한국 CAD/CAM 학회 논문집
제 8권 제 2호 2003년 6월 pp. 75-83

학술논문

OSP Tree를 이용한 무한순차 입력 형상의 실시간 컬링에 관한 연구

표종현*, 채영호**

A Study on the Real Time Culling of Infinite Sets of Geometries Using OSP Tree

Pyo, J. H.* and Chai, Y. H.**

ABSTRACT

In this paper, OSP(Octal Space Partitioning) tree is proposed for the real time culling of infinite sets of geometries in interactive Virtual Environment applications. And MSVBSP(Modified Shadow Volume BSP) tree is suggested for the occlusion culling. Experimental results show that the OSP and MSVBSP tree are efficiently implemented in real time rendering of interactive geometries.

Key words : Virtual Reality, Data structure, Real-time rendering, Visibility culling, Occlusion culling

1. 서 론

디스플레이 시스템은 모니터로부터 시작하여 CAVE 시스템에 이르기까지 발전하고 있으며 그 발전방향은 사용자에게 더 큰 FOV(Field Of View)와 충실한 몰입감을 제공하기 위해 대형화되고 있고 다양한 분야에서 응용되고 있다. 하지만 대부분 가상환경을 구현할 때 사용자는 디스플레이 되는 장면을 관람하거나, 약간의 장치를 통해 장면을 컨트롤 할 수 있을 뿐이다. 사용자들은 자신이 보는 장면과 상호작용을 할 수 있기를 원하고, 좀더 새롭고 다양한 경험을 하고 싶어 한다. 사용자들은 기존의 3D-Max나 Maya와 같은 2D 기반에서 3D 모델링을 하기 보다는 가상환경에서 트랙커를 사용하여 실제로 눈으로 어떤 형태인지 보고 느끼면서 3D 모델링이나 창작활동을 할 수 있는 시스템까지 원하고 있다[1].

이러한 사용자들의 욕구를 충족시키기 위해서 장면과 상호작용을 할 수 있는 다양한 하드웨어의 개발과 가상환경 프로그램의 개발이 요구되고 있다. 또한, 사용자의 위치를 추적해 View-volume을 변화시켜 눈으로 볼 수 있는 영역을 바꿔줌으로써 몰입감을 증대 시켜 효율적인 모델링 작업을 할 수 있도록 한다. 이를 위해 사용자가 트랙커를 이용하여 위치, 방향, 속도를 입력 받아 그림을 그리거나 새로운 물체를 모델링 할 수 있도록 구성한다.

그러나, 가상환경에서 사용자가 계속해서 모델을 추가 구성하면 폴리건의 수가 많아지게 되고 어느 한계 지점에서는 더 이상 실시간으로 모델링을 할 수 없는 상태가 된다. 실시간 렌더링이 되지 않으면 사용자는 가상환경에 쉽게 몰입을 할 수 없고 불편함을 느끼게 된다. 따라서 이런 복잡한 모델을 실시간으로 렌더링 하기 위해서 시야 밖에 있는 모델을 제거하는 View-fustum 컬링과 가려져서 보이지 않는 모델을 제거하는 Occlusion컬링을 해주게 된다[2]. 본 논문에서는 선택된 Occlude를 효율적인 MSVBSP(Modified Shadow Volume BSP) 트리로 구성하여 Occlusion컬링에 사용한다.

이런 효율적인 가시화 테스트를 하기 위해서는 가상환경을 구성하는 모델을 계층적인 자료구조로 구성해야 한다. 그러나 기존의 자료구조는 실시간으로 입력되는 많은 폴리건을 자료구조로 구성할 때, 폴리건이 입력될 때 마다 많은 시간이 필요하다는 단점이 있다. 새로운 폴리건이 입력될 때 마다 처음부터 새로운 자료구조로 재구성 해 주어야 하기 때문이다. 따라서 본 논문에서는 실시간으로 입력되는 폴리건 정보를 공간 계층적 자료구조로 생성하는데 효율적인 방법인 OSP(Octal Space Partitioning) 트리를 제안하고 있다.

다음의 논문구성은 2장에서는 실시간 무한순차 입

*중앙대학교 영상공학과
**종신회원, 중앙대학교 영상공학과
- 논문투고일: 2003. 02. 10
- 심사완료일: 2003. 04. 18

력형상을 위한 자료구조에 대해 설명을 하고, 3장에서는 OSP 트리를 이용한 가시화 컬링에 대해 설명을 하고, 4장에서는 무한 순차 입력 형상 시스템 구현 및 결과에 대한 설명을 한 후, 5장에서는 결론에 관해 적었다.

2. 실시간 무한순차 입력형상을 위한 자료구조

2.1 Octal Space Partitioning Tree(OSP 트리)

본 논문에서는 크기가 제한된 공간에서 실시간으로 입력되는 폴리건 자료를 효율적인 자료구조로 생성하는 알고리즘을 제안한다. OSP 트리(Fig. 1(b))는 Octree[4](Fig. 1(a))와 마찬가지로 공간을 계층적으로 분할하는 형태이지만, 트리를 생성하는 과정에 있어 차이가 있다.

Octree의 경우 부모 노드에서 자식 노드로 나누어 질 때, 8개의 자식 노드로 나누어 지기 위해서 8개의 노드에 대해 각각 어느 노드에 속하는지 테스트를 하게 된다. Fig. 1(a)에서 전체 1의 노드를 4개의 자식노드 중 어디에 위치하는지 알기 위해서는 4개의 자식노드에 대해 2개의 평면에 대한 테스트를 해야 한다[4].

하지만, OSP 트리의 경우는 Fig. 1(b)와 같이 전체 공간을 2개의 평면에 대해서 각각 1번씩 테스트하게 되면 자료가 위치할 공간을 찾을 수 있게 된다. 즉, ①번 평면에 대해서 좌우로 나누고, ②와 ③ 평면에 대해서 상하로 나누면 ⓐ,ⓑ,ⓒ,ⓓ 4개의 공간에 어디에 위치 하는지 빠르게 알 수가 있다.

2.2 Octal Space Partitioning(OSP) Tree의 전처리 알고리즘

실시간으로 입력되는 자료를 계층적으로 구성하기 위해 프레임 마다 입력되는 폴리건이 OSP 트리의 한 부분으로 빠르게 계산되어 입력돼야 한다. 따라서 폴리건 자료가 입력 되기 전에 미리 OSP 트리의 자식 노드들을 계층적으로 만들어 둔다.

전처리 과정(Fig. 2)의 순서는 다음과 같다.

① 제한된 공간의 크기, 중심점, 최대 깊이 정보, State(상태, X)를 입력한다. 상태변수 값은 X, Y, Z가 있는데 현재 노드의 상태변수 값에 따라서 분할평면이 다르게 생성되게 된다.

② 현재 노드의 깊이 값이 최대 깊이 값 보다 작으면 제한된 공간의 중심점을 계산하고, 그 중심점을 지나는 분할 평면YZ(Fig. 4)를 생성한다.

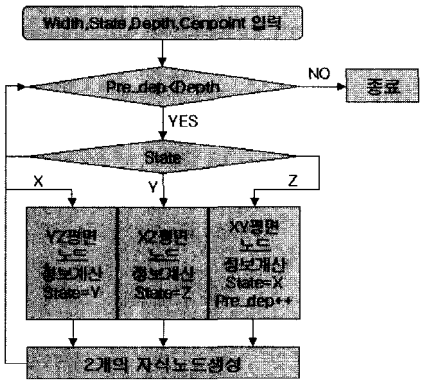

Fig. 2. Flow chart of preprocessing of OSP tree.

```
struct BIOCTREENODE
{
    PLANE           Splter;     // 공간을 나누는 평면
    BIOCTREENODE    *Front;     // 앞쪽에 위치하는 자식노드
    BIOCTREENODE    *Back;      // 뒤쪽에 위치하는 자식노드
    bool            IsLeaf;     // 마지막 노드
    bool            IsPolygon;  // 폴리건을 포함
    bool            IsSubnode;  // 자식노드 포함
    bool            IsFrustumCull; // View-Frustum cull
    eOctreePlane    state;      // X,Y,Z 평면 중에 머느것
    int             Depth;      // 현재 노드의 깊이값
    POINT3D         CenterPoint; // 경계입체 정보의 중심점
    float           Width;      // 현재 노드의 크기
    POINT3D         OSPBox[8];  // 경계입체 정보의 8점
    vector<int>     TriName;    // 포함하고 있는 폴리건 정보
};
```

Fig. 3. Data structure of OSP tree's node.

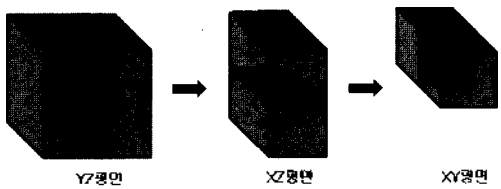

Fig. 4. Sequence of making child-node in OSP tree.

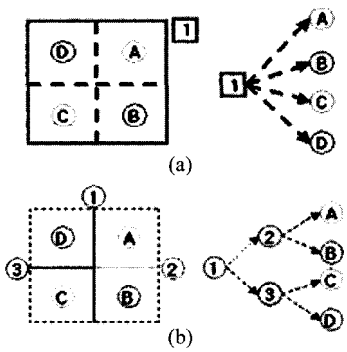

Fig. 1. (a) Octree method (b) OSP tree method.

입력되는 폴리건이 이 분할 평면을 기준으로 어떤 자식노드에 속하게 될지 판단하는 기준이 된다. YZ분할 평면에 의해서 나누어 지는 자식노드를 생성하고, 같은 방법으로 XZ, XY 분할 평면에 대해 실행한다. 이때 각각의 노드를 이루는 8개의 점을 계산하여 노드에 같이 저장한다.

③ ②의 과정은 같은 깊이 값을 가지며, 상태번수에 따라서 분할 평면이 다르게 된다. 1개의 노드가 YZ분할 평면에 의해 2개의 자식노드를 생성하고, 2개의 자식노드는 XZ분할 평면에 의해 4개의 자식노드를 생성하고, 4개의 자식노드는 XY분할 평면에 의해 최종 8개의 자식노드를 생성한다. 위 과정이 한 사이클이며 깊이 값이 최대 깊이 값에 도달할 때까지 반복한다.

Fig. 3에서는 각 노드가 가지는 자료의 형태를 나타내고 있다. 분할 평면을 이용하여 공간을 분할하는 OSP 트리의 특징은 BSP 트리와 비슷한 구조이다. 또한, 1개의 노드가 8개의 자식 노드를 생성하는 특징은 Octree와 비슷한 구조이다. 따라서 OSP 트리는 BSP 트리의 이분적 공간 분할 방식의 장점과 Octree의 3차원 정보를 갖고 있다는 장점이 있다. 이는 3장에서 설명할 가시화 컬링을 위한 중요한 정보이다.

2.3 Octal Space Partitioning Tree(OSP)의 자료 입력

전처리 과정을 거친 OSP 트리에 실시간으로 입력되는 자료를 계층적으로 구성하는 과정은 간단하다.

Fig. 5에서처럼 입력된 폴리건 자료는 우선 OSP 트리의 마지막 노드인지 검사를 하게 된다. 현재의 노드가 마지막 노드라면 폴리건 자료를 현재의 노드에 저장하고, IsPolygon을 활성화 시켜 현재 노드에 폴리건의 정보가 있음을 나타낸다. 현재 노드가 마지막 노드가 아니라면, 현재 노드의 분할 평면에 폴리건이 어디에 위치 하는지 검사를 한다. 폴리건의 포인트를 분할 평면의 방정식인 $ax+by+cz+d$에 대입하여 모든 포인트가 0보다 크게 되면 분할 평면의 앞쪽에 폴리건이 위치하고, 0보다 작게 되면 분할 평면의 뒤쪽에 폴리건이 위치하는 것이다. 나머지 경우는 분할 평면에 의해 폴리건이 분할 되거나 일치하는 경우이다.

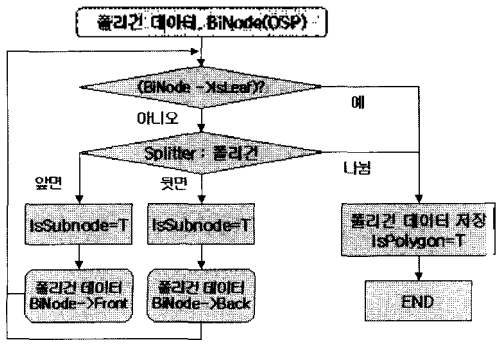

Fig. 5. Flow chart of data input of OSP tree.

폴리건이 앞쪽이나 뒤쪽에 위치 한다면 IsSubnode를 활성화 시켜 현재 노드 아래에 자식 노드가 존재함을 나타낸다. 그리고, 폴리건이 앞쪽에 위치하면 앞쪽의 자식 노드로 이동하고, 뒤쪽에 위치하면 뒤쪽의 자식 노드로 이동하여 위의 과정을 반복한다. 폴리건이 분할되는 경우는 IsPolygon을 활성화 시켜 현재 노드에 폴리건의 정보가 있음을 나타내고 과정을 마친다.

3. OSP 트리를 이용한 가시화 컬링

3.1 OSP 트리를 이용한 View-frustum 컬링

실시간으로 입력되는 자료를 OSP 트리로 구성하고, OSP 트리를 이용하여 View-frustum 컬링을 하게 된다. View-frustum 컬링은 OSP 트리의 특징 중에 하나인 Octree 정보를 이용하여 계산한다.

View-frustum 컬링은 OSP 트리의 가장 큰 상위 노드(Fig. 6A)부터 테스트를 시작 하게 되고, 노드를 구성하는 8점의 3차원 정보인 OSPBox의 모든 점이 View-frustum 평면 안에 위치(Fig. 6B에서 a노드) 한다면, 현재 노드와 그 자식 노드가 포함하는 폴리건 자료는 모두 보이게 되므로 렌더링에 포함하고, 바깥쪽에 위치(Fig. 6A에서 b노드) 한다면 현재 노드와 그 자식 노드는 완전히 시야 밖에 존재하므로 포함하는 폴리건 자료는 렌더링에 포함되지 않는다. 마지막으로 View-frustum의 평면에 대해서 OSPBox의 8개의 점이 분할 될 경우 현재 노드는 시야에 포함되는 부분과 포함되지 않는 부분이 같이 존재한다. 이 경우는 현재 노드에 포함하고 있는 폴리건만 렌더링에 포함시킨후, 2개의 자식노드들에 대해서 다시 위의 테스트를 반복한다.

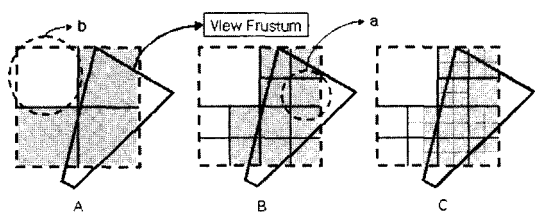

Fig. 6. Process of View-Frustum Culling.

이런 과정을 OSP 트리의 상위노드부터 순차적으로 마지막 노드까지 진행시키면 Fig. 6에서처럼 View-frustum에 포함되는 노드만 가시화된다.

3.2 OSP 트리를 이용한 Occlusion 컬링

Occlusion 컬링은 사용자의 시점에서 폴리건에 가려 보이지 않는 폴리건을 제거하는 방법이다. 이때 다른 물체를 가리는 물체를 Occluder라 하며 가려지는 물체를 Occludee라 한다.

3.2.1 Occluder 선택

Occlusion 컬링에서 가장 중요한 과정중에 하나가 좋은 Occluder를 선택하는 것이다. 좋은 Occluder를 선택하여야만 좀더 효율성 있는 컬링을 할 수 있다.

전처리 과정 없이 Occluder를 프레임 마다 선택하게 되면 렌더링 되는 모든 폴리건을 대상으로 검사를 해야 한다. 따라서 좋은 Occluder가 될 수 있는 폴리건을 미리 선택해 두어야 한다. 하지만, 실시간으로 입력되는 폴리건 이기 때문에 전처리 과정에서 좋은 Occluder를 선택하기가 어렵다. 따라서, 입력되는 폴리건 정보 중 크기를 우선시 하여 Occluder를 선택한다.

전처리 과정에서 선택한 Occluder 중에서 View-frustum 컬링에 의해 제외된 Occluder는 후보군에서 제외 시킨다. Occluder의 면적을 A, Normal 벡터를 N, 관찰자의 시선 벡터를 V, 관찰자에서 Occluder까지의 거리를 D로 정의할 때,

$$\frac{A(N \cdot V)}{\|D\|^2} \tag{1}$$

(1)의 값이 클수록 좋은 Occluder가 된다[5]. 좋은 Occluder는 디스플레이 되는 장면에서 크기가 클수록 좋은 Occluder가 된다.

3.2.2 Shadow Frusta[6]

Shadow Frusta는 관찰자의 시점에서 Occluder에 의해 가려지는 영역이다.

이 Shadow Frusta를 이용한 Occlusion 컬링은 Shadow Frusta를 형성하는 그림자 평면 안쪽에 노드가 위치하면 노드가 포함하는 폴리건은 렌더링 되지 않는다. 하지만, 생성된 Shadow Frusta에 대해 모두 계산을 하기 때문에 Occluder가 많아지면 계산량도 비례해서 복잡해 진다. 따라서 다음 항에서 소개될 SVBSP나 본 논문에서 제안하는 MSVBSP 트리를 이용하면 좀더 효율적인 계산을 할 수 있다.

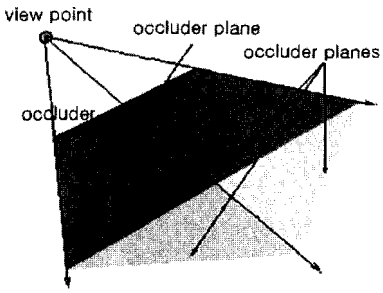

Fig. 7. Shadow Frusta.

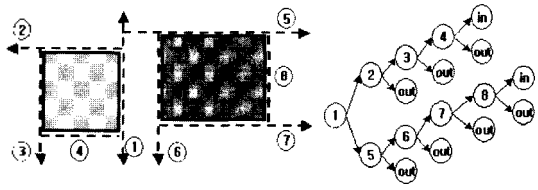

Fig. 8. Formation of shadow volume using BSP tree.

3.2.3 BSP 트리를 이용한 그림자 체적 생성 - Shadow Volume BSP(SVBSP) Trees[7,8]

다수의 Shadow Frusta로 Occlusion 컬링을 하기 위해 BSP 트리를 이용하여 그림자 체적을 생성한다.

Fig. 8에서 사각형은 Occluder이고 숫자는 그림자 체적을 형성하는 그림자 평면을 점선으로 표시하고 있다. 그리고 점선은 공간을 분할하는 형태를 보여 주고 있다. 사용자의 시점은 Occluder 앞쪽에 존재하고 있다. 모든 그림자 평면 앞쪽에 존재하면 Occluder에 가려지게 되도록 그림자 평면을 생성해야 한다. SVBSP 생성 과정은 분할 평면으로 좋은 그림자 평면을 선택하여 앞이나 뒤에 위치하는 그림자 평면으로 각각 분할 한다. 각각 하위 노드를 생성하고 재귀적인 방법으로 반복하여 모든 그림자 평면을 선택할 때까지 한다. 노드의 가시화 테스트 방법은 SVBSP를 이용하여 상위 노드부터 테스트를 한다. Fig. 8에서 OSP트리의 노드를 각 분할평면을 대상으로 테스트 하여 안쪽에 존재하면 가려져 보이지 않게 된다.

3.2.4 BSP 트리를 이용한 변형 그림자 체적 생성 - Modified Shadow Volume BSP(MSVBSP) Trees

Occluder가 많이 선택되면 Fig. 9처럼 Occluder끼리 서로 겹치고 포함하는 경우가 생기고, 이때 SVBSP가 복잡해 지고 분할 평면을 잘못 선택하면 부정확한 SVBSP가 생성된다.

따라서, 이런 문제를 해결할 수 있도록 MSVBSP트리를 제안한다. MSVBSP트리는 SVBSP트리를 생성

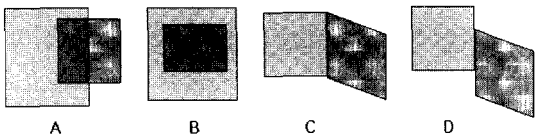

Fig. 9. Compounding of complex Occluders.

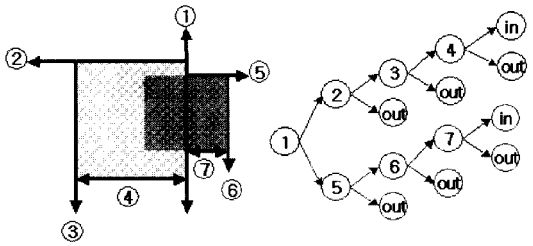

Fig. 10. Formation of MSVBSP tree (Fig. 10A).

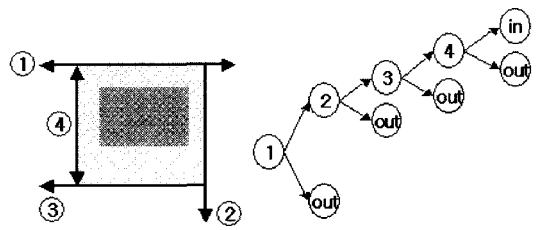

Fig. 11. Formation of MSVBSP tree (Fig. 10B).

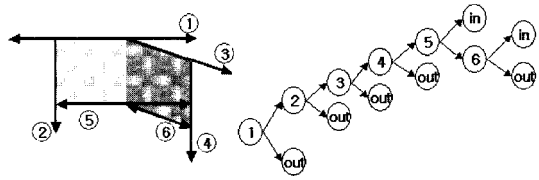

Fig. 12. Formation of MSVBSP tree(Fig. 10C).

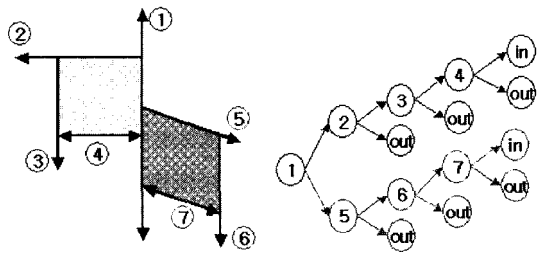

Fig. 13. Formation of MSVBSP tree(Fig. 10D).

할 때 전처리 과정에서 Occluder를 변형시키고, 분할 평면으로 그림자 평면을 선택할 때 몇 가지 제한조건을 두고 생성하는 것이다.

1. MSVBSP 트리를 생성할 때 Occluder 중에서 가장 좋은 점수를 받은 Occluder의 그림자 평면을 우선적으로 선택하여 MSVBSP 트리를 생성한다. 즉, 좋은 Occluder에서 만들어진 그림자 평면을 우선적으로 선택하여 MSVBSP 트리를 생성한다.

Fig. 10에서 ①번 분할 평면은 2개의 Occluder중 좋은 점수를 받은 더 큰 Occluder의 그림자 평면이다.

2. 선택된 분할 평면에 의해서 그림자 평면이 분할될 때 앞과 뒤 자식노드에 포함 시켜서 MSVBSP 트리를 생성한다. Fig. 10에서 ⑤, ⑦번 그림자 평면의 경우 ①번 분할 평면에 대해 분할 된다. 이 경우 ①번의 앞과 뒤 자식노드에 모두에 포함시켜서 진행시킨다.

3. 분할 평면을 선택할 때 1개의 Occluder를 구성하는 그림자 평면에서 우선적으로 선택한다. Fig. 10에서 ①평면이 선택되어 MSVBSP 트리를 생성한다면, 같은 Occluder를 형성하는 ②, ③, ④평면이 선택된 후 ⑤, ⑥, ⑦평면이 선택된다.

4. 3에서 하나의 Occluder에서 형성된 그림자 평면을 모두 선택하여 MSVBSP 트리를 생성한 후에도 앞쪽에 위치하는 평면이 존재 한다면, 앞쪽에 존재하는 모든 그림자 평면을 제거한다. Fig. 11에서 Occluder의 그림자 평면인 ①, ②, ③, ④를 모두 MSVBSP 트리로 생성한 후에도 ④의 앞쪽에 존재하는 작은 Occluder의 그림자 평면은 전에 선택된 Occlude에 의해 가려지게 되므로 모두 제거한다.

5. MSVBSP 트리의 전처리 과정으로 2개의 그림자 평면이 법선 벡터는 반대 방향이고 완전히 일치할 경우 2개의 그림자 평면 모두 제거한다. 나머지 그림자 평면은 하나의 Occluder를 구성하는 그림자 평면으로 통합된다. Fig. 12에서 접하는 그림자 평면은 제거되었고, 나머지 그림자 평면들은 통합되어 분할 평면 선택시 제약 없이 선택되고 있다.

6. MSVBSP 트리의 전처리 과정으로 2개의 Occluder가 접하는 그림자 평면이 완전히 일치하지는 않고 같은 평면 상에 존재하는 경우는 좋은 Occluder의 그림자 평면은 남겨두고 다른 그림자 평면만 제거한다. Fig. 13에서는 ①에서 접하는 평면 중 하나만 제거 되었다.

3.2.5 MSVBSP 트리을 이용한 Occlusion 컬링

MSVBSP 트리를 이용하여 Occlusion 컬링을 할 때 OSP 트리의 상위 노드부터 테스트를 한다. 노드가 MSVBSP 트리의 안쪽에 존재한다면 폴리건은 모두 Occluder에 가려져 보이기 않기 때문에 렌더링 되지 않

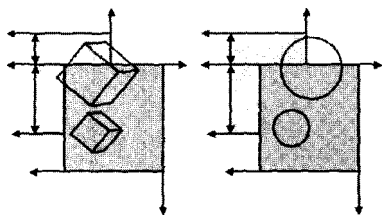

Fig. 14. Occlusion culling using MSVBSP tree.

는다. 노드가 MSVBSP 트리의 밖에 존재한다면 폴리건은 모두 보이기 때문에 렌더링 된다. 노드가 걸치게 되면 하위노드로 이동하여 위의 과정을 반복하게 된다.

Fig. 14에서 화살표는 MSVBSP 트리를 구와 직육면체는 OPS 트리의 노드를 나타낸 것이다. MSVBSP 트리를 이용하여 *Occlusion* 컬링을 테스트 할 때도 두가지 방법이 있다. Fig. 14에서의 왼쪽 그림처럼 OSPBox의 8점을 가지고 테스트를 할 수 있다. 하지만 위쪽에 큰 OSPBox의 8점 모두 MSVBSP 트리를 이용하여 Occlusion 컬링 테스트를 하면 모두 안쪽에 있어서 컬링이 된다. 그러나 실제로는 완전히 Occluder에 가려져 있지 않기 때문에 컬링 되어서는 안 된다. 따라서, Fig. 14에서의 오른쪽 그림처럼 OSPBox를 이용하지 않고 OSPBox를 포함하는 구를 사용하여 테스트 하면 Occluder에 완전히 가려지는 노드만 컬링이 된다. 본 논문에서는 두번째 방법인 구를 이용한 컬링을 하였다.

4. 무한 순차 입력 형상 시스템 구현 및 결과

4.1 시스템 구현 및 순서도

Fig. 15에서처럼 Dual Projection 시스템은 2개의 스크린이 최소의 공간을 이루기 위해서 90°로 이루어져 있다. 이 시스템은 각각의 스크린에는 2대의 프로젝터가 있고, 화면을 입체로 볼 수 있도록 패시브 스테레오 영상을 구현한다. 사용자의 시점에 따라 보여지는 장면이 바뀌는 윈도우 투영 방식을 사용함으로써 몰입감을 증대 시키고 있다. 이 시스템에서 사용자는 직접 트랙커를 이용하여 공간에서 모델링을 하면서 자신이 만든 모델을 3D 영상으로 볼 수 있다.

Fig. 16의 전체 시스템 순서는 다음과 같다.

프로그램이 시작하면 OSP 트리의 전처리 과정으로 공간을 계층적으로 분할하게 된다. 유저 입력을 받아서 모델을 생성하고 입력되는 모델을 가지고 실시간으로 OSP 트리를 생성하게 된다. 또한, OSP 트리를

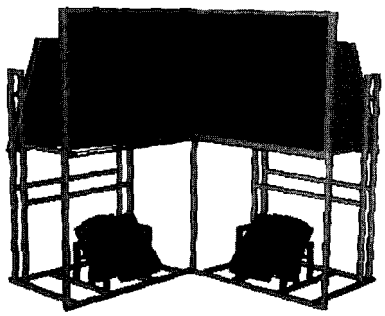

Fig. 15. Dual projection system.

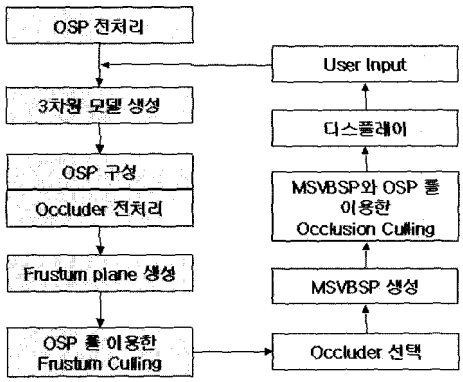

Fig. 16. Flow chart of the whole system.

만들며 좋은 Occluder를 미리 후보로 선정한다. 다음은 매 프레임의 시점에서 OSP 트리를 이용하여 View-frusrum 컬링을 한다. 현재 시점에서 좋은 Occluder를 선택하여 MSVBSP 트리를 생성한다. MSVBSP 트리와 OSP 트리를 이용하여 Occlusion 컬링을 한 후 가시화 되는 폴리건을 렌더링 한다. 사용자 입력을 받은 후 계속해서 재귀적으로 반복한다.

4.2 실시간 3차원 모델링

트랙커를 사용자 입력으로 설정하고 매 프레임 마다 트랙커의 위치와 방향을 추적하여 간단한 모델링을 한다. 간단한 모델링 방법을 통해 실시간으로 입력되는 폴리건 정보를 계층적으로 자료구조화 하고 가시화 테스트를 하는 것에 중점을 두었다. Fig. 17에서처럼 트랙커를 통해 얻은 위치와 방향을 가지고 x축 방향이 법선 벡터인 폴리건을 생성한다.

4.3 OSP 트리와 기존의 Octree의 실시간 입력 결과 비교

기존의 방식인 Octree와 OSP 트리를 실시간으로

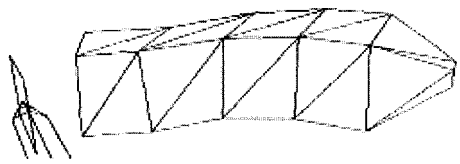

Fig. 17. Modeling using realtime input.

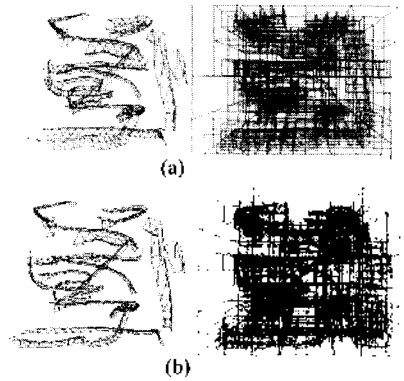

Fig. 18. Modeling usingv (a) Octree, (b) OSP tree.

입력되는 자료를 사용하여 자료구조로 구성한다. 두 번의 실험을 할 때 트랙커의 위치와 방향 입력을 같게 할 수 없기 때문에 한번의 입력을 파일로 저장한다. 그리고, Octree와 OSP 트리를 이용하여 작성한 프로그램에서 저장해 둔 입력을 매 프레임 마다 하나씩 읽어 들여 사용하였다.

Fig. 18(a)와 (b)는 모델링 한 것을 테스트 한 것이다. 사용자의 입력을 저장한 후 각각 다시 불러 들어 사용하였기 때문에 비슷하게 모델링이 된 것을 알 수 있다.

Fig. 19에서는 실시간으로 입력되는 폴리건을 Octree와 OSP 트리로 구성할 때 프레임 수를 측정한 것이다. 최고 속도는 Octree가 OSP 트리 보다 빠르고, 이 부분은 새로운 폴리건이 입력되지 않고 렌더링만 되는 부분이다. 이유는 Octree는 항상 새로운 모델이 입력 될 때마다 최적의 조건으로 자료구조를 재구성하기 때문이다. 하지만 Octree는 새로운 폴리건이 입력되어 자료구조를 재구성 할 때는 초당 프레임수가 많이 낮아진다. OSP 트리의 경우 폴리건이 입력되지 않는 부분에서의 프레임 수는 약간 떨어지지만, 실시간으로 입력되는 폴리건을 자료구조로 재구성 할 때는 Octree만큼 프레임수가 떨어지지 않는다. 따라서 OSP 트리는 Octree에 비해 실시간으로 입력되는 폴리건을 자료구조로 구성하고 렌더링 하기에 효율적이다.

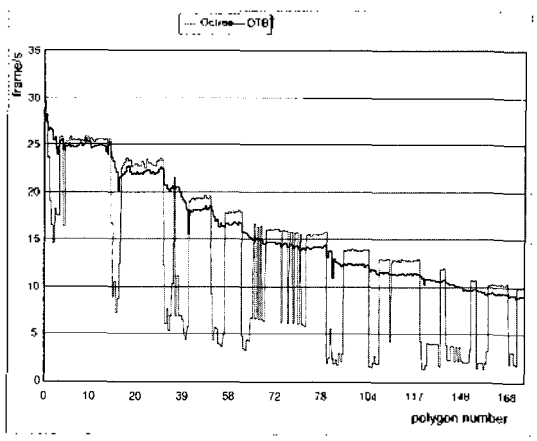

Fig. 19. The comparison of Octree and OSP tree(Result 1).

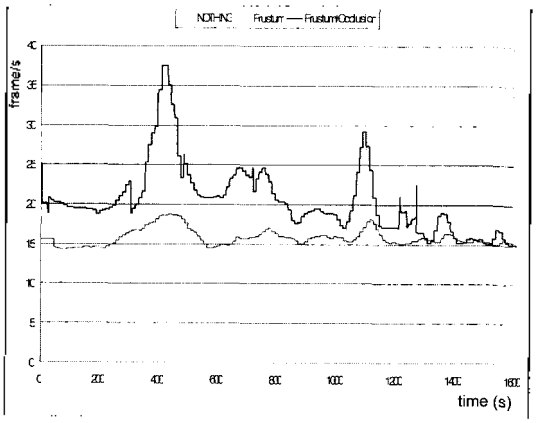

Fig. 20. Fustum culling and Occlusion culling of OSP tree.

Fig. 20의 실험 결과는 폴리건 입력은 하지 않는 상태에서 시점을 바꾸며 이동하면서 프레임 수를 저장한 것이다. View-fustum 컬링만 한 경우는 OSP 트리 시야 밖으로 나갔을 경우 높은 프레임 수를 나타낸다. View-fustum 컬링과 Occlusion 컬링을 모두 한 경우 거의 모든 구간에서 높은 프레임 수를 보여준다. 일부의 구간에서는 컬링 되는 폴리건의 양보다 계산 시간이 많아 낮은 프레임 수가 나오는 구간도 있다.

앞의 실험에서는 3000개 이하의 폴리건에서 실험을 하였지만 2번째 실험에서는 시스템에서의 한계 폴리건 개수까지 입력 시키면서 실험을 하였고 실험 방법은 앞에서의 실험 방법과 동일하다. 전체 실험구간을 특성있는 3부분으로 나누어서 도시화 하였다.

Fig. 21에서는 작은 수의 폴리건에서의 Octree와 OSP 트리를 비교한 것이다. 이 구간은 처음 구간으로

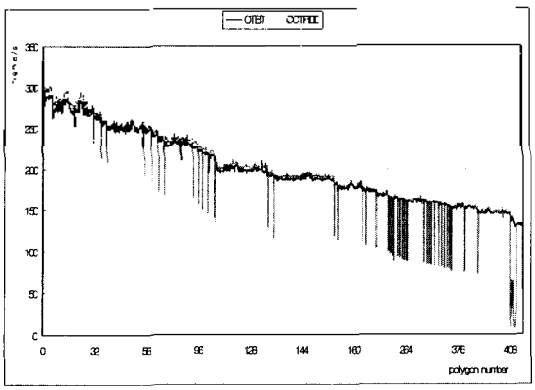

Fig. 21. The comparison of Octree and OSP tree 1(Result 2).

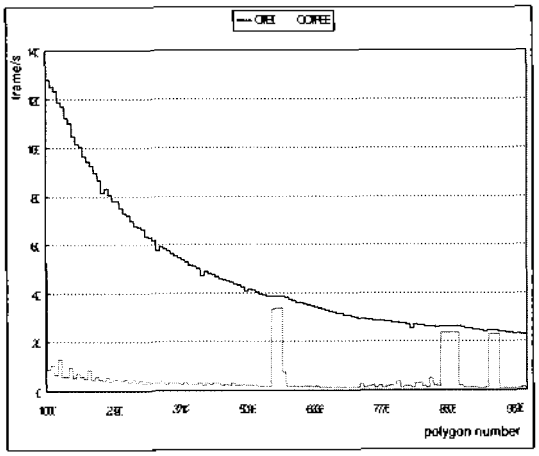

Fig. 22. The comparison of Octree and OSP tree 2(Result 2).

총 폴리건 수도 작고 입력되는 폴리건의 수도 작은 구간이다. OSP 트리는 실시간으로 입력이 되고, Octree의 경우 입력될 때 약간의 프레임 수가 낮아 지지만 거의 일정하게 입력되고 있다.

Fig. 22에서는 1000~10000개 사이의 폴리건이 입력될 때의 비교이며, 이 구간은 급격히 많은 폴리건이 입력되고 있다. Octree의 경우 폴리건이 입력될 때마다 계속해서 자료구조를 생성해야 하기 때문에 프레임 수가 낮아져, 화면이 끊기면서 제대로 모델링 작업을 할 수 없다. OSP 트리의 경우 폴리건이 증가함에 따라 프레임 수는 서서히 낮아진다. 이는 폴리건을 렌더링하는 시간이 점차 오래 걸려서 낮아지지만 자료구조를 형성하는 시간은 차이가 없다.

Fig. 23에서는 10000개 이상의 폴리건에서 폴리건이 계속해서 입력될 때의 그래프이다. OSP 트리의 경우 앞의 두 그래프와 마찬가지로 폴리건 수가 많아 짐에 따라 렌더링 하는 시간이 길어질 뿐이지 실시간으로 입력되는 폴리건을 자료구조로 생성하는 데는 많은 시간이 걸리지 않는다. 하지만 Octree의 경우 폴리건이 입력되면 시스템이 움직이지 않을 정도로 자료구조를 다시 생성하는데 오랜 시간이 걸리게 됨을 알 수 있다. 모든 구간에서 OSP 트리를 사용한 것이 Octree보다 실시간으로 입력되는 폴리건을 자료구조로 생성하는데 더 좋은 성능을 나타내고 있음을 앞의 2번의 실험을 통해서 알 수 있다.

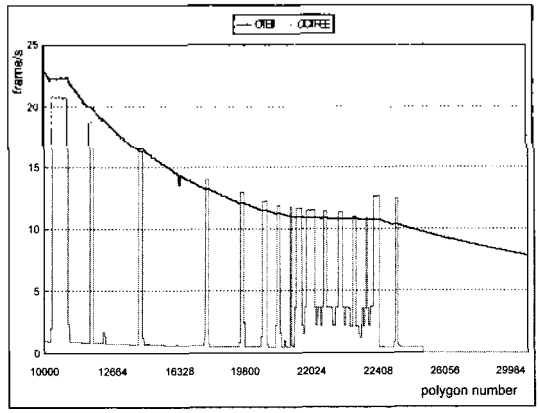

Fig. 23. The comparison of Octree and OSP tree 3(Result 2).

5. 결 론

OSP 트리를 제안한 목적은 실시간으로 입력되는 자료로 모델링을 하면서 효율적인 가시화 컬링을 할 수 있는 자료구조로 만드는 알고리즘의 필요성 때문이다.

OSP 트리는 모델링 된 자료를 전처리 과정을 거쳐 최대한 효율적인 자료구조로 만드는 Octree, BSP 트리와는 다른 목적을 가지고 있다. 저장하고 있는 폴리건을 렌더링 할 때, OSP 트리는 전처리 과정을 거쳐 만든 Octree나 BSP 트리보다 적합하지 않은 자료구조 형태이다. 즉, OSP 트리는 크기가 제한된 공간이기 때문에 제한된 공간에 비해 입력된 폴리건의 분포 형태가 매우 크거나 또는 매우 작은 경우는 비효율적이다.

모델링 된 폴리건을 렌더링 하면서 새로운 폴리건이 계속해서 추가되는 시스템이 있다. 이런 시스템의 경우 Octree나 BSP 트리는 새로운 폴리건을 포함하는 자료구조를 생성할 때, 많은 시간이 걸려 실시간 렌더링을 할 수 없다. 이런 점을 개선하기 위해 렌더링을 위한 최적화된 자료구조의 형태는 아니지만 실시간으

로 입력되는 폴리건을 자료구조화 할 때 최적화 된 것이 OSP 트리이다.

실시간으로 입력되는 폴리건을 효율적인 자료구조로 생성하더라도, 많은 수의 폴리건이 입력되면 실시간 모델링을 할 수가 없다. 따라서 생성된 자료구조를 이용하여 효율적인 가시성 테스트를 할 수 있어야 한다. OSP 트리의 경우 생성 방식은 BSP 트리와 유사하고, 생성된 후의 노드 정보는 Octree와 유사한 형태를 가짐으로써 View-frustum 컬링과 Occlusion 컬링을 효율적으로 계산할 수 있다.

OSP 트리를 이용하여 View-Frustum 컬링을 하고, MSVBSP 트리를 사용하여 효율적인 그림자 체적을 생성한 후 Occlusion 컬링을 하게 되면 많은 수의 폴리건이 입력 되더라도 실시간 모델링을 할 수 있다.

그러므로 OSP 트리는 실시간으로 입력되는 폴리건 정보를 자료구조화 하는데 효율적이며, 계층적으로 구성된 자료구조를 통해 가시화 테스트를 할 때에도 Octree와 같은 노드 정보를 가지고 있으므로 효율적인 테스트를 할 수 있다. 따라서 OSP 트리는 사용자의 입력정보를 통해 실시간으로 모델링을 하면서 렌더링을 하는 시스템에서는 효율적인 자료구조의 형태라 할 수 있다.

하지만, Occlusion 컬링 알고리즘의 경우 입력되는 폴리건이 너무 작은 것들로만 입력이 되면 MSVBSP 트리를 생성하여 Occlusion 컬링을 하더라도 계산되는 시간에 비해 제거되는 폴리건의 개수가 적기 때문에 오히려 역효과를 보는 경우가 발생한다. 향후 이런 작은 폴리건 자료에도 효율적인 Occlusion 컬링에 관한 연구가 필요하다.

감사의 글

이 논문은 2002년도 중앙대학교 학술연구비 지원에 의한 것임.

참고문헌

1. Keefe, D. and Acevedo, D., "CavePainting: A Fully Immer-sive 3D Artistic Medium and Interactive Experience," *Proceedings of the 2001 Symposium on Interactive 3D Graphics*, pp. 85-93, March 2001.
2. Teller, J. and Sequin, H., "Visibility Preprocessing for Inter-active Walkthroughs," *Proceedings of SIGGRAPH'91*, Vol. 25, No. 4, pp. 61-69, July 1991.
3. gametutorials 홈페이지, "www.gametutorials.com/Tutorials/OpenGL/Octree.htm".
4. 류중현, 김영우, 김덕수, "인터넷을 통한 3D 형상 데이터의 실시간 전송을 위한 효율적인 Octree 인코딩 방법에 관한 연구," 한국CAD/CAM학회 논문집, 제7권, 제4호, pp. 262-268, 2002.
5. Coorg, S. and Teller, S., "Real-Time Occlusion Culling for Models with Large Occluders," *In Proceeding 1997 Symp-osium on Interactive 3D Graphics*, pp. 83-89, April 1997.
6. Hudson, T. and Manocha, D., "Accelerated Occlusion Culling using Shadow Frusta," *In Proc. 13th Annual ACM Symposium on Computational Geometry*, pp. 1-10, June 1997.
7. Chin, N. and Feiner, S., "Near Real-time Shadow Generation using BSP Trees," *Proceedings of SIGGRAPH'89*, Vol. 23, No. 3, pp. 99-106, July 1989.
8. Greene, N., "Hierarchical Polygon Tiling with Coverage Masks," *Proceedings of SIGGRAPH '96*, pp. 65-74, August 1996.

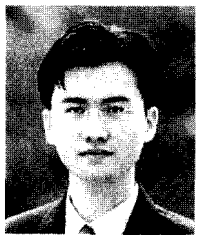

표 종 현

2001년 중앙대학교 기계공학과 학사
2003년 중앙대학교 첨단영상대학원 영상공학과 석사
관심분야: Virtual Environments, Steer-by-wire

채 영 호

1989년 중앙대학교 기계공학과 학사
1994년 SUNY at Buffalo 기계공학과 석사
1997년 Iowa State University 기계공학과 박사
1989~1992년 (주)삼성전기 CAD/CAM 실 연구원
1998년 CASE Virtual Prototyping Lab. Consultant
1998~1999년 중앙대학교 기계공학부 조교수
1999년~현재 중앙대학교 첨단영상대학원 부교수
관심분야: Haptics, Virtual Environments, IBM/IBR