International Journal of Engineering & Technology, 7 (2.33) (2018) 359-363

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Level of detail parent and dynamic culling scheme for flight simulator

ChungJae Lee [1], KyongHoon Kim [2], Ki-Il Kim [3] *

[1] LIG System Co., LTD, Seoul, Korea
[2] Department of Informatics, Gyeongsang National University, Jinju 52828, Korea
[3] Department of Computer Science and Engineering, Chungnam National University, Daejeon 34134, Korea
**Corresponding author E-mail: kikim@cnu.ac.kr*

Abstract

Background/Objectives: Due to huge volume of data, it is required to reduce the number of vertices in flight simulator through level of detail and culling. However, it is required to integrate them accordingly.
Methods/Statistical analysis: Due to high point of view, terrain data in wide area are usually rendered in the flight simulator. So, conservational level of detail and culling schemes have adaptation problem in flight simulator. To defeat this problem, dynamic culling scheme and level of detail parent method based on line of sight are proposed.
Findings: Experiment scenarios are built to measure and compare the frame per second and number of vertices in four separate schemes, that is, scheme without the proposed algorithm, applying view frustum culling and applying culling and level of detail together and graphic processing unit based parallel processing algorithm, respectively. The proposed scheme reveals the similar values in the case of small-volume of terrain data. On the other hand, frame per second is significantly improved in huge volume of terrain data by reducing the number of vertices through dynamic culling and level of detail parent method. Specially, when level of detail parent method is applied to rendering the city with large number of buildings, the proposed scheme reveals the best frame per second among the comparative schemes.
Improvements/Applications: Improved rendering algorithm was proposed to handle huge volume of terrain data and thereby to prove the applicability of the proposed scheme in other simulator.

Keywords: *Flight Simulator; Culling; Level of Detail Parent; Frame per Second; Real-Time Rendering*

1. Introduction

The major objective of flight simulator is to train the pilot in efficient way as well as to reduce the complexity for the aircraft design. The flight simulator mainly consists of aero dynamic model, motion system and aircraft display system. Specially, aero dynamic model generates aircraft state information according to user input as well as actual dynamic model of aircraft. Thus, various motion and display systems are implemented according to development objectives and cost. Especially, display system is to provide realistic scene to pilot through satellite image data and altitude information. This process called real-time rendering makes use of graphic library to compute data and pass them to graphic hardware. In this process, huge amount of vertices are rendered in the flight simulator.
Thus, a flight simulator suffers from computational overhead to render the data. To solve this problem, most of flight simulator employs the specific method to reduce the rendering data in efficient way. Two famous methods, Level of Detail (LOD) [1] and culling scheme [2], have been proposed but they work in completely different way. The former is to reduce the amount of data by introducing multiple levels to represent the object. Therefore, the closer objective locates more detailed description it has. By this way, many details are not processed for the object in long distance. On the other hand, the latter is to reduce the amount of rendering data which is not visible from the point of view. It just allows rendering the visible area according to the current viewpoint. These two methods usually operate separately.
Unlike the other simulators, which employ LOD and culling scheme for real-time rendering, one of the unique properties of flight simulation is higher point of view than other simulators. This implies that wider area is usually rendered in flight simulator. On the other hand, line of sight is completed different from one in other simulators. So, it is required to improve LOD and culling scheme based on these properties.
Based on above analysis and demands, in this paper, we propose a new scheme to improve real-time rendering which makes use of dynamic culling scheme with LOS information according to altitude and LOD parent scheme in urban area. The former is to optimize the rendering area as the point of view moves while the latter is to merge the separate objects in accordance with LOD scheme. Specially, many buildings in urban area is specially suitable for LOD parent model. Finally, we present the experimental results to prove the performance improvement in the point of Frame Per-Second (FPS) and number of vertices.
The rest of this paper is organized as follows. Following by the introduction, we describe the fundamentals of LOD and culling scheme. The proposed scheme is explained in the following section. The experimental results are presented and analyzed in the next section. Finally, we make a conclusion and present the further issues.

2. LOD and culling

As briefly explained before, LOD and culling are the basic operations to improve rendering speed in flight simulator. Thus, there is some research work related to LOD and culling scheme.
First, LOD is used to represent the details of object according to distance between viewpoint and itself. It is categorized into static and dynamic LOD. In static LOD scheme, it adjusts the details of object according to distance which is made in advance. To be detailed, multiple meshes are created in advance. And then, each mesh is replaced by others according to distance between object and camera. In this scheme, there is memory overhead because each mesh should be loaded into memory. Therefore, static LOD is usually used to represent small objects rather than large ones. On the other hand, dynamic LOD adjusts the number of vertices in accordance with distance between object and camera. Based on this operation, its memory usage is less than static one. So, dynamic LOD is usually used to represent large object. In addition to two types of LOD scheme, there are some enhanced LOD algorithms to improve the performance. They includes LOD scheme based on distance[3] and fuzzy LOD [4].Second, culling scheme aims at excluding invisible area in rendering operation and is categorized into view-frustum [5], back-face[6] and occlusion culling [7]. View-frustum culling in figure 1 sets view-frustum with field of view (FOV) and renders this area. Back-face culling excludes rendering area which locates behind mesh while occlusion culling is based on distance. In occlusion culling, visible object called occlude and invisible one called occludee is categized and then occludee is excluded in the rendering. Related to culling, culling scheme based on Z-Buffer [8] is proposed to reduce the time to conduct z-test. Moreover, selective z-test architecture is proposed to reduce the data size in pipeline between graphic processor and memory. Also, H/W occlusion culling [9] test visibility on CPU rather than CPU by creating occlusion map.

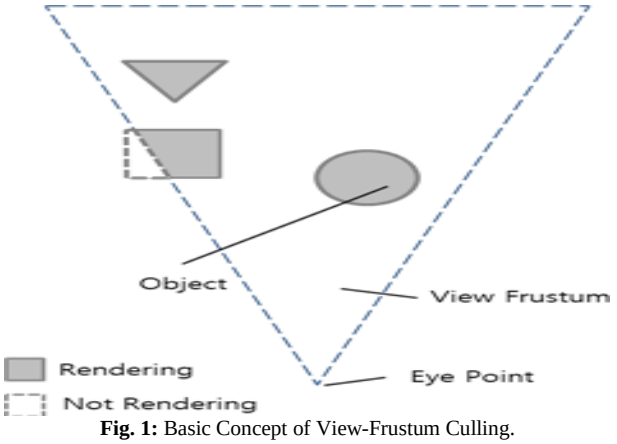

Fig. 1: Basic Concept of View-Frustum Culling.

Unlike the previous works which apply LOD and culling separately, there is research to combine two schemes. As good example, C. Peng and Y. Cao [10]propose how to integrate LOD parallel processing based on GPU and occlusion culling. In the proposed scheme, two steps are taken for efficient rendering. One is to compute visibility and the other is to apply it to bounding box.
Based on analysis above, since occlusion culling consumes additional resource to determine the visuality, it is not suitable for flight simulator. Rather, back-face culling is regarded as more suitable one for flight simulator. Moreover, it is required to apply back-face culling according to surrounding environment. The most important feature in flight simulator is that there is not too much difference two data, before visuality test and after one. So, it is not possible to achieve performance improvement.

3. Proposed scheme

In order to integrate the proposed LOD parent and dynamic culling scheme, we need new flight simulator to employ these functions as well as host system to control the flight simulator. The proposed simulator architecture is shown in figure 2. As you can see in figure 2, two system operates and communicate to exchange rendering state and data. The proposed scheme is implemented in optimized controller in right side. Also, real-time rendering software is implemented over OpenSceneGraph (OSG).

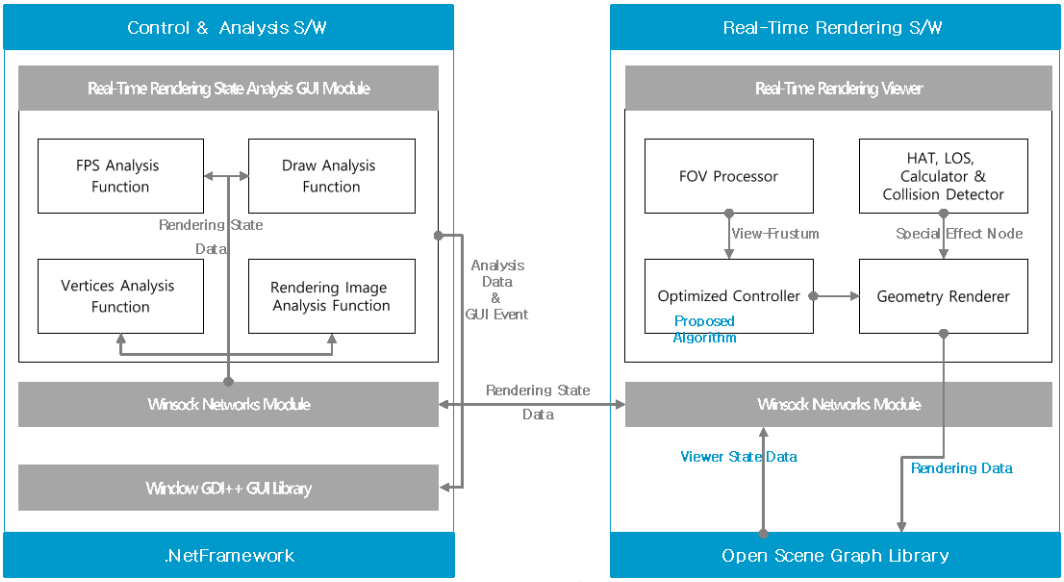

Fig. 2: Architecture of Simulator.

3.1. Dynamic culling and LOD scale algorithm through LOS

Usually, various situations such as low altitude flight, approaching the land and take-off needs special rendering process since the distance between terrain and viewpoint is too close. However, previous simulators do not take this situation into account by adjusting resolution of terrain according to predetermined LOD.

Moreover, view-frustum culling scheme is only applied while considering high altitude. This approach is able to maintain the similar FPS but hard to achieve realistic environment. To solve this problem, we apply different scheme according to altitude. Figure 3 shows the view frustum according to altitude. Since view frustum is affected by eye point greatly, it is very import to define the area according to eye point accordingly. Therefore, low and high altitude is determined by line of sight (LOS).

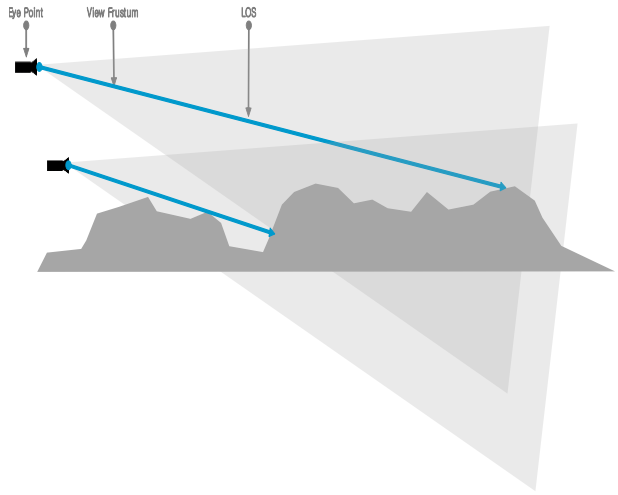

Fig. 3: View Frustum According to Altitude.

At the low attitude, the scene behind the viewpoint is not visible. So, it is better to apply back-face culling rather than view frustum one. On the other hand, back-face culling is not suitable at the high altitude since it cause high computational overhead.

For the LOD scheme, dynamic LOD is employed. However, since most of previous work fix the LOD level, they implement LOD without considering altitude. To solve this problem, we propose to set the LOD level with LOS information. So, we compute the average of multiple LOS values and adjust LOD level with LOD scale parameter accordingly. Figure 4 shows the detail procedure to optimize the rendering process through culling and LOD. From line 1 to 14, we compute the LOD value as much as LOS_count. According to this value, LOD scale is adjusted accordingly. The latter part, culling_mode_set, is to determine the culling area in the zBuffer.

Rendering Optimaization Algorithm

Require : terrainNode
Require : aircraft_state_information {from host }

```
1:  for i = 0 to rendering_Time do
2:    for i = 0 to LOS_count do
3:      X_LOSPoint[i] = (tan(pitch) * Altitude * sin(yaw)
4:      Y_LOSPoint[i] = (tan(pitch) * Altitude * cos(yaw)
5:      function Coordinate_Transform (aircraft_state_information)
6:      function Coordinate_Transform (LOSPoint)
7:      Dis_LOS[i] ← function Calculate_Two_Point
8:      if Dis_LOS[i] < Pre_LOS[i] then
9:        function culling_mode_set()
10:     else
11:       continue
12:     endif
13:     TotalDistance_LOS = TotalDistance_LOS + Distance_LOS[i]
14:   endfor
15:   Average_LOS = TotalDistance_LOS / i
16:   LODScale = Average_LOS
17:   LODScale set
18: endfor
```

culling_mode_set ()

```
1:  function Create_zMap (zBuffer, zMapSize)
2:  zMap send to VGA
3:  if occludee_Depth > occluder_Depth then
4:    depth ← occludee_Depth
5:    if depth = 1 then
6:      traingle is visibile
7:    else
8 :     continue
9 :   endif
10: else
11:   triangle culling
12: endif
```

Fig. 4: Algorithm for Rendering Optimization.

3.2. LOD parent algorithm for urban area

In the proposed scheme, we propose to reduce the number of rendering object. In order to reduce the number of object, we list the location information for urban area and then apply LOD parent algorithm if the urban area is included in the view frustum. By the help of this scheme, we can improve rendering procedure by merging multiple objects to one. This implies that multiple mesh information is replaced by one texture in OSG. To determine whether urban area is included in view frustum, LOS information is also computed. Figure 5 shows the detail procedure to apply LOD parent algorithm for urban area.

In figure 5, we repeat the computation for distance between aircraft location to LOS. Since the view frustum is affected by heading information, it is computed by rotation-translation equation between two points. Based on distance between this endpoint of view frustum and listed urban area, we extract the root node by removing the child nodes in the OSG tree. In this way, the child node is not rendered so the number of vertices is greatly reduced.

4. Performance evaluation

In order to evaluate the performance of the proposed flight simulator, we implement it through OSG. The flight simulator is written by C# with terrain data and control software. The rendering and control software exchanges data through UDP communication. The implemented software is evaluated in the following configuration as shown in table 1.

Table 1: Experimental environment

CPU	Intel i5-3570 CPU 3.4GHz
VGA	Nvidia GeForce 6GB
Memory	24GB
OS	Window 10 pro
Graphic library	OpenSceneGraph 3.4.0

For the simulation scenarios, two different terrain data sets are tested. One is 1.78 GB with 128,657 vertices and the other does

137 GB with 1,585,574 ones. It also includes urban area information which is 249Mb with 48,657 vertices.

Rendering Optimaization Algorithm 2

Require : UrbanNode
Require : aircraft_state_information {from host }
Require : UrbanList
Require : replacementNode

```
1:  for i = 0 to rendering_Time do
2:      P_u[i] ← function Calculate_Two_Point
3:      if P_u[i] < P_c then
4:          if P_u[i] < P_L3 then
5:              for j = 0 to Urban_Node_List do
5:                  if UNL[j] = UNL_Max then
6:                      UNL_Max ← removeChild()
7:                      UNL_Max ← changeNode(replacementNode)
8:                  else
9:                      continue
10:                 endif
11:             endfor
12:         else
13:             continue
14:     else
15:         continue
16: endfor
```

Fig. 5: Lod Parent Algorithm for Urban Area.

4.1. Experimental results

We measure the FPS for each scenario and illustrate the result in figure 6. As you can see in figure 6, the proposed scheme shows FPS ranged from 1,300 to 1,400. Also, we can observe 52,000 vertices in the proposed scheme.

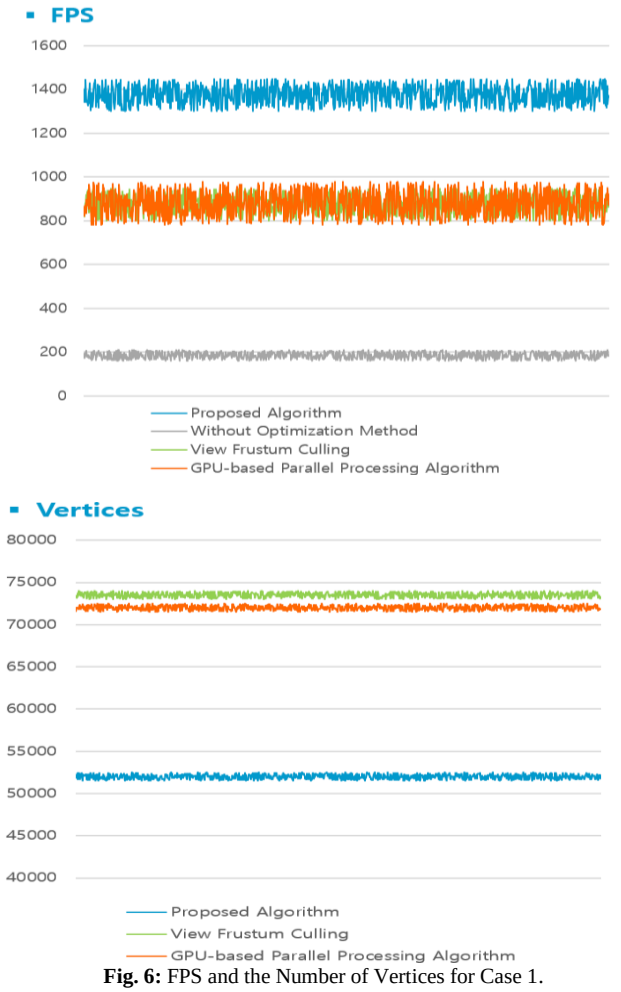

Fig. 6: FPS and the Number of Vertices for Case 1.

On the other hand, GPU based parallel processing scheme shows the similar performance to view frustum culling. Moreover, worst performance is shown in the original scheme without rendering optimization. This implies that the proposed scheme reduces the number of vertices at the low altitude significantly. Moreover, our scheme based on LOS leads to more accurate decision for visibility.

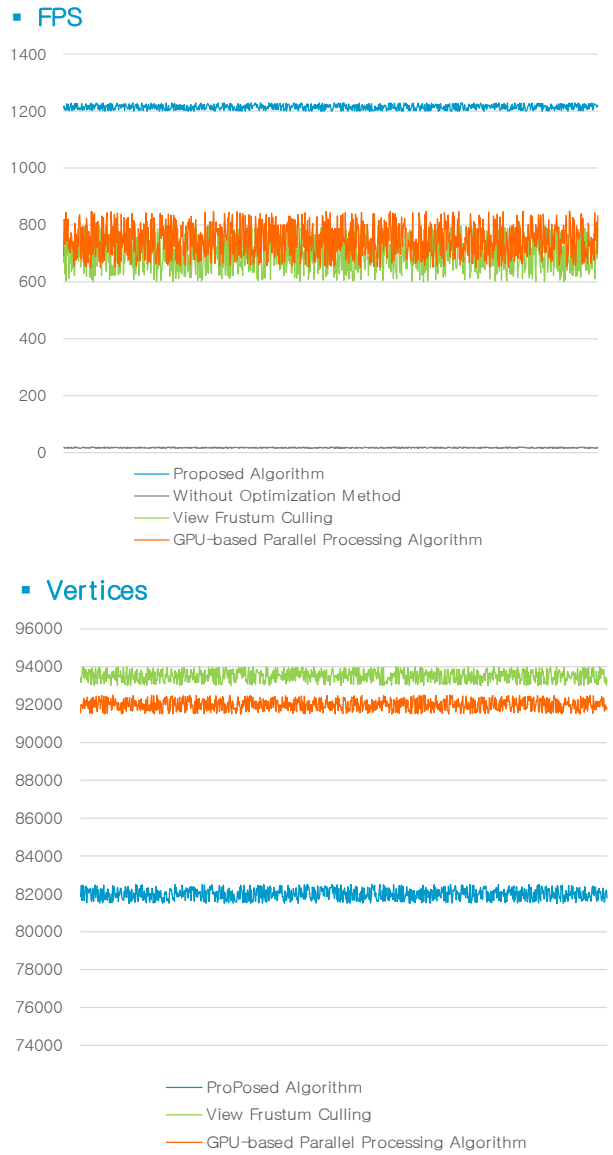

Fig. 7: FPS and the Number of Vertices for Case 2.

As compared to case 1, the proposed scheme shows the number of vertices around 82,000 and 1,200 FPS in figure 7. On the other hand, GPU based parallel processing shows lower performance than view frustum culling. Among the four schemes, the proposed scheme shows the best performance. The proposed scheme can reduce the number of vertices at the low altitude rather than GPU based parallel processing. It is resulted from the accuracy of visibility in the proposed scheme. Through this experimental result, we prove that occlusion culling is not suitable for flight simulator. Also, the proposed scheme shows the better performance than view frustum culling by applying two schemes according to LOS. Another factor to improve the performance is LOD parent algorithm for urban area. Due to drastic reduction of vertices for urban area, the FPS is in greatly improved.

5. Conclusion

This paper proposed real-time rending algorithm to employ LOD and culling scheme according to LOS information. Unlike the

previous scheme, the proposed scheme can reduce the number of vertices based on more accuracy for rendering area and dynamic LOD scale. Specially, the number of vertices is significantly reduced due to viewpoint. The improved performance is observed by the experimental result for two terrain set.

Acknowledgment

This work was supported by Human Resources Program in Energy Technology of the Korea Institute of Energy Technology Evaluation and Planning (KETEP), granted financial resource from the Ministry of Trade, Industry & Energy, Republic of Korea. (No. 20174030201440).

References

[1] Heok, T., & Daman, D. (2004). A review on level of detail. Paper presented at the Proceedings of the International Conference on Computer Graphics & Imaging and Visualization, 70-75.

[2] Zhoul, S., Yoo, I., Benes B., & Chen, G. (2014). A hybrid level-of-detail representation for large-scale urban scenes rendering. *International Journal of Computer Animation and Virtual Worlds*, 25(3), 243-253.

[3] Bao, G., Li, H., Zhang, X., & Dong, W. (2012). Large-scale forest rendering: Real-time, realistic, and progressive. *Computer & Graphics*, 36(3), 140-151.

[4] Schlender, D., & Peters, O. (2000). Managing levels of detail with fuzzy control. *Journal of Computer and Graphics*, 24(2), 245-251.

[5] Shahrizal, S., Zin, A., &Sembok, T. (2008). Improved view frustum culling technique for real-time virtual heritage application. *International Journal of Virtual Reality*, 7(3), 43-48.

[6] Kumar, S., Manocha, D., Garrett, B., & Lin, M. (1996). Hierarchical back-face culling. Paper presented at the Proceedings of Eurographics Workshop on Rendering, 235-254.

[7] Staneker, D., Bartz, D., &Straßer, W. (2004). Occlusion culling in OpenSG PLUS. *International Journal of Computers & Graphics*, 28(1), 87-92.

[8] Park, J., Kim, I., Park, W., Park, Y. & Han, T. (2013).A pixel pipeline architecture with selective z-test scheme for 3D graphics processors. *Microprocessors and Microsystems*, 37(3), 373-380.

[9] Biljecki, F., Ledoux, H., Stoter, J. & Zhao, J. (2014).Formalisation of the level of detail in 3D city modelling. *Journal of Computers Environment and Urban Systems*. 48, 1-15.

[10] Peng, C., & Cao, Y. (2014). Integrating occlusion culling with parallel LOD for rendering complex 3D environments on GPU. Paper presented at the Proceedings of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, 187-195.