International Journal of u- and e- Service, Science and Technology
Vol.8, No. 7 (2015), pp.151-160
http://dx.doi.org/10.14257/ijunesst.2015.8.7.15

Dynamic Culling Scheme for Graphic Rendering System of Flight Simulator

Chung Jae Lee[1], Kyoung Choon Park [2] and Ki-Il Kim[1*]

[1]Department of Informatics, Gyeongsang National University, Engineering Research Institute, Jinju, Korea, 660-701
[2] Aero Master Corporation, Sacheon, Korea
kikim@gnu.ac.kr

Abstract

Recently real-time overhead processing technology has received great attention from many researchers since it is expected to play a great role in improving frame per second. Also, this technology is suitable for flight simulator where large number of vertices as well as high resolution image is demanded. Even though typical culling or level of detail can be applied to flight simulator, more deep consideration is required because flight simulator operates in three dimensional space in the air. Among the possible technologies, view-frustum culling is one of key research area by improving rendering speed. But, despite of importance, since unnecessary data processing remains, it is reasonable to execute culling technology by adding the several elements such as aircraft altitude information, pitch angle dynamically. To develop a new dynamic culling, in this paper, we propose hybrid dynamic culling scheme by combining the view-frustum and the back-face culling together. The proposed culling scheme uses the fuzzy logic to determine culling ratio based on aircraft state information for visibility test. Finally, we demonstrate that the proposed scheme shows faster rendering speed than current culling scheme through experimental scenarios which are implemented by open source Open Scene Graph graphic library.

Keywords: *flight simulator, culling, fuzzy logic*

1. Introduction

Generally, Frame Per Second (FPS) of rendering system on the flight simulator should be maintained above the requirement for pilot training purposes [1]. However, number of vertices and rendering speed has inverse proportion to each other since the FPS decreases when the large number of vertices is required. Therefore, many research works in real-time rendering system have sought new overhead processing technique that improves realism as well as FPS. Specially, overhead processing of real-time rendering system have got much more attention from many researchers. Among them, culling and Level of Detail (LOD) are regarded as approach to figure out faced problem in rendering software with large scale terrain. Since both culling and LOD techniques decrease the amount of rendering data, it can make use of benefits obtained from CPU and VGA.

However, despite of above advantage, there are still too many research challenges not explored yet since the overhead processing technique in rendering system possesses significantly different properties to culling and LOD techniques. While the LOD is a mechanism for changing the accuracy of the object according to the specified distance between the current position and the object [2-3], culling approach excludes invisible region outside from Field of View (FOV).

As for former one, LOD is divided into static and dynamic one. Static LOD is a technique that renders a different accuracy depending on the object distance [4]. When small data is used, it can be expected to improve the FPS, however, fail in case of large terrain data. Thus, real-time rendering system requires a technique for solving these problems, known as dynamic LOD. Dynamic LOD is a technique that renders vertices to be divided by binary, quad and octa tree depending on the distance [5]. This technique is an efficient technique to improve the FPS when the terrain data of a large scale is assumed to render. Thus, LOD technique is effective scheme to improve the FPS by reducing the number of vertices, but invisible region is also rendered since it does not consider the FOV(Field Of View).

In order to maintain acceptable FPS without regard to data for changed FOV, culling approach can be another possible solution which is likely to be employed. Generally, culling scheme is classified into three techniques; view-frustum, back-face and occlusion culling. Among them, view-frustum culling is suitable for flight simulation because mainly used in the system to render a large scale terrain data. However, view frustum culling does not distinguish the occluders and occludees since it does not concern height of the FOV at the low altitude state. This can be main cause of reduced performance. In order to solve this problem, real-time rendering systems require adaptive culling scheme according to the height of the FOV.

Based on above analysis, selection of culling scheme for flight simulator requires deep thought. Since an aircraft changes their position and altitude, the culling scheme should be determined by the aircraft state information. However, when it comes to determine the culling scheme based on the absolute value, reasonable selection scheme is not achieved since small difference on absolute value make big difference. Thus, in this paper, we propose a dynamic culling technique to improve the FPS through fuzzy logic based on the height and pitch angle. In addition, we use the open source OSG graphic library to implement the proposed scheme under actual terrain image data. Briefly, the proposed scheme is compared with other culling schemes in terms of the number triangles and FPS.

The rest of this paper is organized as follows. In the Section 2, we describe the existing research work for the culling scheme on real-time rendering system. The proposed mechanism is explained in the Section 3. In the Section 4, details of the implementation and experimental results are presented in this paper. Finally, Section 5 presents the conclusion of this paper.

2. Related Work

In this section, we describe the existing research work on the culling processing technique, which is employed for three-dimensional image generating software. Culling technique used mainly in flight simulation system is view-frustum culling. This technique renders only data in the view-frustum [6]. This technique is known as easy to implement. In addition, this mainly uses in flight simulation system because is suitable to render the terrain of large size. Therefore, the study about this technique addresses how to compute the calculation speed and accuracy about data in view-frustum. However, since the view-frustum culling technique calculates whole data in view-frustum, it also renders the back-face of object. This problem should be resolved since this causes deterioration of the rendering system. In addition, since the FPS of view frustum culling is proportional to the size of the cube, it is not suitable at flight simulation because realism is proportional to the line-of-sight.

A technique to solve this problem is to back-face culling. It calculates the directional about each meshes data in view-frustum. After this, eliminates the back-face of object [7]. In addition, it reduces the number of vertices and textures in the rendering system where the height of the view is low. Enhance of FPS causes the improvement of efficiency of rendering system so realism is improved. However, this technique increases the unnecessary calculation in a small area such as inside a building since it includes the calculation about the occludees and occluders according to perspective.

A technique to address these issues is known as the occlusion culling. The occlusion culling divides the occludees and occluders according to calculation result later than calculates the perspective [8]. And then, the classified objects remove. There are two kinds of occlusion culling; software and hardware technique.

The software occlusion culling technique creates the depth buffer later than translates the scene to the quad. Also, it conducts the visibility test by CPU. Finally, CPU removes the occludes [9]. In addition, the software occlusion culling technique is conducted by the rendering and visibility test by CPU. For this reason, when use the terrain data of large scale, using the culling techniques occurs more overhead than before. The technique for solving the overload of the CPU is a hardware occlusion culling.

Hardware occlusion culling is not the scheme that it handles both the visibility test and rendering calculation by CPU, but CPU conducts the rendering calculation. GPU conducts the only visibility test. Therefore, since the CPU becomes available for another the calculation while the GPU is performing the visibility test, the improving of the rendering system is achieved [10]. The proposed scheme in [10] creates the volumes from objects in the view-frustum. After this, the rendering system sends them to GPU later than creates the depth buffers based on volumes. Finally, the GPU performs a visibility test using parallel processing based on the received depth buffer. In the process, a depth map creates with 4 by 4 sizes. After that, the generated result is transmitted to the CPU and classified by the occludees and occluders. Finally, this data excludes in the rendering calculation. The proposed technique improves the FPS of by about 20% as compared to the common software occlusion culling. By the help of this scheme, it shows almost similar results to GPU transfer scheme for visibility test that is proposed algorithm in [11]. However, differences between both can avoid occludees by using occlusion queries in order to find node in last rendering frame. Moreover, occludee nodes can avoid CPU stall and GPU starvation by rendering without waiting occlusion queries.

Thus, the occlusion culling technique improves the FPS in three-dimensional rendering system later than identify the occludees and occluders. As previously mentioned, the occlusion culling technique improves the FPS later than classifies the occludees and occluders in the three-dimensional rendering system. However, flight simulation is the boundary of the object disappears obscured with objects away because it operates at a high point. However, in flight simulation occludees and occluders disappear when it operates at the high height. Therefore, it handles the overhead better than using the view-frustum culling because increases the visibility test calculations.

In this way, the goal of overhead processing decreasing the vertices at the real-time rendering system is achieved. However, most of research work does not consider the changed view position properly. Thus, in this paper, we propose a new scheme to improve the FPS based on the changed view position rather than unconditional reduced vertices.

3. Proposed Mechanism

3.1. Decision for Culling Scheme Based on Fuzzy Logic

In this section, we describe the proposed culling scheme based on fuzzy logic. For the visibility test, we use the aircraft state information. In the process of determining the culling scheme, it is very important to classify the low or high altitude based on the aircraft state information. For example, when absolute value is low altitude, the culling scheme is changed because difference of only 1 feet. Therefore, it is necessary to determine a reasonable culling scheme. For this goal, we determine the culling scheme based on fuzzy logic. Fuzzy-logic has a contribution of each element belonging to the set where the elements belonging to the set of fuzzy logic represents the fuzzy membership function to decimals between 0 and 1.0 [12].

In this paper, we use the height and pitch angle information of the aircraft to determine the value of the culling scheme. Altitude information is used to classify the occludee and occluder in low-altitude case. In this paper, we classify the low-altitude which is landing altitude (4,000ft) at Yecheon airport. In this foundation, equation (1) calculates the fuzzy logic membership function values based on the height and pitch angle. In addition, respective fuzzy memberships are summarized in Table 1.

$$f(x) = x_{phase} - \frac{(h_i - h_{phase})(h_{max} - h_i)x_{offset}}{h_{offset}} \tag{1}$$

As given in equation (1), fuzzy membership function values are classified to four phase from 4000ft to 8000ft in 1000ft unit. Finally, membership function values are calculated according to the step-by-step offset as 10ft unit. Results of calculation, membership function values are changed the value per 1ft at 4000ft and becomes zero at 8000ft. In addition, if the pitch angle is 30 degrees, membership function values become 1 because the view is set towards the sky. The membership function values to determine culling scheme based on value is given by Table 1 and Figure 1.

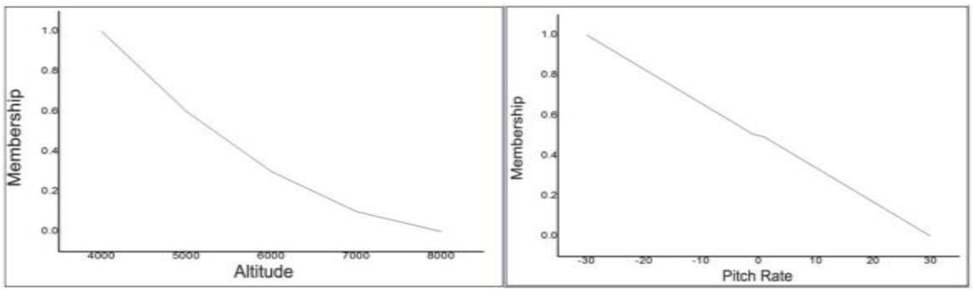

Figure 1. Fuzzy Membership Function (Top is Fuzzy Membership Function based on Altitude, Bottom is Fuzzy Membership Function based on Pitch Angle)

Table 1. Fuzzy Membership Based on Altitude

Altitude	Membership	Pitch Rate	Membership
4,000ft	1.0	30	0
4,010ft	0.996	29	0.017
…	…	…	…
5,000ft	0.600	…	…
5,010ft	0.597	…	…
…	…	1	0.493
6,000ft	0.300	0	0.5
6,010ft	0.298	-1	0.507
…	…	…	…
7,000ft	0.1	…	…
7,010ft	0.099	…	…
…	…	…	…
7,990ft	0.001	-29	0.983
8,000ft	0.0	-30	1.0

As shown in Figure 1, membership function is 1.0 in case of 4,000ft where the membership function increases according to the pitch angle and altitude. When the pitch angle becomes higher, the terrain of the scene decreases. Therefore, the membership function values are converged to the zero. Finally, the membership function values are used to determine the culling scheme.

$$u = \frac{\sum_{i=0}^{n} \lambda_i \mu_i u_i}{\sum_{i=0}^{n} \lambda_i \mu_i} \tag{2}$$

Equation (2) is used to determine the proposed culling scheme as D. schlender [13] proposed. u of Equation (2) denotes the distance between the view and object, λ_i refers to the weight of the membership function. For each rule of the rule base the degrees of membership are calculated and the minimum is taken and denoted as μ_i [13]. Distance between the field of view and object based on fuzzy sets are computed and used to select culling scheme. Thus, the culling scheme is determined by the equation (2). If the decision is set to the back culling and then, the size of the depth map is determined by the membership function values. The size of the depth map based on membership function values is shown in Table 2.

Table 2. Fuzzy Membership Based on Altitude

Membership	Depth Map Size
2.0 ~ 1.5	800 * 600
1.5 ~ 1.0	640 * 480
1.0 ~ 0.5	400 * 300
0.5 ~ 0.1	320 * 240

The size of the depth map becomes large when the membership function values increase. Similarly, it gets smaller when values decrease. The size of the depth map becomes larger when the membership function values increase. This depth map is delivered to the GPU in case of back-face culling. And then, it is used to test visibility.

3.2. Culling Scheme Decision based on Fuzzy-Logic

In this section, we propose the culling scheme and creation of dynamic depth-map by using value generated by membership function. The sequence of the proposed mechanism is shown in Figure 2.

Table 3. Required Data for Proposed Culling Scheme

Data	Explain
terrainNode	Terrain Object
zBuffer	Depth Buffer for Visibility Test
zMapSize	Size of Depth Map
zMapCameraNode	The Visibility Test Calculation
Aircraft_state_infomation	Received from Commercial Software
Triangles	Vertices in the Current Scene
Altitude_Phase, Max_Altitude	Altitude_Phase, Max_Altitude
Membership_Function_Phase	Membership_Function_Phase
Membership_Function and Altitude Offset	Membership_Function and Altitude Offset

In order to apply the culling scheme in real time rendering system, we require some data. Table 3 shows the related data. The proposed culling scheme demands z-map_camera, z-buffer, z-map_size information[14] in order to test visibility. In addition, we require the calculation result of membership function to determine culling scheme. It uses the aircraft state information that is received from other flight simulator.

If the calculation result of membership function is less than 0.1 when the aircraft state is the high altitude or high pitch angle state, the culling scheme applies the view-frustum culling. However, for the opposite case, dynamic culling scheme based on created depth map is applied. At this time, z-map_camera is not rendered because it is subordinate to the current rendering scene.

The created depth map is sent to GPU. After this, visibility test is performed with it. GPU performs the visibility test for all pixels by the size of the depth map and distinguish the back-face of object. If the result of visibility test is true, it is classified into the occulders. For the opposite case, the result of visibility test is classified into the occludees so it does not progress the rendering. Thus, performance is improved because occludees are removed from the real-time rendering system.

Proposed Algorithm

```
Require : terrainNode
Require : zBuffer
Require : zMapSize
Require : zMapCameraNode
Require : aircraft_state_information {from host }
Require : triangles { the amount of terrain vertices }
Require : altitude_Phase, altitude_Max
Require : membership_function_Phase
Require : membership_function and altitude Offset

function $aircraft\_state\_information\ (receive\ from\ host)$
function $Read\_zBuffer\ (aircraft\_state\_information)$
function $base\ (terrain\_vertices,\ altitude,\ pitch)$
  for $i = 0$ to $terrainNode$ do
    $Mem_{curr} = (alt_i - alt_{phase})(alt_{max} - alt_i)Mem_{offset}/alt_{offset}$
    $Mem_{result} = Mem_{phase} - Mem_{curr}$
    if $Mem_{result} = Mem_{range}$ then
      $base = 1$
    else
      $base = 0$
    endif
  endfor
if $base = 1$ then
  $culling\ scheme\ is\ dynmic\ culling$
  function $zMapSize\ (fuzzy\_membership)$
endif
for $i = 0$ to $zMapSizeX$ do
  for $j = 0$ to $zMapSizeY$ do
    $input\ zBuffer(i, j) \leftarrow pos$
  endfor
endfor
function $Create\_zMap\ (zBuffer,\ zMapSize)$
$zMap\ send\ to\ GPU$
if $occludee\_Depth > occluder\_Depth$ then
  $depth \leftarrow occludee\_Depth$
  if $depth = 1$ then
    $traingle\ is\ visibile$
  else
    continue
  endif
else
  $triangle\ culling$
endif
```

Figure 2. Proposed Algorithm

4. Implementation and Experiments

In this section, we present the implementation details and experimental results based on the mechanism proposed. For the experimental results, we use the OSG that it is open source and wildly used graphic library. OSG is based on scene graph theory under the hierarchical tree structure to manage the number of virtual and multimedia objects effectively.

We introduce the terrain data of 100km area around the Yecheon airport, Korea, in order to evaluate the performance of the proposed mechanism. Terrain data is composed to the DTED (Digital Terrain Elevation Data) and satellite image of 10m resolution. This terrain data is loaded according to defined method in osgDB class [15-16]. In addition, aircraft state information are received from the commercial flight simulation software [17].

Implementation and performance evaluation of the proposed scheme carries out in the environment of Table 4. As described in Table 4, real-time rendering system consists of the host software based on C# and the real-time rendering software based on C++. Method of communication between two software use the UDP and the communication period is set to 60Hz. In this environment, the experiment is to compare the three culling scheme; Case without culling, Case for Common View frustum culling, Case for proposed culling. Comparative information compares the triangles, the vertices and FPS according to the each altitude.

Table 4. Experimental Environment

OS	Windows 7 64bit
CPU	Intel i7 980 2.80GHz
RAM	8GB
VGA	nVidia GTX 470

We calculate the membership function based on the received aircraft state information while its result values are used to determine the culling scheme. Then, the rendering system will reduce the number of vertices in accordance with the corresponding culling scheme. So, it eventually enhances the performance of real-time system. The result of this process is shown in Figure 3.

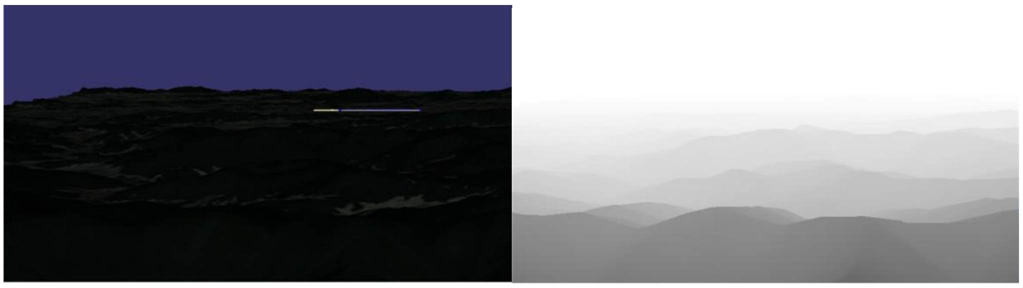

Figure 3. Z-Map and Rendering Image of Low-Altitude State

Figure 3 shows the rendered image and the depth map of the low-altitude state. The left side image of the Figure 3 represents the result of depth map in the low altitude state while the right image of the Figure 3 does the result of rendering processing in the same altitude state. At this time, the right side image of Figure 3 does not affect the performance of the entire system because it is not used by the visibility test in the rendering processing. As shown in Figure 3, the proposed culling scheme is performed the rendering process twice at the low-altitude state. First rendering process is performed at the subordinate z-map_camera in the current scene. The z-map_camera creates the depth buffer and used when creating the depth map. Thereafter, the depth map is sent to the GPU over the OSG library, and the GPU performs a visibility test based on the transmitted depth map. Finally, The GPU returns the occludees object and the back-face to the results value of visibility test, and GPU transmits the result to the CPU. CPU is removing the received result in the rendering processing so it renders the remaining objects. Figure 4 shows the rendered image result when use the three culling schemes.

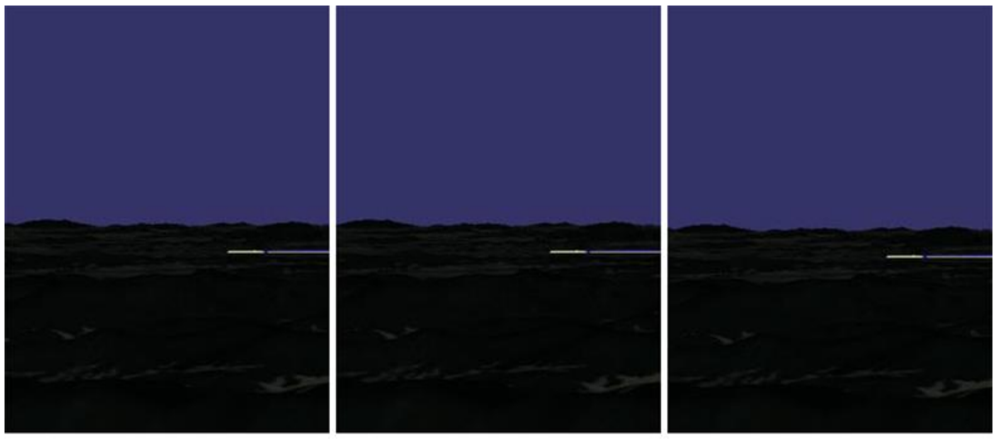

Figure 4. Real-Time Rendering Image Comparison of Culling Scheme

Table 5. Comparison of Altitude Based Culling Scheme

Altitude	Case Without Culling		
	Triangles	Vertices	FPS
2,000ft	11,079	204,100	16.53 fps
4,000ft	11,079	204,100	16.63 fps
6,000ft	11,079	204,100	16.58 fps
8,000ft	11,079	204,100	16.66 fps
Altitude	Case For Common View Frustum Culling		
	Triangles	Vertices	FPS
2,000ft	2,560	106,126	60 fps
4,000ft	2,634	107,310	58.12 fps
6,000ft	2,724	108,750	55.37 fps
8,000ft	2,854	110,980	50.72 fps
Altitude	Case For Proposed Culling		
	Triangles	Vertices	FPS
2,000ft	2,176	99,782	60 fps
4,000ft	2,304	101,654	60 fps
6,000ft	2,412	102,159	60 fps
8,000ft	2,526	104,256	60 fps

Figure 5 shows the rendering rate of the three culling schemes. In addition, the results of triangles, vertices and FPS about each altitude states in Figure 4 are shown in Table 5.

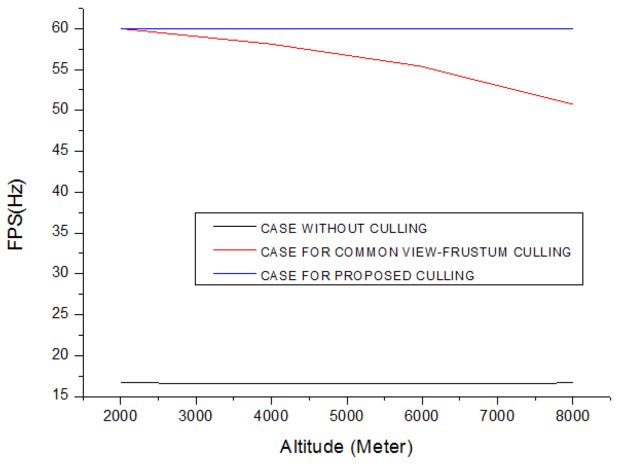

Figure 5. Compare to Real-Time Rendering Rate of Culling Scheme

In case of no culling scheme, since whole data are included in the rendering process, real-time rendering system operates irrelevant to the change of the altitude and pitch angle. Therefore, real-time rendering software will render all triangles and vertices of the terrain data. However, since only data in common in view-frustum is rendered, the triangles decrease by about 80% while the vertices do by about 50%. In addition, the proposed culling scheme decrease number of the triangles by about 15% and about 10% vertices rather than the common view-frustum culling scheme. Finally, FPS is improved by about more 73% than the case of without culling scheme. In addition, FPS is improved by more 8% than the common view-frustum culling. In this experimental result, since range of rendering becomes larger according to height of altitude, FPS of view-frustum culling scheme is reduced.

5. Conclusion and Further Work

As for research work in flight simulation, dynamic culling scheme study becomes essential procedure to improve the performance. It is possible to satisfy both the conservation of realism and the improving of FPS by the help of dynamic culling scheme. Based above possibility, we proposed the dynamic culling scheme by using fuzzy logic based on the information of aircraft state for flight simulator. In addition, the proposed scheme evaluated the performance to using actual terrain data.

However, our proposed algorithm uses only the altitude and pitch information in the calculation of fuzzy membership function. Therefore, we are scheduled to be extend current dynamic culling scheme by introducing other flight state information to get more accurate information. Also, according to the diverse inputs, more rational fuzzy logic will be devised.

Acknowledgements

This research was supported by SW Master's course of hiring contract program grant funded by the Ministry of Science, ICT and Future Planning and Production Technology Commercialization Support program by the Korea Industrial Complex Corporation.

References

[1] M. K. Zyskowski, and W. A. Redmond, “Aircraft simulation techniques used in low-cost, commercial software”, Proceedings of AIAA Modeling & Simulation Technologies Conference and Exhibit, **(2003)** August 11-14, Texas, USA.

[2] T. K. Heok and D. Daman, “An International Conference on Computer Graphics & Imaging and Visualization”, **(2004)** July 70-75.

[3] E. Walia and V. Verma, “A Computationally Efficient Framework for 3D Warping Technique”, International Journal of Computer Graphics, vol. 3, no. 1, **(2012)**, pp. 1-10.
[4] S. Zhou1, I. Yoo, B. Benes and G. Chen, “A hybrid level-of-detail representation for large-scale urban scenes rendering”, International Journal of Computer Animation and Virtual Worlds, vol. 25, no. 3-4, **(2014)**, pp. 243-253.
[5] J. Levenberg, “Fast View-Dependent Level-of-Detail Rendering Using Cached Geometry”, Proceedings of Visualization, **(2002)** November, pp. 259-265, Boston, USA.
[6] M. S. Sunar, A. M. Zin and T. M. T. Sembok, “Improved View Frustum Culling Technique for Real-Time Virtual Heritage Application”, International Journal of Virtual Reality, vol. 7, no. 3, **(2008)**, pp. 43-48.
[7] S. Kumar, D. Manocha, B. Garrett and M. Lin, “Hierarchical Back-Face Culling”, Proceedings of Eurographics Workshop on Rendering, **(1996)** December, pp. 235-254, Porto, Portugal.
[8] D. Staneker, D. Bartz and W. Straßer, “Occlusion culling in OpenSG PLUS”, International Journal of Computers & Graphics, vol. 28, no. 1, **(2004)**, pp. 87-92.
[9] “Software occlusion culling, Intel Corporation”, http://software.intel.com/enus/articles/software-occlusion-culling/.
[10] L. R. Barbagallo, M. N. Leone and R. N. Garcia, “Vertex Discard Occlusion Culling”, Proceedings of XVIII Congreso Argentino de Ciencias de la Computación, **(2013)** October, pp. 397-407, Bahía Blanca, Argentina.
[11] J. Bittner, M. Wimmer1, H. Piringer and W. Purgathofer, “Coherent hierarchical culling: hardware occlusion queries made useful”, International Journal of Computer Graphics Forum, vol. 23, no. 3, **(2004)**, pp. 615-624.
[12] L. A. Zadeh, “Fuzzy sets. International Journal of Information and Control, vol. 8, **(1965)**, pp. 338-353.
[13] D. Schlender and O. H. Peters, “Managing levels of detail with fuzzy control. Managing levels of detail with fuzzy control”, International Journal of Computers & Graphics, vol. 24, **(2000)**, pp. 245-251.
[14] C. Lee and K.-I. Kim, “Fuzzy Based Culling Scheme for Flight Simulator”, International Conference on Computer & Information and Application, **(2015)** May 22-25, Yeosu, Korea.
[15] R. Wang and X. Qian, “OpenSceneGraph 3.0 Beginner’s Guide”, PACKT, Birmingham, **(2010)**.
[16] R. Wang and X. Qian, “OpenSceneGraph 3.0 Cookbook”, PACKT, Birmingham, **(2010)**.
[17] “ESP Simconnect SDK”, Intel Corporation, https://msdn.microsoft.com/en-us/library/ cc527012.aspx/.

Authors

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.

Kyoung Choon Park, received the M.S. degree in computer science from Gyeongsang National University. He is currently working at Aero Master Corporation. His research interests include avionics software, software engineering and signal processing.

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 Informatics at Gyeongsang National University. His research interests include routing for MANET, QoS in wireless network, multicast, sensor networks and avionics software.