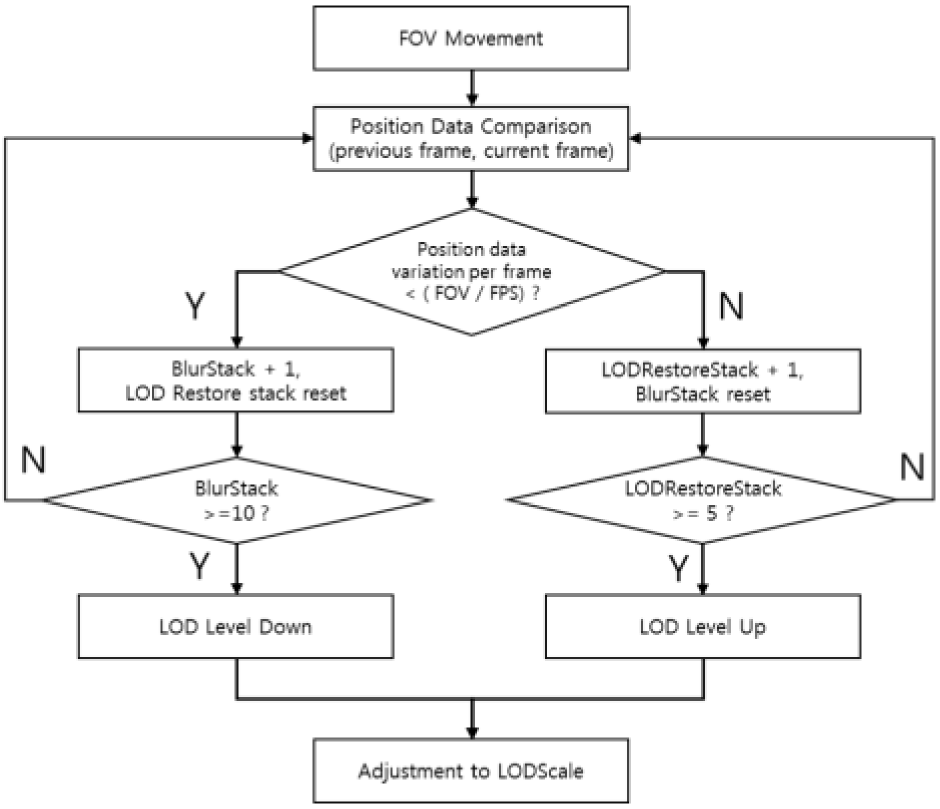

FOV Movement
Position Data Comparison (previous frame, current frame)
Position data variation per frame < (FOV / FPS) ?
Y
N
BlurStack + 1, LOD Restore stack reset
LODRestoreStack + 1, BlurStack reset
N
BlurStack >=10 ?
LODRestoreStack >= 5 ?
N
Y
Y
LOD Level Down
LOD Level Up
Adjustment to LODScale

Figure 5. The Flow from the Process of Determining FOV Movement to Adjustment with LOD Scale

As you can see in Figure 4, FOV Movement determines the movement by the position value of the aircraft or the pilot. After the data of the pitch and heading of the previous frame and the current frame are stored, the data of the previous frame and the current frame are compared with each other, and the amount of change of the position value is checked to determine whether the field of view is moving. The change of the position value can be judged whether the speed is fast or slow by using the following equation 1.

$$Position\ data\ variation\ per\ frame > \frac{maximum\ left\ FOV + maximum\ right\ FOV}{Average\ FPS} \quad (1)$$

When FOV is divided by the average FPS, a value for the viewing angle per frame is obtained. If the variation of the position per frame is larger than the viewing angle, it can be judged that the speed is fast. Conversely, if the variation of the position value per frame is smaller than the viewing angle, it can be judged that the speed is slow. After determining the speed, BlurStack and LODRestoreStack are used. It takes LOD step applied to the previous frame and accumulates the variation of the position per frame in two stacks according to the speed. While the variation of the position per frame is accumulated in the BlurStack, it is judged that the speed is high if the value continues for 10 frames or more. In order to reduce the performance, LOD level is lowered and the blur effect is applied at the same time. On the contrary, while the variation of the position per frame accumulates to the LODRestoreStack for more than 5 frames, it is determined that the speed is slowed, the LOD level is increased again, and the applied blur effect is removed. The reason for accumulating the stack over a certain frame is that if the LOD level is lowered even for small field of view movement, more operations occur and performance deteriorates. In addition, when LOD level is lowered and restored, the performance deteriorates momentarily. Therefore, it is possible to prevent the problem by adjusting LOD level in advance using the frame reference. The reason why the duration of the LODRestoreStack is set to 5 frames is that the motion of the field of view is predicted to be stopped when the LOD level is restored, momentary deteriorates of performance is prevented. This procedure is illustrated in Figure 5.

4. Experiment Results

This section is for the experimental result of the software proposed in this paper. The proposed mechanism is verified in the environment shown in Table 1 below. Table 2 shows the data used in this paper.

Table 1. Experimental Environment

Item	Description
CPU	Intel Core I5-3570 3.40Ghz
RAM	4GB DDR3
GPU	NVIDIA GeForce GTX 470
OS	Windows7
Graphic Library	OpenSceneGraph
Language	C++

Table 2. Experimental Data

Type	Description and value
Range	1. Yekcheon Airport 100km Radius 2. South Korea Area
Data Size	1. 3.1GB 2. 13.7GB
Vertices	1. 4,320,857 2. 23,427,133
LOD Level	LOD 3 Level
Max View Movement Range	1. Heading : 180° 2. Pitch : 90°
View Movement Speed	180° FOV (Max : 1000ms)

Experimental data used 100Km data around Yecheon airport and data from all over South Korea. The LOD level of the two data is applied to the 3-step LOD, and the maximum field motion range is 180 degrees heading and 90 degrees pitch. The visual field motion speed moved the movement of the 180 degree viewing angle up to 1 second. In addition, we used Heading and Pitch values according to the position information of the mouse pointer based on the pixel information of the screen in order to implement the visual field motion of the HMD. In this environment, Figure 6 shows a screen that implements the HMD simulation software proposed in this paper.

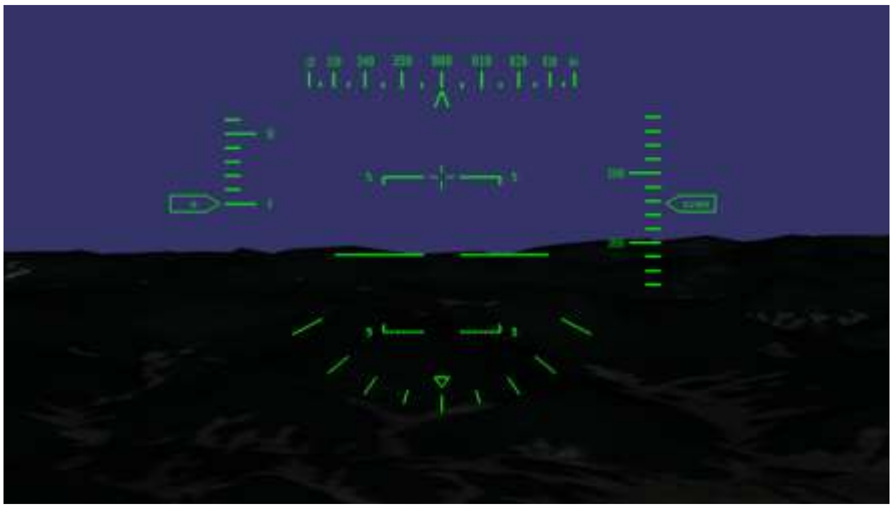
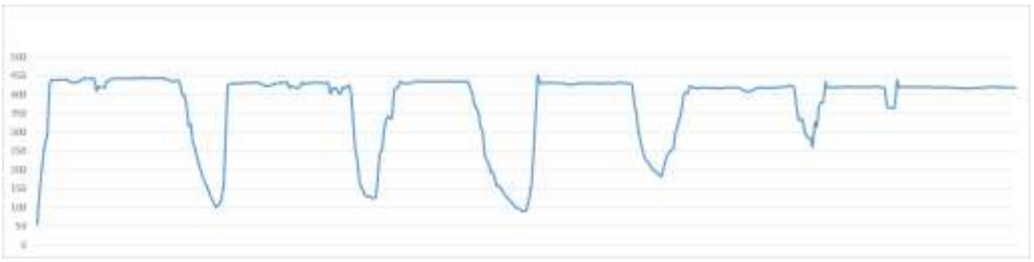
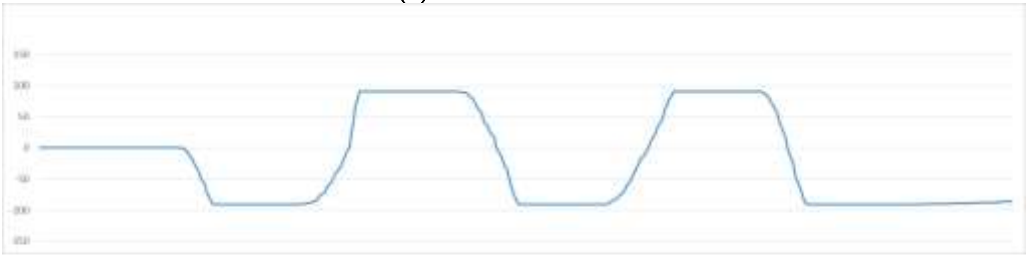


Figure 6. HDM Simulation Software

As shown in Figure 6, the proposed HMD simulation software renders terrain data and HMD symbols in real time using the OSG graphic library and displays them on the screen. In addition, the implemented HMD simulation software can set whether or not to display symbol according to the user's setting, and can edit color and position. The proposed FPS of the HMD simulation software showed a maximum of 450 FPS. Figure 7 and 8 show the performance results of the proposed optimization mechanism to improve the FPS, which decreases as the field of view moves faster.

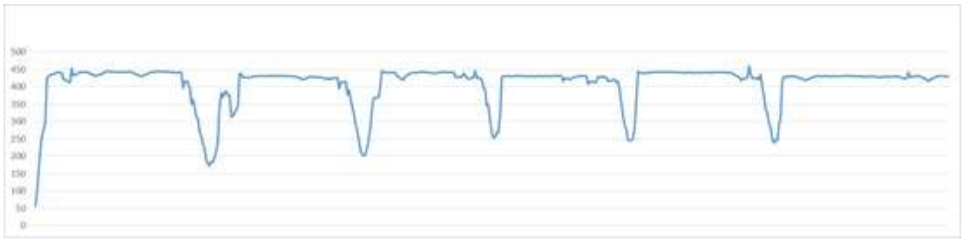


(a) Variation of FPS

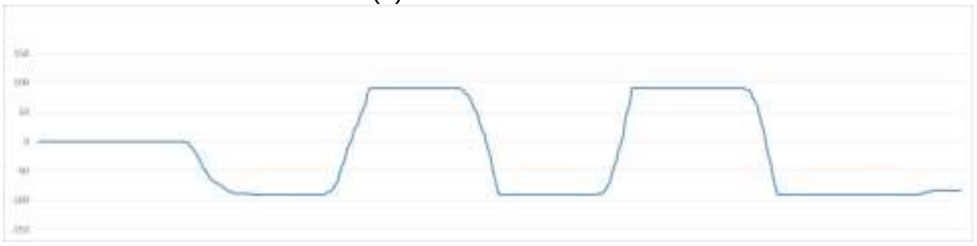


(b) Variation of FOV

Figure 7. Experimental Result before Using the Proposed Mechanism



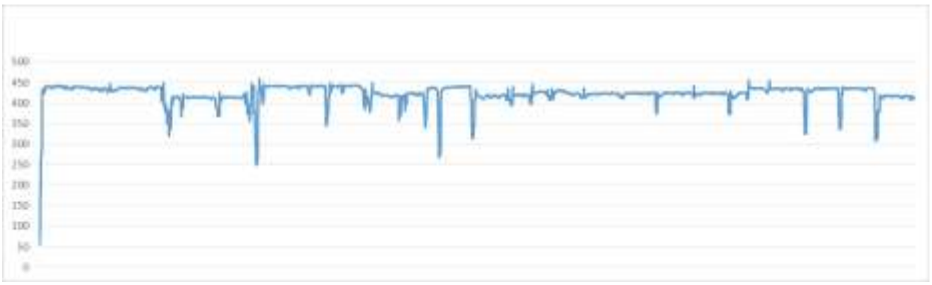
(a) Variation of FPS



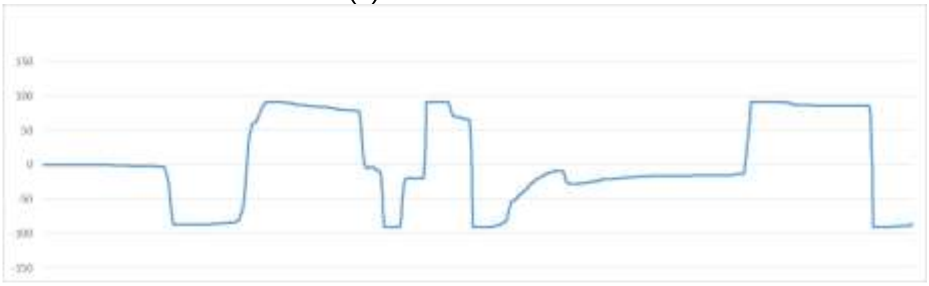
(b) Variation of FOV

Figure 8. Experimental Result after Using the Proposed Mechanism

In order to evaluate the performance of the mechanism proposed in this paper, we measured the FPS according to the LOD scale control technique proposed in Section 3 and the change of viewing angle before and after using the motion blur effect. As shown in Figure 7, the proposed method showed a maximum FPS of 450 FPS and a minimum performance of 100 FPS when the visual field motion was changed. This degrades performance because the data in the view-frustum for display on the screen at the time of rendering changes rapidly according to the visual field movement. However, with the proposed mechanism, the maximum FPS is the same, but the performance is improved by about 70% with the minimum FPS of 170 FPS. The minimum FPS is 170 FPS, which is about 70% better. The proposed mechanism showed better performance because the vertices included in the rendering operation decreased as the LOD scale changed. Figure 9 shows the experimental results of the mechanism for reconstructing the LOD scale when the change in the field of view suddenly decreases.



(a) Variation of FPS



(b) Variation of FOV

Figure 9. Experimental Results after Application of LOD Scale Restoration Mechanism

In the case of applying the proposed LOD scale restoration mechanism, we experimented by varying the motion speed of the field of view as shown in Figure 9 (b). As a result, there was a case where the FPS rapidly dropped. However, the FPS variation graph of Figure 9 (a) shows that range of rapidly dropped FPS variation is decrease compared to before the mechanism was applied. If the variation range of the FPS is large, the possibility that the screen is disconnected increases, which can give a visually unrealistic feeling to the user. As a result, it is confirmed that the proposed restoration algorithm provides a higher reality to the user. Finally, we measured the vertices information before and after the use of the proposed mechanism, and the measurement results are shown in Figure 10.

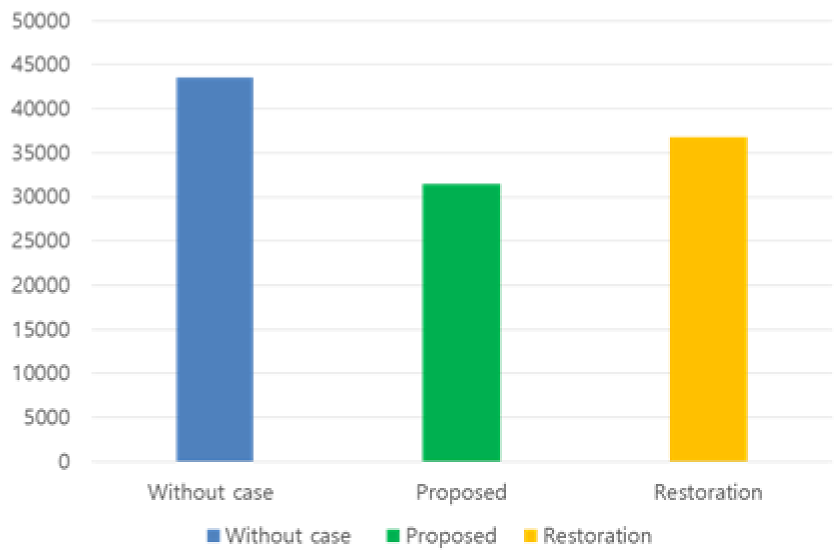


Figure 10. Variation of Vertices According to Mechanism Usage

As shown in Figure 10, the variation of Vertices decreased from 44000 to 31000 by using the LOD mechanism and increased to 36000 after adding the LOD Scale restoration mechanism. The restoration mechanism has increased the vertices compared to using the LODmechanism, but the reality of the visual effects seen by the user has increased. As the scale of the LOD changes, the size of the rendering data decreases and the vertices change.

5. Conclusion

In this paper, we proposed HMD simulation software for performance verification at the implementation stage before applying various equipment of aircraft. The proposed software is implemented as a real-time rendering system using the three-dimensional open source graphic library. In the real-time rendering system environment, maintaining of FPS is one of the important requirements. However, there is a problem that the FPS is rapidly deteriorated due to the rapid visual field change of the HMD. To solve this problem, we applied the technique of changing the LOD scale according to FPS. Also, we apply the motion blur effect to prevent the T-type facture phenomenon caused by the sudden change of the LOD scale and to improve the reality. To measure the real - time rendering performance of the proposed mechanism, we used the mouse pointer value to compute the LOS information and use the terrain data for the aircraft simulator. Experimental results of the proposed HMD simulation software show that the FPS is improved by about 70% before and after using the optimization technique and it is possible to test the performance before the HMD development.

However, since the proposed HMD simulation software is based only on topographic data, future studies will employ environmental changes such as meteorological phenomena. In addition, we will study the method to prevent the performance degradation in advance by using the prediction algorithm of the vertices information in View-Frustum instead of using the FPS of the previous frame when restoring the LOD scale of the optimization mechanism.

Acknowledgements

The work was supported by the Korea Evaluation Institute of Industrial Technology (KEIT) under “Avionics Integration Software Development for Supersonic trainer (10059058)” funded by the Ministry of Trade, Industry & Energy.

References

[1] S. Nozomi and T. Komuro, “Dynamic 3D interaction using an optical See-through HMD”, International Conference on Virtual Reality, Arles, France, **(2015)**.

[2] A. Basu, C. Ball, B. Manning and K. Johnsen, “Effects of user physical fitness on performance in virtual reality”, Symposium on 3D User Interfaces(3DUI). Greenville, USA, **(2016)**.

[3] F. Biljecki, H. Ledoux, J. Stoter and J. Zhao, “Formalization of the level of detail in 3D city modeling”, Journal of Computers Environment and Urban Systems, vol. 48, **(2014)**, pp. 1-15.

[4] W. Ying and W. Yanjie, “Implementation of a fast simulation algorithm for terrain based on Dynamic LOD”, International Conference on Electronic Measurement & Instruments (ICEMI), Chengdu, China, **(2011)**.

[5] M. Duchaineau, M. Wolinsky, D. E. Sigeti, M. C. Miller, C. Aldrich and M. B. Mineev-Weinstein, “ROAMing terrain: real-time optimally adapting meshes”, Proceedings of the 8th conference on Visualization. Los Alamitos, USA, **(1997)**.

[6] G. Bao, H. Li and X. Zhang, “Large-scale forest rendering: Real-time, realistic, and progressive”, Journal of Computer & Graphics, vol. 36, no. 3, **(2012)**, pp. 104-151.

[7] M. M. Wloka and R. C. Zeleznik, “Interactive real-time motion blur”, Journal of Visual Computer, vol. 12, no. 6, **(1996)**, pp. 283-293.

[8] R. Wang and X. Qian, “OpenSceneGraph 3.0 Beginner’s Guide”, PACKT, Birmingham, **(2010)**.

[9] R. Wang and X. Qian, “OpenSceneGraph 3.0 Beginner’s Guide”, PACKT, Birmingham, **(2010)**.

[10] R. Wang and X. Qian, “OpenSceneGraph 3.0 Beginner’s Guide”, PACKT, Birmingham, **(2010)**.

[11] About features, OpenSceneGraph, http://www.openscenegraph.org/index.php/about/features

Authors

JuHo Lim, received the B.S degrees in informatics from Gyeongsang National University. He is currently working toward M.S. degree at Gyeongsang National University. His research interests include Flight Simulation Software.

Chung Jae Lee, received the B.S. degree in avionics and simulation from Hanseo University and M.S. degree in aerospace engineering from Gyeongsang National University. He is currently working toward Ph.D. degree at Gyeongsang National University. His research interests include Flight Simulation Software.

Kyong Hoon Kim, received his B.S., M.S., and Ph.D. degrees in Computer Science and Engineering from POSTECH, Korea, in 1998, 2000, 2005, respectively. Since 2007, he has been an associate professor at the Department of Informatics, Gyeongsang National University, Jinju, South Korea. From 2005 to 2007, he was a post-doctoral research fellow at CLOUDS lab in the Department of Computer Science and Software Engineering, the University of Melbourne, Australia. His research interests include real-time systems, Cloud computing, and security.

Ki-Il Kim, received the M.S. and Ph.D. degrees in computer science from the ChungNam National University, Daejeon, Korea, in 2002 and 2005, respectively. He is currently with the Department of Computer Science and Engineering at Chungnam National University. His research interests include routing for MANET, QoS in wireless network, multicast, sensor networks and avionics software.