컨벡스 헐을 이용한 개선된 구 좌표계 기반 렌더링 방법※

김남중°, 홍현기*
중앙대학교 첨단영상대학원 영상학과°·*
sogalman@naver.com, honghk@cau.ac.kr

Improved Rendering on Spherical Coordinate System using Convex Hull

Namjung Kim°, Hyunki Hong*
Image Dept., GSAIM, Chung-Ang University

요 약

본 논문에서는 컨벡스 헐을 이용한 구 좌표계 기반 실시간 렌더링 알고리즘이 제안되었다. OpenGL 렌더링 파이프라인은 물체의 모든 정점들을 고려하지만, 제안된 방법은 물체의 가시 삼각형들을 검사하여 보이는 정점들만을 고려한다. 본 논문에서는 구좌표계 표현에서의 물체의 가시 영역을 결정하기 위하여, 카메라 절두체를 이루는 6개의 평면 방정식과 물체의 경계구와의 기하 관계를 이용한다. 또한 대상 물체의 컨벡스 헐(convex hull)의 최대 측면 성분(maximum side factor)을 고려하여 은면(hidden surface)을 제거하는 효과적인 방법이 구현되었다. 실험결과로부터 결과 영상이 원본 영상과 거의 같고, 렌더링 성능이 크게 개선됐음을 확인하였다.

ABSTRACT

This paper presents a novel real-time rendering algorithm based on spherical coordinate system of the object using convex hull. While OpenGL rendering pipeline touches all vertices of an object, the proposed method takes account the only visible vertices by examining the visible triangles of the object. In order to determine the visible areas of the object in its spherical coordinate representation, the proposed method uses 3D geometric relation of 6 plane equations of the camera frustum and the bounding sphere of the object. In addition, we compute the convex hull of the object and its maximum side factors for hidden surface removal. Simulation results showed that the quality of result image is almost same compared to original image and rendering performance is greatly improved.

Keyword : real-time rendering, spherical coordinate system, visibility, convex hull, GPU

접수일자 : 2009년 08월 21일
일차수정 : 2009년 12월 31일
심사완료 : 2010년 01월 21일
※ 이 연구는 서울시 산학연협력사업(10570)으로 구축된 서울 미래형콘텐츠컨버전스 클러스터, 교육부 2단계 두뇌한국21(BK21) 지원사업으로 수행되었음.

1. 서 론

많은 데이터를 가진 장면을 실시간에 렌더링하는 기술은 컴퓨터그래픽에서 중요한 연구분야이다. 최근까지 그래픽스 하드웨어의 처리 능력이 꾸준하게 발전하고 있지만, 렌더링하고자 하는 장면의 데이터 수도 급격하게 증가하기 때문에 실시간 렌더링의 성능에는 한계가 존재한다. 이런 문제를 해결하기 위해, 장면의 기하 데이터를 효율적으로 구성하는 공간자료구조(spatial data structure) 방법, 최종 영상의 정확도를 유지하면서 렌더링과정에서 고려되는 모델 데이터 수를 줄이는 방법등의 연구가 다양하게 제안되었다[1,2,3,4,5].

장면 및 물체에 대한 공간자료구조를 구성하는 연구는 3차원 공간상에 장면에 대한 효율적인 기하 구조를 사용하여 광선(ray)과 물체 사이의 교차 관계 등의 연산 시간을 줄이며, BVH (Bounding Volume Hierarchy), BSP(Binary Space Partitioning), 8진 트리, 장면 그래프 등의 가속화(acceleration) 구조 방법 등이 있다[1,2,3]. 또한 카메라에 의해 보이지 않는 부분을 렌더링과정에서 생략하는 선별(culling), 특정 물체에 대한 간략한 표현을 사용하는 상세단계(level of detail) 방법 등이 있다[4,5]. 장면 내 물체를 구성하는 모든 정점(vertex)을 고려하지 않기 위해 후면 선별 등의 가속화 방법이 널리 사용되지만, 시선 벡터와 면 법선 벡터간의 각도를 조사하는 연산과정이 요구되어 속도저하를 야기시킨다[6].

그리고 물체가 카메라로부터 멀리 떨어져 있으면 낮은 상세 단계로, 가까이 있으면 높은 상세 단계로 렌더링하여 실시간으로 상세단계를 변경하는 점진적 메쉬 방법이 제안되었다[7,8]. 그러나 점진적 메쉬 방법은 메쉬를 구성하는 정점을 분리하거나 변을 제거하기 때문에, 근본적으로 메쉬 정보가 변경되는 문제가 발생한다. 또한 카메라에 보이는 가시 기하 집합에 대한 고려가 전혀 없다는 문제점이 있다.

그리고 주어진 시점에서 가장 보일만한 기하 집합을 근사(approximation)하여 결정하는 우선순위 기반 계층적 투영(Prioritized Layered Projection; PLP) 방법이 제안되었다[9]. PLP 방법은 실시간으로 렌더링되는 장점이 있지만 영상의 품질을 보장하지는 못하며, 철저하게 사용자에 의해 정의된 기획안(budget)을 통해서 배열(ordering), 렌더링되는 단점이 있다. 이러한 문제점을 보완하기 위하여 CPLP(Conservative PLP)방법이 개발되었지만, 처리 속도가 저하되어 실시간 렌더링에 맞지 않는 단점이 있다[10].

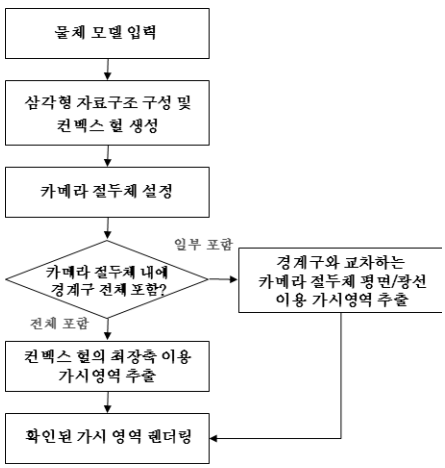
렌더링과정에서 일반적으로 물체를 구성하는 다각형을 기준으로 가시성(visibility)을 결정하지만, 점들로 구성된 물체의 점 클라우드(cloud)에 대해 주어진 시점에서의 가시성을 결정하는 방법이 제안되었다[11]. 이 방법은 점 클라우드로부터 생성한 컨벡스 헐(convex hull)에 존재하는 점들을 추출하여 정확하게 가시성을 판단한다. 하지만 수행속도가 느려서 실시간 렌더링에 맞지 않으며, 점 클라우드에 대해서 처리하기 때문에 일반적으로 적용되기 어려운 단점이 있다.

일반적으로 3차원 물체는 직교 좌표계에서 표현되지만, 좌표계 변환을 통해 구 좌표계 상의 표현을 이용해 렌더링 성능을 향상시키는 방법이 제안되었다[12]. 이 방법은 구좌표상의 표현을 이용해 물체의 가시 영역을 판단하고 가려지는 영역을 구분하는 가능성을 제시하였으나, 가시 영역이 얼마나 정확하게 추출되었는지 평가하는 기준이 없고 영역적으로 정확한 영상을 생성하며 원본 영상과의 비교도 없다.

본 논문에서는 일반적으로 렌더링 과정에서 사용되는 삼각형 메쉬를 기반으로 하여, 사용자의 수동적인 입력 없이 실제로 보이는 가시 영역에 가까운 가시 집합을 추출하는 새로운 실시간 렌더링 알고리즘을 제안한다. 또한 가시 집합을 얼마나 정확하게 추출하는지에 대한 객관적 평가 기준을 제시하며, 모든 결과 영상에 대하여 원본 영상과의 비교를 통하여 추출된 가시 집합의 정확성을 객관적으로 평가하였다. 그리고 물체의 경계구와 물체

의 컨벡스 헐, 시점, 시각 절두체(view frustum) 등과의 기하 관계를 이용하여 가시 집합을 추출하여 실시간으로 렌더링 하는 시스템을 구현하였다. 제안된 방법의 흐름도를 [그림 1]에 나타내었다. 대상 장면에 존재하는 물체의 구좌표계 표현과 해당 물체의 경계구 및 컨벡스 헐을 사전 작업을 통해 구한다. 렌더링 과정에서 실제한다. 의 시각 절두체가 주어지면, 물체의 경계구의 포함 여부에 따라 가시 영역을 결정한다. 특히 물체 전체가 가시 영역으로 판단되는 경우, 컨벡스 헐의 최장축을 구해 물체 자신에 의해 가려지는 은면을 결정하여 렌더링 성능을 향상시킨다.

2장에서는 물체의 구 좌표 변환과 자료구조들과 상세 알고리즘에 대해서 소개하고, 3장에서는 제안된 방법에 의해 얻어진 렌더링 영상과 성능을 분석하며, 결론 및 이후 연구를 4장에 기술한다.



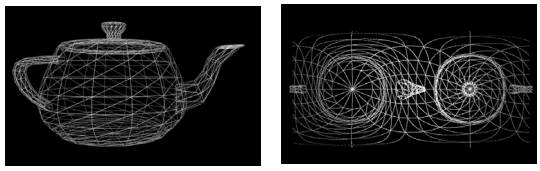
[그림 1] 제안된 알고리즘의 흐름도

2. 제안된 방법

본 논문에서는 물체의 구좌표 표현을 이용한 렌더링 방법[12]의 단점을 보완하며 렌더링 영상의 품질과 렌더링 속도 모두 크게 개선된 방법이 제안된다. 기존 방법에서는 절두체의 시선광선 4개와 물체의 경계구와의 교차점을 구하며, 물체 자체에 의해 가려지는 은면을 제거하기 위해 경계구를 시선벡터에 따라 샘플링한다. 이 과정에서 물체를 이루는 모든 삼각형의 내부가 어떤 구좌표 값으로 채워져 있는지 삼각형 색인(index) 맵에 정보를 저장한다. 그러나 삼각형 색인 맵은 기존 방법의 자료구조에서 가장 용량이 크며, 물체를 이루는 삼각형의 개수가 많아질수록 삼각형 색인 맵을 생성하는 전처리 시간은 급속히 증가하며, 삼각형 색인 맵을 생성하지 못하는 모델들은 처리하지 못하는 문제점 등이 있다. 또한 삼각형 색인 맵을 사용하여 절두체의 시선 광선 4개와 물체의 경계구가 교차하는 경우를 고려했기 때문에 구현 상에서 물체의 내부에 시점을 설정하였으며, 이를 다시 회전 및 이동하는 등의 추가 변환작업이 요구된다.

본 논문에서는 카메라의 절두체를 이루는 6개의 평면과 물체와의 기하 관계, 컨벡스 헐을 이용한 기하 관계를 이용하여 삼각형 색인 맵 없이도 가시 공간 내에 존재하는 물체 영역의 구좌표계 범위를 정확하게 구분할 수 있다. 또한 가시 집합을 얼마나 정확하게 추출하는지에 대한 객관적 평가 기준을 제시하였다.

2.1 물체의 구 좌표계 변환



[그림 2] 물체 모델(좌)과 구 좌표계에서 표현(우)

물체의 모델 데이터는 직교 좌표계 (x, y, z)에서 정점과 각 정점들이 이루는 표면, 법선벡터 정보 등으로 구성된다. 효율적인 렌더링을 위해 물체의 모델정보는 구 좌표계에서 고도각(zenith: θ = 0~360°), 방위각(azimuth: ϕ = 0~180°), 그리고 구 중심에서 해당 지점까지의 거리 R로 표현된다. 이 관계를 [식 1]과 [그림 2]에 각각 나타내었다.

복잡한 형태의 물체는 동일 방위각과 고도각의 위치에서도 여러 개의 표면 데이터가 존재할 수 있으며, 이 경우에는 R에 의해 구분된다.

$$x = R \cdot sin(\theta)cos(\phi),\ y = R \cdot sin(\theta)sin(\phi),\ z = R \cdot cos(\theta)$$
$$R = \sqrt{x^2 + y^2 + z^2},\ \theta = arctan(\frac{\sqrt{x^2 + y^2}}{z}),\ \phi = arctan(\frac{y}{x})$$

[식 1]

2.2 자료구조 구성

일반적으로 렌더링의 기본적인 기하 요소는 삼각형이며, 제안된 방법에서는 정점들이 구성하는 삼각형의 무게 중심 값을 계산해 구 좌표계 상에 저장한다. 즉, 물체 표면을 구성하는 삼각형의 무게 중심의 위치를 거리 값 및 각도(R, θ, ϕ)에 따라 무게중심 맵에 저장하며, 같은 각도에 존재하는 여러 면들은 거리정보에 따라 정렬된다. 이렇게 구성한 무게중심 맵의 해당 θ, ϕ 에 존재하는 삼각형들이 저장된 주소들을 메모리 접근 맵에 저장한다. 제안된 방법은 카메라에 따른 가시 영역에 해당하는 삼각형들의 구좌표계의 고도 및 방위각을 결정하며, 이 범위에 존재하는 삼각형들을 OpenGL 정점 배열(array)을 이용해 렌더링 파이프라인으로 넘기고 그래픽스 처리장치(Graphics Processing Unit)를 이용해 렌더링 한다.

기존의 구좌표계 기반 렌더링 방법에서는 무게중심 맵, 삼각형 색인 맵, 메모리 접근 맵 등의 3개의 자료구조를 생성했지만, 본 논문에서는 가장 용량이 크고 전처리 시간이 오래 걸리는 등의 문제점이 있는 삼각형 색인 맵을 제거하고, 무게중심 맵과 메모리 접근 맵만을 이용해 메모리 공간과 전처리 시간을 절약하는 장점이 있다.

2.3 컨벡스 헐 구성

수학적 의미에서 컨벡스 헐은 실수 벡터 공간에서 점 집합에 대한 볼록 포락선(envelope)으로 정의하며, 이후 컴퓨터그래픽의 렌더링, 모델링 단계에서 널리 활용되고 있다. 본 논문에서는 3차원 물체의 경계구가 시각 절두체 내부에 모두 포함되는 경우, 렌더링되는 물체의 컨벡스 헐을 생성하여 가시 집합을 추출한다.

3차원 컨벡스 헐을 구성하는 알고리즘은 증가(incremental) 알고리즘을 이용하였다[13,14]. 3차원 물체에 대한 컨벡스 헐은 최소한 동일 평면상에 존재하지 않는 4개 이상의 정점이 존재하며, 최소한의 기본이 되는 모형은 4개의 정점으로 이루어지는 4면체(tetrahedron)가 된다. 그리고 3차원 물체의 정점들로 평면을 구성하면서 정점들과 평면들이 컨벡스 헐 내부에 존재하는지 검사하고 가시성을 검사하면서 컨벡스 헐의 모형을 만든다. 컨벡스 헐의 생성 알고리즘의 의사코드(pseudo code)를 [그림 3]에 정리했다.

```
Initialize H3 to tetrahedron(p0, p1, p2, p3).
for i = 4,….n-1 do
for each face f of Hi-1 do
    Compute volume of tetrahedron determined
    by f and pi.
    Mark f visible if volume < 0.
    If no faces are visible
        Then Discard pi (it is inside Hi-1)
    else
        for each border edge e of Hi-1 do
            Construct cone face determined by
            e and pi.
            for each visible face f do
                Delete f.
                Update Hi.
```

[그림 3] 3차원 컨벡스 헐의 생성 의사 코드

2.4 가시 집합 추출

제안된 방법은 효율적인 실시간 렌더링을 위해 물체의 가시 집합을 추출하여 가시 집합에 해당하는 삼각형들만 렌더링한다. 이를 위해 먼저 물체의 경계구가 시각 절두체에 모두 포함되는지, 일부분 포함되는지, 전혀 포함되지 않는지 여부를 시각 절두체 클리핑 알고리즘을 이용하여 판단한다[15].

이 알고리즘은 먼저 원근 사영(perspective projection)을 설정할 때 사용하는 X축 및 Y축의 뷰 각도(view angle), 근단면(near plane) 및 원(far)단면의 거리(distance)를 이용해서 시각 절두체를 구성하는 8개의 정점을 구한다. 8개의 정점을 이용하여 시각절두체를 구성하는 6개 평면의 법선벡터를 구하고, 각각의 평면의 법선벡터와 평면위의 점을 이용하여 평면의 방정식을 계산한다.

이렇게 얻은 시각 절두체를 구성하는 6개 평면 방정식의 데이터와 경계구의 중심위치, 경계구의 반지름, 경계구와 6개 평면과의 거리를 이용하여 경계구가 시각 절두체 바깥에 있는지 내부에 있는지 판단한다. 만약 물체의 경계구가 시각 절두체의 바깥에 있으면 물체의 경계구가 카메라 시야 외부에 존재하므로 렌더링에서 제외시킨다. [식 2]는 평면과 경계구와의 거리를 구하는 과정을 나타낸다.

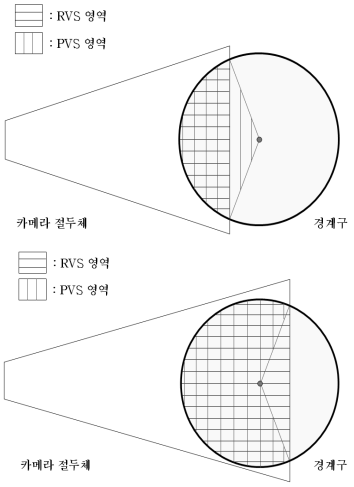
$$ax + by + cz + d = 0, \quad (3D\ plane\ equation)$$
$$Dist = a*BC\_x + b*BC\_y + c*BC\_z + d, \qquad [식\ 2]$$

where *Dist* represents a distance of view frustum's plane and the bounding sphere's center point (*BC_x*, *BC_y*, and *BC_z*).

물체의 경계구가 시각 절두체에 일부분 포함되면 시각 절두체의 각 평면과 경계구의 중심의 거리, 경계구의 반지름을 이용하여 물체가 시각 절두체 평면에 포함되는 구좌표 범위를 추출한다. 또한 물체의 구좌표계 기본(base) 축과 시선방향의 각도를 구해서 가시 집합에 해당하는 삼각형들을 결정한다. 이렇게 가시 집합에 해당하는 삼각형들을 렌더링 파이프라인으로 전달하여 렌더링하게 된다.

추출된 가시 집합에 대하여 더 자세히 살펴보면, 추출된 가시 집합을 PVS(Potentially Visible Set)으로 정의할 수 있으며[4], 물체에서 실제로 보이는 표면을 RVS(Real Visible Set), 대상 물체의 전체 표면의 집합을 TOS(Total Object Set)으로 정의할 수 있다. 이러한 PVS(Potentially Visible Set)는 TOS(Total Object Set)보다는 작아야 하며, RVS(Real Visible Set)보다 적어서는 안되고 최소한 같거나 많아야 한다. 의학 및 과학 시각화 등의 분야에서는 렌더링의 처리속도의 향상도 필요하지만, 지나치게 적은 기하 요소를 고려하면서 발생하는 결과 이미지의 품질 저하도 중요한 요소이다. 따라서 렌더링 성능을 유지하면서 대상 장면에 대한 실제 가시 집합을 적절하게 선택해야 한다. 본 논문에서 목표로 하는 추출 가시영역의 집합을 수식으로 표현하면 다음과 같다.

$$RVS \le PVS < TOS \qquad [식\ 3]$$



[그림 4] RVS 및 PVS 영역 비교

[그림 4]와 같이 물체의 경계구가 시각 절두체에 절반 이하로 포함되는 경우, [식 3]의 조건을 완벽하게 만족한다. 반면에 물체의 경계구가 시각 절두체에 절반을 초과하여 포함될 때는 추출한 가시영역 집합이 실제 보이는 집합보다 작게 된다. 경계구의 반지름과 물체의 삼각형들의 구좌표의 거리 성분을 비교해서 정확하게 가시영역을 추출할 수 있지만, 계산시간이 많이 소요되기 때문에 관련 추가계산을 배제하였다.

물체 전체가 시각 절두체에 포함되는 경우, 물체 내에 가려지는 영역을 판단하기 위해 컨벡스 헐을 이용하여 은면을 제거한다. 컨벡스 헐을 구성하는 정점과 폴리곤 수는 실제의 물체 보다 매우 적으므로 실시간 가시성 검사에 효율적이다. 컨벡스 헐의 구성 표면에 대한 일관성을 검사한 다음, 시선벡터에 직교하는 수직축을 중심으로 좌우의

최장 측면 성분을 추출한다. 얻어진 두 지점에 대한 구좌표계의 방위각을 기준으로 각 방향으로 뒤에 존재하는 영역은 앞의 삼각형에 의해서 가려지는 은면으로 처리한다. [식 4]는 카메라의 시점기준으로 양 방향으로 가장 긴 성분들을 추출하는 의사코드이다. 이렇게 추출한 최장 성분을 이용해 구좌표계의 가시 범위를 계산하고 이들 구좌표 범위에 존재하는 물체 모델의 삼각형들을 렌더링 파이프라인에 전달한다.

$$MSF = ConvexVertex\_R * \sin(\frac{ConvexVertex\_\theta}{180.0}) * \sin(ConvexVertex\_\phi)$$

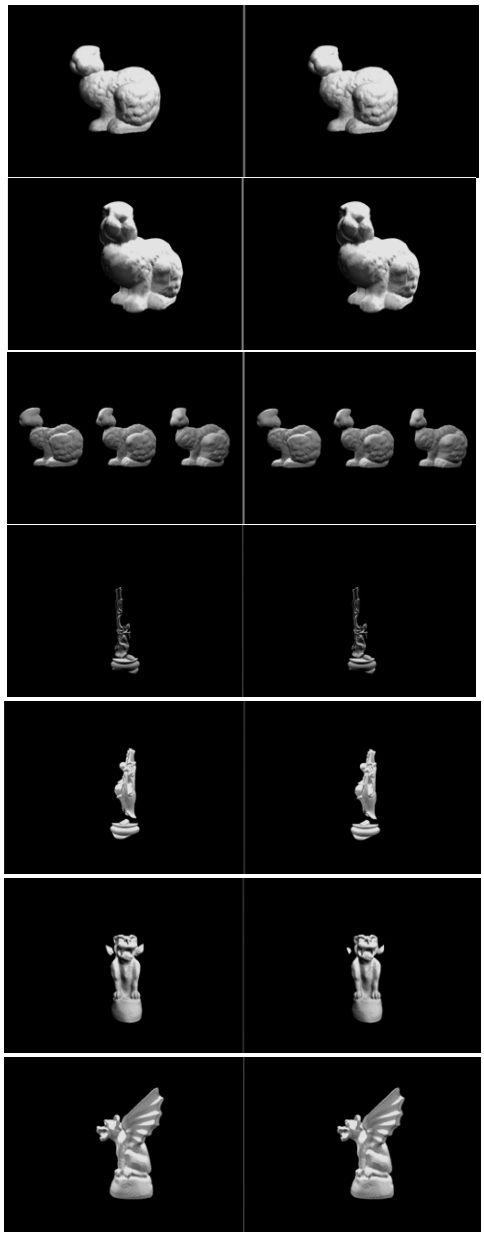
[식 4]

3. 실험 및 검토

제안된 알고리즘의 성능을 검증하기 위해 Intel Zeon CPU 2.33Ghz, RAM 4GB, Nvidia Quadro FX5500 그래픽스 하드웨어가 장착된 컴퓨터에서 여러 모델을 대상으로 실험했으며, 윈도우 XP 기반의 C++와 OpenGL, Nvidia 社의 Cg 셰이더(shader) 언어[16]를 이용했다. 스탠포드 대학의 버니(Bunny) 모델의 경우, 144,046개의 면과 72,027개의 정점으로 구성되어 있으며 모델 데이터의 크기는 약 10MB(Byte)이다. 제안된 방법에서는 무게중심 맵은 4.3MB, 그리고 동적으로 할당되는 메모리 접근 맵은 약 0.51MB의 저장공간이 요구된다. 이전 연구[12]에서 가시 영역을 판단하기 위해 6.7MB의 삼각형 색인 맵이 추가로 요구된다.

원본 영상과 제안된 알고리즘의 결과 영상 모두 OpenGL에서 GPU와 정점배열을 이용하여 렌더링 교하였으며 각각의 렌더링 성능과 고려되는 삼각형의 개수를 비교하였다. 또한 원본 영상과 결과 영상 모두 800x600 해상도에서 렌더링하였다. 렌더링 성능의 직접적인 비교를 위해 람버트 쉐이딩 모델을 기반으로 하였고 그림자 등은 고려하지 않았다.

스탠포드 대학의 버니(Bunny), Budda, Gargoyle 등의 다양한 모델을 대상으로 카메라의 시각 절두체에 일부 포함된 상황에서 OpenGL에 의해 렌더링된 원본 영상과 제안된 방법에 의해 얻어진 결과 영상을 [그림 5]에 나타내었으며, 각각의 렌더링 성능과 고려된 삼각형 개수를 [표 1]에 나타내었다.



[그림 5] 원본영상(좌)과 제안된 방법 영상(우)

[표 1] 렌더링 성능과 고려한 삼각형 개수 비교

	원본 영상		제안된 방법	
	고려된 삼각형 개수	fps	고려된 삼각형 개수	fps
Stanford Bunny	144,046	97	41,156	295
Stanford Bunny	144,046	97	59,630	209
Stanford Bunny 3개	432,138	33	123,492	115
Happy Buddha	100,001	41	36,751	84
Happy Buddha	100,001	41	40,788	78
Gargoyle	200,001	17	80,561	90
Gargoyle	200,001	17	91,693	76

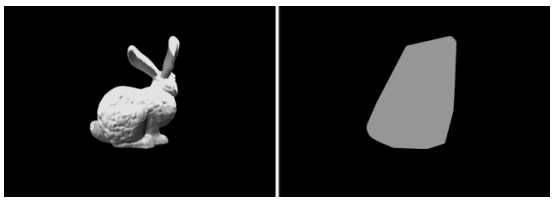
[표 2] 기존[12] 및 제안된 방법과의 성능 비교

		원본 영상		가시집합 영상	
		고려된 삼각형 개수	fps	고려된 삼각형 개수	fps
경우 1	기존 방법	144,046	295	13,508	486
	제안된 방법	144,046	97	16,588	565
경우 2	기존 방법	144,046	300	16,434	432
	제안된 방법	144,046	97	20,795	544

버니 모델을 대상으로 물체의 일부가 가시 영역에 포함되는 경우, OpenGL과 기존 연구[12], 제안된 방법의 결과 영상 및 최종 렌더링 성능비교를 [표 2]에 나타내었다. 2장에서 기술한 바와 같이 기존 연구는 구현과정에서 물체의 내부에 시점을 설정하였으며, 이를 다시 회전 및 이동하는 등의 추가 변환작업이 요구된다. 따라서 동일 장면에 대해 기존 방법, OpenGL 및 제안된 방법에 의해 얻어진 최종 렌더링 장면을 직접적으로 비교하기는 어렵다. 그러나 기존 연구에서 제시된 렌더링 성능 자료를 바탕으로 동일 모델에 대해 물체의 일부가 포함되는 경우(경우 1,2)에서 고려되는 삼각형 수를 기준으로 간접적인 비교 결과를 제시한다. OpenGL에 의해 얻어진 영상을 원본으로 기존 연구에서는 13,508개와 16,434개의 삼각형을 고려해 320×240 영상을 초당 486 및 432 프레임으로 각각 렌더링하지만, 알고리즘의 개선을 통해 제안된 방법에서는 이보다 많은 삼각형을 고려해도 800×600 영상을 초당 565 및 544 프레임으로 각각 렌더링한다.

버니 모델로부터 구성된 컨벡스 헐([그림 6])의 정점과 정점으로 구성된 삼각형의 개수가 각각 3,023개와 6,042이며, 전처리 과정으로 컨벡스 헐의 생성에는 약 5초가 소요된다.

대상 모델이 카메라의 시각 절두체에 모두 포함될 때 원본 영상과 결과 영상을 [그림 7]에 나타내었고, 각각의 렌더링 성능과 고려한 삼각형 개수를 [표 3]에 나타내었다. 렌더링 성능이 크게 향상되었고, 원본 영상과 제안된 방법에 의한 결과 영상 간의 차이도 거의 발생하지 않음을 확인하였다.

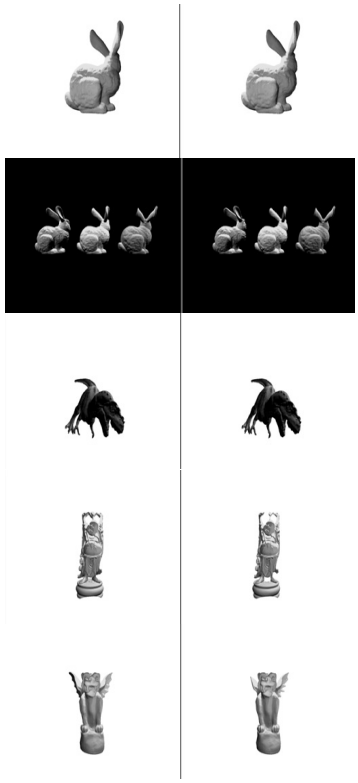


[그림 6] 버니 모델과 구성된 컨벡스 헐

물체가 카메라의 시각 절두체에 모두 포함되는 상황에서 은면 제거의 성능을 기존 방법[12]과 추가로 비교하기 위해 제안된 방법에서 10개 이상의 시점에서 최대, 최소 및 평균 렌더링 성능을 표 4에 나타내었다. 기존 논문에서는 640×480 영상을 평균 170 fps 속도로 렌더링 했지만, OpenGL 방법에 비해 성능개선이 불규칙하다. 그러나 제안된 방법은 800×600 영상을 평균 168fps 속도로 렌더링 했으며, 상대적으로 안정적으로 렌더링 성능이 개선됨을 확인하였다.

4. 결 론

본 논문에서는 컨벡스 헐을 이용한 구 좌표계 기반 실시간 렌더링 방법이 제안되었다. 기존 방법은 카메라의 시각 절두체를 구성하는 4개의 시선 광선과 경계구와의 교차점을 이용하며, 카메라 시점의 설정과정에서 추가의 변환이 요구되었다. 제안된 방법은 카메라의 시각 절두체를 구성하는 평면 방정식과 물체의 경계구와의 기하 관계를 이용하여 가시 영역을 정확하게 판단하였으며, 모델이 이루는 컨벡스 헐의 최장 성분을 고려하여 은면을 효과적으로 제거하였다. 실험을 통하여 결과 영상이 원본 영상과 거의 같고, 전체 렌더링 소요시간이 크게 단축됨을 확인하였다. 이후 연구에서는 BVH 등의 가속화 방법과 결합하여 보다 효율적인 렌더링 알고리즘을 개발하고, 복잡한 장면을 대상으로 제안된 방법의 활용 범위를 분석할 예정이다.



[그림 7] 원본영상(좌)과 제안된 방법(우) 결과영상

[표 3] 렌더링 성능과 고려된 삼각형 개수 비교

	원본 영상		제안된 방법	
	고려된 삼각형 개수	fps	고려된 삼각형 개수	fps
Stanford Bunny	144,046	97	80,804	136
Stanford Bunny 3개	432,138	33	242,326	51
Tyrano	200,001	18	171,192	42
Happy Buddha	100,001	41	63,257	53
Gargoyle	200,001	17	120,896	60

[표 4] 기존 방법[12]과 제안된 방법의 성능 비교

	기존방법(fps) [640x480]			제안된방법(fps) [800x600]		
	최대	최소	평균	최대	최소	평균
원본영상	134	134	134	97	97	97
가시집합 영상	199	137	170	194	136	168

참고문헌

[1] J. W. Ratcliff, "Sphere trees for visibility culling, ray tracing, and range searching", Game Programming Gems 2, Charles River Media, pp. 384-387, 2001.

[2] H. Samet, "The design and analysis of spatial data structures", Addison-Wesley, Reading, Massachusetts, 1989.

[3] T. Akenine-Möller, E. Haines, and N. Hoffman, "Real-time rendering", A.K. Peters Ltd., 2002.

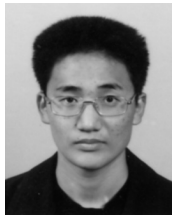
[4] D. Cohen-Or, Y. Chrysanthou, C. T. Silva, and F. Durand, "A survey of visibility for walkthrough applications", IEEE Trans on Visualization and Computer Graphics, Vol. 9, No. 3, pp. 412-431, 2003.

[5] J. H. Clark, "Hierarchical geometric models for visible surface algorithms," Communications of the ACM, Vol. 19, No. 10, pp. 547-554, 1976.

[6] D. Shreiner, M. Woo, J. Neider, and T. Davis, "OpenGL programming guide", 5th Ed., Addison-Wesley Ltd., 2005.

[7] H. Hoppe, "Progressive meshes", Proc. of

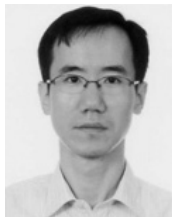
SIGGRAPH, pp. 99-108, Aug. 1996.
[8] H. Hoppe, "View-dependent refinement of progressive meshes", Proc. of SIGGRAPH, pp. 189-198, Aug. 1997.
[9] J. T. Klosowski and C. T. Silva, "The prioritized-layered projection algorithm for visible set estimation", IEEE Trans. on Visualization and Computer Graphics, Vol. 6, No. 2, pp. 108-123, 2000.
[10] J. T. Klosowski and C. T. Silva, "Efficient conservative visibility culling using the prioritized-layered projection algorithm", IEEE Trans. on Visualization and Computer Graphics, Vol. 7, No. 4, pp. 365-379, 2001.
[11] S. Katz, A. Tal, and R. Basri, "Direct visibility of point sets", ACM Trans. on Graphics, Vol. 26, No. 3, pp. 24:1-11, July 2007.
[12] 한은호, 홍현기, "물체의 구 좌표계 표현을 이용한 효율적인 렌더링 방법", 한국게임학회 논문지 8권, 3호, pp. 69-76, 2008.
[13] Joseph O'Rourke, "Computational geometry in C", Cambridge University Press, 1998.
[14] M. Berg, O. Cheong, M. Kreveld, and M. Overmars, "Computational geometry: Algorithms and applications", 3rd Ed., Springer, 2000.
[15] Tim Round, "Object occlusion culling," Game Programming Gems 2, Charles River Media, pp. 421-431, 2001.
[16] R. Fernando and M. J. Kilgard, "The Cg tutorial: the definitive guide to programmable real-time graphics", Addison-Wesley Professional, 2003.



김 남 중 (Namjung Kim)

2004년 8월 중앙대학교 컴퓨터공학과 학사
2008년 9월-현재 중앙대학교 첨단영상대학원 영상학과 석사과정

관심분야 : 컴퓨터그래픽스, 게임 등



홍 현 기 (Hyunki Hong)

1998년 8월 중앙대학교 전자공학과 박사
1998년 9월-1999년 8월 서울대학교 자동제어특화 연구센터 연구원
1999년 9월-2000년 2월 중앙대학교 정보통신연구소 연구교수
2002년 2월-2003년 1월 Univ. of Colorado at Denver, Post-Doc.
2000년 3월-현재 중앙대학교 첨단영상대학원 첨단영상학과 부교수 재직 중.

관심분야 : 컴퓨터그래픽스, 컴퓨터비젼, 증강현실 등