논문 2015-10-10

고도에 따른 렌더링 시스템을 위한 동적 컬링 방안

(Dynamic Culling Scheme Based on Altitude for Real-Time Rendering System)

이 충 재, 강 석 윤, 김 기 일*
(Chungjae Lee, Seokyoon Kang, Ki Il Kim)

Abstract : Dynamic culling scheme is usually implemented to handle overhead caused by rendering the massive large-scale terrain data in flight simulator. However, existing culling scheme without considering altitude is not suitable for flight simulator due to additional computational overhead. To solve this problem, in this paper, we propose hybrid approach by applying two dynamic culling schemes depending on altitude. In addition, we remove unnessary computational overhead by creating different z-map resolution when aircraft changes its altitude. The proposed scheme is implemented with open graphic library and tested with real terrain data. Through the experimental results, we can recognize the improved rendering speed about 8 to 73 percents as compared to existing scheme.

Keywords : Culling, Flight simulation, Open scene graph

Ⅰ. 서론

항공기 시뮬레이터의 영상 발생 장치 소프트웨어는 사용자에게 3차원 영상을 표시하여 비행 훈련 효과를 증가시킨다. 이를 위해 높은 해상도의 고도 데이터 및 위성영상을 기반으로 제작된 지형을 사용하거나 각종 특수 효과를 사용한다[1]. 하지만 고해상도의 지형을 화면에 출력하는 과정에서 그래픽 하드웨어의 성능에 따라 초당 화면 갱신율이 변화하지만 그래픽 하드웨어의 경우 처리하는 속도에 한계가 있기 때문에 이를 극복하기 위한 과부하 처리 기술의 연구가 진행 중이다. 이러한 영상 발생 장치 소프트웨어의 과부하 처리 기술은 대표적으로 LOD(Level Of Detail)와 컬링이 있다. 이 중 LOD는 렌더링 이전에 삼각형으로 이루어진 3차원 물체를 거리에 따라 다르게 표현하여 연산을 감소한다.

이러한 기법은 실시간 렌더링 시스템에서 시점에 따라 보이지 않는 영역도 렌더링 하여 불필요한 연산이 발생시키는 단점이 존재한다. 이와는 다르게 컬링은 시점에서 제외된 영역을 렌더링 시 제외함으로써 초당 화면 갱신율을 향상하는 기법을 말한다. 하지만 현재 자동차나 사람과 같은 지면에서의 움직임을 기반으로 하는 컬링 기법에 관한 연구는 존재 하지만 아직까지 3차원 움직임을 기반으로 하는 항공 시뮬레이션의 경우에는 시각 절두체 컬링 이외 기법을 사용하는 연구는 전무하다. 또한 항공 시뮬레이터는 항공기의 비행 상황에 대한 훈련을 목적으로 개발하기 때문에 높은 고도 상에서의 비행 상황만을 고려하는 컬링 기법이 대부분이다. 따라서 이, 착륙 및 낮은 고도에서도 비행 상황에 대한 컬링 기법 필요하다. 이러한 컬링 기법은 시뮬레이터 동작 시 지형 데이터의 해상도가 높을수록 증가하는 데이터의 연산을 감소할 수 있다. 또한 시뮬레이터 개발 비용은 한계가 있기 때문에 대용량의 지형을 사용하면서 동시에 일반적인 렌더링 하드웨어 상에서도 렌더링 속도의 보장이 가능하게 한다.

따라서 본 논문에서는 항공기 3차원 움직임을 기반으로 변화하는 시점에서 고도에 따라 동작하는 과부하 처리 방안을 제안한다. 또한 제안한 방안의

*Corresponding Author (kikim@gnu.ac.kr)
Received: 7 Oct. 2014, Revised: 10 Nov. 2014, Accepted: 30 Nov. 2014.
C. Lee, S. Kang, K.I. Kim: Gyeongsang National University, Engineering Research Institute
※ 본 논문은 미래창조과학부의 고용계약형 SW석사과정 지원사업을 지원받아 수행한 연구결과임

 2015 Apr. 10(2) 73-79
ISSN : 1975-5066
http://dx.doi.org/10.14372/IEMEK.2015.10.2.73

구현과 성능 평가를 위해 OSG (Open Scene Graph)를 사용하여 구현하였으며 실제 지형 데이터를 사용하여 컬링기법에 따라 삼각망의 수와 초당 화면 갱신율를 비교하였다. 본 논문의 구성은 다음과 같다. 1장은 서론이며 2장은 컬링에 관한 기존 연구를 소개한다. 3장은 제안 메커니즘이며, 4,5장은 실험 결과 및 결론을 소개한다.

Ⅱ. 관련 연구

과부하 처리 기술 중 대표적인 컬링 기법은 시각 절두체 컬링이다. 시각 절두체 컬링은 현재 시점을 기준으로 육면체를 생성하여 이를 시각 절두체라 하며 시각 절두체 내부의 데이터만 화면에 표시하는 기법이다. 이처럼 시각 절두체 컬링은 육면체 내의 데이터를 얼마나 빠르고 정확하게 선별하는 알고리즘을 기반으로 한다[2]. 이러한 기법은 구현이 간단하고 성능이 뛰어나 실시간 렌더링 시스템에서 많이 사용하지만 시각 절두체 내부의 모든 데이터를 연산하여 렌더링 하므로 지형을 이루는 삼각망의 수가 많은 경우 연산을 줄이기 위해 육면체의 크기를 감소해야한다. 이때 육면체가 감소하는 비례에 따라 가시거리 또한 감소하여 현실감 저하라는 문제점이 발생한다.

이러한 단점을 보안한 컬링 기법을 후면 컬링이며 이는 현재 시점에서 물체의 후면을 렌더링 연산에서 제거함으로써 시뮬레이션의 전체 성능을 향상시킨다[3]. 하지만 후면 컬링은 물체의 후면만 제거할 뿐 시점 앞에 위치한 물체에 의해 가려지는 물체를 고려하지 않아 불필요한 연산이 증가한다.

따라서 이러한 문제점을 보완하는 것을 차폐 컬링이라 하며, 이는 소프트웨어와 하드웨어 차폐 컬링으로 분류한다. 이 중 전자는 현재 시점의 화면을 평면 쿼드로 변환 후 크기가 작은 깊이 버퍼를 통하여 CPU가 가려지는 물체와 가리는 물체에 대한 가시성 테스트를 수행하여 보이지 않는 물체를 컬링하는 기법이다[4]. 하지만 소프트웨어 차폐 컬링은 앞서 언급한 바와 같이 깊이 버퍼를 기반으로 CPU가 가시성 테스트를 수행하기 때문에 시스템의 오버헤드가 증가한다. 또한 항공기 시뮬레이션과 같이 대용량의 지형을 사용하는 경우에는 CPU가 지형 렌더링과 더불어 가시성 테스트도 수행하여 컬링을 적용하지 않는 경우보다 더 많은 오버헤드가 발생한다.

이와는 다르게 하드웨어 차폐 컬링은 CPU를 이용하여 가시성 테스트를 연산하는 방식이 아니라 GPU에게 전송하여 처리된 값을 기반으로 연산하기 때문에 CPU는 가시성 테스트 도중에 다른 연산이 가능하므로 전체 시스템의 성능을 향상할 수 있다[5]. [5]에서 제안하는 방식은 현재 시점에 가리는 물체를 선택 후 모든 가리는 물체에 대한 볼륨을 생성한다. 이 후 이러한 볼륨을 기반으로 깊이 버퍼를 생성하고 이를 GPU에게 전송하여 가시성 테스트를 수행하며 이 과정에서 여러 개의 병렬 처리 기법을 활용한다. 이는 4*4의 깊이 맵을 생성하고 각각의 분할된 타일에 대한 가시성 테스트를 수행한다. 최종적으로 CPU는 GPU가 수행한 가시성 테스트 값을 렌더링에서 제외함으로써 삼각망 수를 감소한다. 제안하는 방식은 소프트웨어 차폐 컬링에 비해 초당 화면 갱신율이 20% 정도 향상하였다.

하지만 [6]에서는 GPU가 가시성 테스트를 수행하는 도중에 쿼리가 수신되지 않는 경우 CPU의 실속과 GPU 기아 현상이 발생할 수 있다고 소개하며 이와 같은 문제점을 해결하는 기법을 제안한다. 이를 위해서 차폐 쿼리 수를 최소화하고 렌더링 패키지를 통합하여 단순함과 범용성을 향상하였다[6]. [6]에서 제안하는 알고리즘은 앞서 언급한 알고리즘과 동일하게 가시성 테스트를 위한 깊이 맵에 대한 GPU 전송은 동일하다. 하지만 [5]의 알고리즘과 차이는 마지막 프레임에서 연산된 노드의 차폐 쿼리를 사용함으로써 가려지는 물체가 발생하면 이후의 가려진 물체에 대한 쿼리를 처리하지 않기 때문에 CPU 실속과 GPU 기아 현상을 방지한다. 이와 같이 실시간 렌더링 시스템의 차폐 컬링 기법은 대용량의 지형 데이터의 처리 시 초당 화면 갱신율의 저하를 방지할 수 있다. 하지만 차폐 컬링은 모든 물체에 볼륨을 적용하여 가시성 테스트를 수행하기 때문에 대용량 지형을 렌더링하는 시스템에서는 시점이 높을 경우 가리는 물체와 가려지는 물체의 경계가 사라진다. 따라서 가시성 테스트의 연산도 많아 질 뿐만 아니라 시각 절두체 컬링을 사용하는 경우보다 더 많은 오버헤드를 발생한다.

이렇게 대부분의 실시간 렌더링 시스템의 과부하 처리 기술은 단순히 물체를 이루는 삼각망을 감소하는 것에 초점을 맞추고 변화하는 시점에 관한 연구는 부족하다. 그러므로 본 논문에서는 무조건적인 삼각망을 감소하는 방식이 아닌 시점 변화를 중점으로 지형 데이터를 컬링하는 방안을 제안한다.

Ⅲ. 제안 메커니즘

1. 가시성 테스트를 위한 고도 상태 결정

항공기 시뮬레이션 동작 중 저고도 상태는 크게 세 가지이며 이는 이륙/착륙 및 저고도 비행으로써 고도와 상승각을 기준으로 분류한다. 하지만 저고도 비행 상황에서 특정 고도를 저고도라도 단정하기 어렵다. 예를 들어 항공기의 착륙 시 활주로 접근 고도가 4500ft라고 가정 시 4500ft 이하를 저고도라고 분류할 경우 4501ft 는 고고도로 분류가 되며 실시간 렌더링 시 시각 절두체 컬링만 적용되고 1ft 차이로 컬링 기법이 바뀌는 경우가 발생한다. 따라서 특정 고도를 저고도와 고고도를 분류 기준으로 사용하는 것은 피해야 한다. 이러한 문제점을 해결하기 위해 퍼지 이론을 사용하여 기준이 모호한 고도를 분류한다. 퍼지 이론이란 각 원소는 집합에 속하는 기여도가 존재하며 이때 기여도는 0과 1사이의 실수로 표현한다. 이는 원소가 집합에 속하는 경우를 1, 전혀 속하지 않는 경우를 0으로 정의한다[7]. 본 논문에서 제안하는 고도에 따른 기여도는 식 (1)에 의해 연산되며 각각의 기여도는 표 1에서 제시한다.

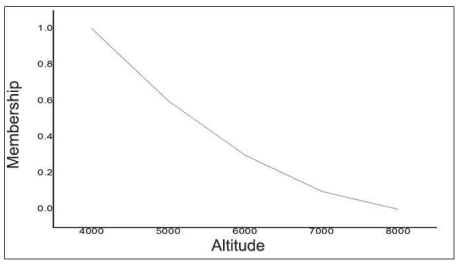
$$Membership = Membership\,Phase - (\frac{Current\,Altitude - Altitude\,Phase}{Altitude\,OffSet} * ((Max\,Altitude - Current\,Alitude) * Memvership\,OffSet))) \quad (1)$$

수식 1에서와 같이 기여도는 각각 4단계로 분류되며 4000ft에서 8000ft까지 1000ft 단위로 분류된다. 이러한 분류에 의해 10ft 단위로 고도가 변화할 때마다 현재 고도값과 단계별 기여도 오프셋에 따라 최종 기여도가 연산된다. 표 1과 같이 고도는 4000ft에서 1ft당 기여도가 변화하며 8000ft 상에서의 기여도는 0이 된다. 또한 상승각의 경우 30도의 경우에는 전체 지형이 다 보이는 상태이므로 기여도가 1이다. 하지만 상승각이 -30인 경우에는 하늘만 보이는 상태이므로 기여도가 0이 된다. 표 1을 기반으로 컬링 메소드 결정을 위한 기여도 함수는 그림 1과 같다.

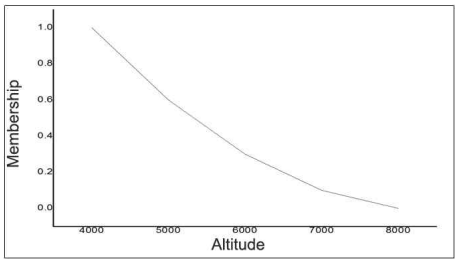
그림 1과 같이 4000ft의 기여도는 1.0이 되며 고도의 상승과 항공기 상승각에 따라 기여도가 증가한다. 상승각이 상승할수록 화면상의 지형은 감소하기 때문에 기여도는 0에 수렴하며 하강 할수록 화면상에 표시되는 지형이 많기 때문에 기여도는 1에 수렴한다. 이와 같이 연산된 기여도에 따라 컬링 메소드가 결정되며 후면 컬링 결정 시 기여도가 높을수록 높은 해상도의 깊이맵을 생성하고 낮을수록 낮은 해상도의 깊이맵을 생성한다. 기여도에 따른 깊이 맵의 크기는 표 2에 제시한다.

표 1. 고도와 상승각에 따른 퍼지 이론 기여도
Table 1. Fuzzy Membership based on Altitude

Altitude	Membership	Pitch Rate	Membership
4000ft	1.0	30	0
4010ft	0.996	29	0.017
…	…	…	…
5000ft	0.600	…	…
5010ft	0.597	…	…
…	…	1	0.493
6000ft	0.300	0	0.5
6010ft	0.298	-1	0.507
…	…	…	…
7000ft	0.1	…	…
7010ft	0.099	…	…
…	…	…	…
7990ft	0.001	-29	0.983
8000ft	0.0	-30	1.0



(a) 고도에 따른 기여도
(a) Fuzzy Membership Based on Altitude



(b) 상승각에 따른 기여도
(b) Fuzzy Membership Function Based on Pitch Rate

그림 1. 컬링 메소드 결정을 위한 퍼지 기여도 함수
Fig. 1 Fuzzy membership function for culling method function

표 2. 퍼지 기여도에 따른 깊이맵 크기
Table 2. Z-Map size based on Fuzzy Membership

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

Proposed Algorithm

```
Require : terrainNode
Require : zBuffer
Require : zMapSize
Require : zMapCameraNode
Require : aircraft_position {from host }
Require : triangles { the amount of terrain vertices }

1:  function aircraft_position (receive from host)
2:  function Read_zBuffer (aircraft_position)
3:  function base (terrain_vertices, altitude, pitch)
4:  if base = 1 then
5:     culling method is dynmic culling
6:     function zMapSize (fuzzy_membership)
7:  endif
8:  for i = 0 to zMapSizeX do
9:     for j = 0 to zMapSizeY do
10:       input zBuffer(i, j) ← pos
11:    endfor
12: endfor
13: function Create_zMap (zBuffer, zMapSize)
14: zMap send to GPU
15: if occludee_Depth > occluder_Depth then
16:    depth ← occludee_Depth
17:    if depth = 1 then
18:       traingle is visibile
19:    else
20 :      continue
21 :   endif
22: else
23:    triangle culling
24: endif
```

그림 2. 가시성 결정과 컬링 알고리즘
Fig. 2 Visibility Set and Culling Algorithm

이렇게 설정된 깊이 맵은 컬링 메소드가 후면 컬링으로 결정 시 GPU로 전송되며 가시성 판단에 사용한다.

2. 가시성 결정 및 컬링

이번 절에서는 이전 절에서 설정된 컬링 메소드 결과값을 사용하여 동적인 깊이 맵의 생성과 보이지 않는 영역에 대한 컬링 방식을 제안한다. 제안하는 메커니즘의 순서는 아래 그림 2에 제시한다.

우선 컬링을 적용할 지형의 정보와 가시성 판단을 위한 z-map_camera, z-buffer, z-map_size가 필요하다. 또한 컬링 메소드를 결정하는 항공기 상태 정보가 필요하다. 우선 항공기의 상태 정보는 항공 역학을 포함하는 호스트에서 생성된 정보를 수신한다. 수신된 정보를 기준으로 이전 절에서 설명한 바와 같이 고도와 상승각을 기준으로 결정된 컬링 메소드가 고고도 일 경우는 시각 절두체 컬링을 적용한다. 이때 z-map_camera는 현재 시점 카메라에 종속된 카메라로써 오직 깊이 맵의 생성을 위한 것이기 때문에 실시간 렌더링에는 관여하지 않는다. 이렇게 z-map_camera로부터 생성된 깊이 맵을 기준으로 고고도 상태 일 경우에는 보이지 않는 지형의 삼각망을 컬링 시 일반적인 시각 절두체 컬링을 사용하는 것이 효과적이므로 시각 절두체 컬링을 사용한다. 하지만 저고도 상태일 경우에는 지형의 후면 컬링을 사용하게 되므로 깊이 버퍼 값을 사용하여 깊이 맵을 생성한다. 또한 생성된 깊이 맵은 GPU의 가시성 테스트를 위해 전송된다. GPU는 수신된 깊이 맵의 크기만큼의 모든 픽셀마다 가시성 판단을 하게 되고 물체의 후면을 판단한다. 수행된 가시성 테스트의 결과가 1이면 가리는 물체로 판단하여 실시간 렌더링을 진행하고 결과가 0인 경우는 가려지는 물체로 판단하여 컬링한다. 이렇게 컬링된 삼각망은 실시간 렌더링 시 제외되며 전체 시스템의 성능을 향상한다.

Ⅳ. 구현 및 성능 평가

이번 절에서는 본 논문에서 제안하는 메커니즘을 기반으로 구현된 내용과 실험 결과를 제시한다. 본 논문에서는 구현을 위한 오픈 소스 그래픽 라이브러리인 Open Scene Graph를 사용한다. OSG는 다수의 가상 물체 및 멀티미디어 객체를 효과적으로 관리하기 위해 계측적 트리 형식의 자료구조인 Scene Graph 이론에 기반한다[8]. 이러한 OSG 그래픽 라이브러리를 사용하여 본 논문에서는 구현된 메커니즘의 성능 평가를 위해 예천 공항 주변 100km 지역의 지형 정보를 사용하며 각각 10m 해상도의 DTED(Digital Terrain Elevation Data)와 위성 영상을 사용하였다. 이러한 지형은 수신된 항공기의 상태 정보에 따라 컬링 메소드를 결정하고 이에 따라 지형을 컬링하여 실시간 렌더링 시 삼각망의 수를 감소한다. 그림 3는 저고도 상태일 경우의 렌더링 화면을 보여준다.

그림 3에서 보는 바와 같이 저고도 상태에서 후면 컬링을 위해 두 번의 렌더링 과정을 거친다. 첫 번째는 (a)의 이미지와 같이 화면 렌더링을 위한 카메라의 종속된 zmap_camera에서 깊이 맵 생성을 위한 깊이 버퍼를 생성한다. 이 후 가시성 테스트를

(a) 렌더링 화면
(a) Rendering Image

(b) 깊이 맵
(b) Depth Map

그림 3. 저고도 상태의 깊이 맵과 렌더링 화면
Fig. 3 Z-Map and Rendering image of low-altitude state

거쳐 반환된 결과 값에 따라 삼각망을 컬링하며 (a)의 이미지와 같이 현재 카메라에서 렌더링을 하게 된다. 실제 사용자가 화면상에 보이는 모습은 (a)의 화면만 보이며, (b)의 이미지는 가시성 테스트만 수행 할 뿐 실제 렌더링 화면에는 출력되지 않는다. 이렇게 제안하는 기법의 구현은 Window 7 OS, Inter I7 2.8GHz의 CPU, 8GB의 RAM, nVidia GTX 470의 VGA를 포함하는 일반 PC환경에서 실험하였다. 이러한 환경에서 실시간 렌더링 시스템은 C++ 기반 뷰어 소프트웨어와 C# 기반 호스트 시스템 소프트웨어로 구성된다. 두 소프트웨어의 데이터 통신 주기는 60Hz 주기로 UDP 환경에서 통신한다. 이러한 환경에서 고도에 따른 실험 데이터는 표 3와 같다. 또한 4000ft 고도 상에서 컬링 사용 전과 시각 절두체 컬링 및 제안하는 컬링 기법의 렌더링 화면은 그림 4에서 보여진다.

그림 4의 (a)는 컬링 사용 전이며 (b)는 시각 절두체 컬링을 사용했을 경우이고 (c)는 제안하는 컬링 기법을 사용했을 경우이다. 각각의 컬링 기법을 사용한 실제 렌더링 이미지는 동일하게 표현된다. 표 3와 같이 컬링을 전혀 사용하지 않는 상태에서의 지형의 삼각망의 수는 11,079개이다. 그러나 일반적인 시각 절두체 컬링의 사용 시 컬링을 사용하

표 3. 고도 기반 컬링 비교
Table 3. Comparison of Altitude based culling

(a) 컬링 기법 사용 전
(a) Case without Culling

ALTITUDE	TRIANGLE	VERTICES
2,000ft	11,079	204,100
4,000ft	11,079	204,100
6,000ft	11,079	204,100
8,000ft	11,079	204,100

(b) 시각 절두체 컬링
(b) Case for View Frustum Culling

ALTITUDE	TRIANGLE	VERTICES
2,000ft	2,560	106,126
4,000ft	2,634	107,310
6,000ft	2,724	108,750
8,000ft	2,854	110,980

(c) 제안하는 컬링 기법
(c) Case for Proposed Culling

ALTITUDE	TRIANGLE	VERTICES
2,000ft	2,176	99,982
4,000ft	2,304	102,030
6,000ft	2,412	103,758
8,000ft	2,526	105,732

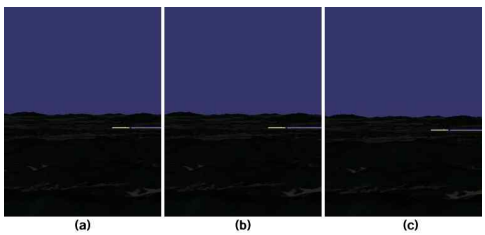


그림 4. 컬링 메소드의 실시간 렌더링 화면 비교
Fig. 4 Real-time rendering Image comparison of Culling

지 않는 경우와 비교하여 약 80%의 삼각망 수가 감소하였다. 마지막으로 제안하는 알고리즘의 경우 시각 절두체 컬링과 비교하여 삼각망의 수가 약 15% 정도 감소하였다. 이렇게 삼각망의 감소가 실제 렌더링 과정에서 초당 화면 갱신율에 적용되는 모습은 그림 5와 표 4에서 제시한다.

표 4과 그림 5에서 보이는 바와 같이 각각의 알고리즘 적용 시 초당 화면 갱신율은 차이를 보였다. 먼저 컬링 미적용 상태에서는 지형의 모든 고도 상태에서 16.65 fps의 성능을 보인다. 하지만 시각

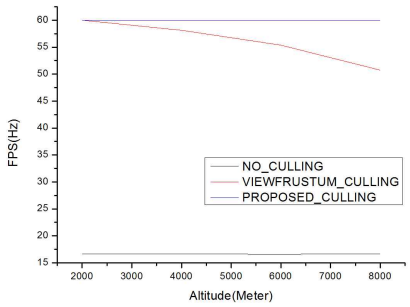


그림 5 컬링 메소드의 실시간 렌더링 속도 비교
Fig. 5 Real-time rendering FPS comparison of Culling

표 4 컬링 메소드의 실시간 렌더링 속도 비교
Table. 4 Real-time rendering FPS comparison of Culling method

	Case without Culling	Case for View Frustum Culling	Case for Proposed Culling
2,000ft	16.65 fps	60 fps	60 fps
4,000ft	16.63 fps	58.12 fps	60 fps
6,000ft	16.58 fps	55.37 fps	60 fps
8,000ft	16.66 fps	50.72 fps	60 fps

절두체 컬링은 삼각망이 감소하여 50~60 fps의 성능을 보였다. 마지막으로 제안하는 방안은 60 fps를 보이며 이는 구현된 소프트웨어의 최대 갱신율인 60 fps을 만족하였다. 이처럼 제안하는 방안은 컬링 미적용 상태와 비교하여 약 80%의 삼각망 수를 감소하였으며 73%의 실시간 렌더링 속도의 향상을 보였다. 또한 시각 절두체 컬링과 비교하여 15%의 삼각망을 감소하였고 8%의 실시간 렌더링 속도 향상을 보였다.

Ⅳ. 결 론

본 논문에서는 항공기 시뮬레이션에서 고도가 낮을 경우 발생하는 불필요한 연산을 방지하기 위해 고도와 상승각 정보를 기반으로 처리하는 컬링 기법을 제안하였다. 이를 위해 퍼지 이론을 사용하여 고고도와 저고도를 판별하고 각 상황에 맞는 깊이 맵을 생성하였다. 또한 이를 GPU에 전송하여 처리하게 함으로써 CPU의 실속과 GPU의 기아 상태를 해결하였다. 마지막으로 제안하는 방안을 실제 지형 데이터에 적용하여 실험하여 15%~ 80%의 삼각망을 감소와 8%~73%의 렌더링 속도를 향상하였다. 이를 통해 저고도 상태에서는 불필요한 연산이 포함된다는 사실과 실제 시뮬레이터에 적용 가능성을 확인하였다. 하지만 현재 제안하는 방안은 항공기의 일반적인 수평 비행 상태에서 퍼지 이론에 따른 기여도를 합산하여 적용하였다. 따라서 향후 제안하는 방안을 확장하여 항공기의 모든 상태 정보를 기반으로 동적 컬링 방안에 대한 연구를 진행할 예정이다.

References

[1] M.K. Zyskowski, “Aircraft simulation techniques used in low-cost, commercial software,” Proceedings of AIAA Modeling & Simulation Technologies Conference and Exhibit, pp. 11-14, 2003.

[2] M.S. Sunar, A.M. Zin, T. Mohd, T. Sembok, “Improved view frustum culling technique for real-time virtual heritage application,” The International Journal of Virtual Reality, Vol. 7, No. 3, pp. 43-48, 2008.

[3] S. Kumar, D. Manocha, B. Garrett, M. Lin, "Hierarchical back-face culling," Proceedings of Workshop on Eurographics Rendering, pp. 231-240, 1996

[4] M.N. Leone, L.R. Barbagallo, M.M. Banquiero, D. Agromayor, A. Bursztyn, “Implementing software occlusion culling for real-time applications,” Proceedings of CACIC, pp. 416-426, 2012.

[5] L.R. Barbagallo, M.N. Leone, R.N. Garcia, “Vertex discard occlusion culling,” RedUNCI, pp. 309-407, 2012.

[6] J. Bittner, M. Wimmer, H. Piringer, W. Purgathofer, “Coherent hierarchical culling: hardware occlusion queries made useful,” Proceedings of Workshop on Digital Media and Digital Content Management, pp. 30-34, 2011.

[7] L.A. Zadeh, “Fuzzy sets,” The International Journal of Information and Control, Vol. 8, pp. 338-353, 1965.

[8] X. Hu, S. Bo, Z. Huiqin, X. Bing, W. Hao, H.

Ge, “Visual simulation system for flight simulation based on OSG,” Proceedings of International Conference on Audio Language and Image Processing, pp. 562-566, 2010.

Chungjae Lee (이 충 재)

He 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.
Email: cjlee@gnu.ac.kr

Seokyoon Kang (강 석 윤)

He received the B.S. and M.S. degrees in informatics from Gyeongsang National University. He is currently working toward Ph.D. degree at Gyeongsang National University. His research interests include Flight Simulation Software.
Email: syk@gnu.ac.kr

Ki-Il Kim (김 기 일)

He received M.S. and Ph.D. degrees in computer science from the ChungNam National University, Daejeon, Korea. He is currently with the Department of Informatics at Gyeongsang National University. His research interests include ad hoc/ sensor networks and avionics software.
Email: kikim@gnu.ac.kr