MMORPG 게임엔진의 성능개선을 위한 분할공간에서의 충돌검출

이 승 욱†

요 약

최근 하드웨어의 급속한 발전으로 3차원 그래픽의 적용 분야도 다양화 되어가고 있다. 3차원 가상도시를 배경으로 하는 3D MMORPG (Massive Multi-play Online Role Playing Game)와 같은 게임을 설계하기 위하여 필요한 세부 기술은 다양한 이론이 병합되어야 한다. 3D MMORPG 게임엔진은 거대한 3차원 도시의 수많은 빌딩과 개체를 실시간으로 빠르게 처리되어야 하기 때문에 렌드링의 처리뿐만 아니라 속도에 영향을 미치는 많은 요소를 가지고 있다. 이러한 게임엔진의 설계에서 중요하게 다루어지는 것은 처리 속도이다. 기존의 3DMMORPG에서 충돌검출의 방법으로 경계상자를 적용하지만 이 방법은 거대지형에서의 충돌 검출 시 속도가 느려지기 때문에 적용하기에는 바람직하지 않다. 따라서 본 논문은 거대지형상의 3D MMORPG 게임에서 발생되는 충돌검출 속도를 향상시키고자 한다. 즉 이러한 처리에서 본 논문은 다음과 같이 제시한다. 첫째 폴리곤의 충돌검사를 모두 하지 않고 빠른 시간에 충돌검출을 판단할 수 있다. 둘째 경계상자의 충돌검출에 대한 비용이 3차원 개체 개수에 대해 비례하여 증가하는 데에 대한 개선 방법을 제시한다. 그 처리 과정은 3D MMORPG 넓은 가상공간을 동적으로 처리하기 위해서는 제한적 OSP를 사용하여 공간분할을 한다. 분할된 3차원 공간을 계층적 경계상자를 이용함으로써 충돌검출에 필요한 개체를 검색하고 이를 통하여 충돌검출 속도를 개선시킬 수 있을 것이다.

A Collision detection from division space for performance improvement of MMORPG game engine

Sung Ug Lee†

ABSTRACT

Application field of third dimension graphic is becoming diversification by the fast development of hardware recently. Various theory of details technology necessary to design game such as 3D MMORPG (Massive Multi-play Online Role Playing Game) that do with third dimension. Cyber city should be absorbed. It is the detection speed that this treatise is necessary in game engine design. 3D MMORPG game engine has much factor that influence to speed as well as rendering processing because it express huge third dimension city's grate many building and individual fast effectively by real time. This treatise may get concept about the collision in 3D MMORPG and detection speed elevation of game engine through improved detection method. Space division is need to process fast dynamically wide outside that is 3D MMORPG's main detection target. 3D is constructed with tree construct individual that need collision using processing geometry dataset that is given through new graph. We may search individual that need in collision detection and improve the collision detection speed as using hierarchical bounding box that use it with detection volume. Octree that will use by division octree is used mainly to express rightly static object but this paper use limited OSP by limited space division structure to use this in dynamic environment. Limited OSP space use limited space with method that divide square to classify typically complicated 3D space's object. Through this detection, this paper propose follow contents, first, this detection may judge collision detection at early time without doing all polygon's collision examination. Second, this paper may improve detection efficiency of game engine through and then reduce detection time because detection time of bounding box's collision detection.

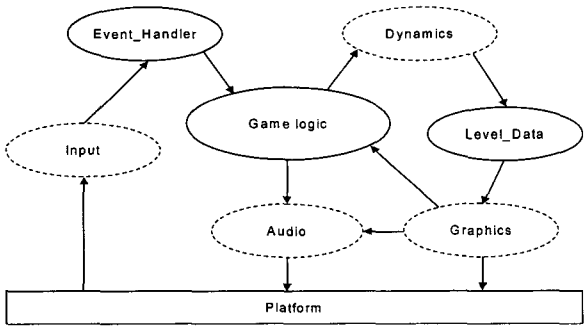
키워드 : 충돌검출(Collision Detection), 계층적 경계상자(Hierarchical bounding boxoctree), 옥트리

1. 서 론

게임은 크게 게임엔진과 게임 컨텐츠로 구성된다. 게임엔진은 특정 게임 컨텐츠 포맷을 처리하는 컴포넌트의 집합으로 다개 게임의 장르에 맞는 게임 컨텐츠의 틀을 가지고 있다. 따라서 게임 엔진은 레벨포맷, 오디오처리, 이벤트 핸들러, 입력처리 등의 핵심 로직을 모두 포함한다. 다음 (그림 1)은 게임엔진의 중요한 세부 요소를 나타내고 있다. 게임의 속도향상을 위해 고려해야 할 사항은 다음과 같다. 첫 번째로 3차원 게임에 사용되는 모델의 폴리곤 수를 최대한 줄여야 한다. 실사와 같은 영상을 얻기 위해서는 폴리곤의 수가 많으면 많을수록 좋겠지만 게임과 같은 환경에서는

† 정 회 원 : 동아대학교 전기전자컴퓨터 BK21 교수
논문접수 : 2003년 5월 21일, 심사완료 : 2003년 8월 20일

무엇보다도 속도가 문제가 되기 때문에 될 수 있으면 폴리곤 수를 줄여야 한다. 두 번째로 미리 계산할 수 있는 것들은 미리 계산하여 처리한다. 광원의 처리 같은 경우 이를 계산하는 것은 많은 시간을 필요로 하기 때문에 실시간 3차원 게임에 사용하기에는 적당하지 않다. 세 번째로 공간을 쪼개서 렌더링 파이프 라인에 들어가는 물체의 수를 줄이고 카메라의 뷰 볼륨 안에 있는 물체만 렌더링 하도록 한다. 이러한 부분들은 게임에 있어 그래픽처리 성능을 통한 속도를 향상시키는 아주 중요한 요소이다[1, 3, 12].



(그림 1) 게임의 구조

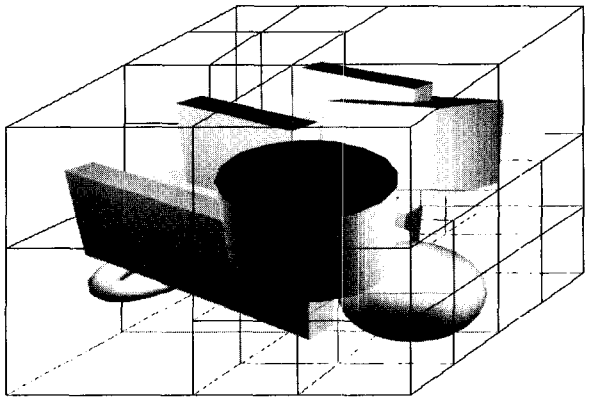
본 논문은 3D MMORP에서 동적 환경의 처리 속도를 높이기 위해 제한적 OSP와 계층적 경계상자를 적용하고자 한다.

이를 위해서 기존의 MMORPG에서의 충돌검출 방법을 고찰하고 한계점을 제시한다. 이것의 해결 방법으로 제한적 OSP와 계층적 경계상자를 이용한 구현과정을 제시하고자 한다. 이 때 기존의 계층적 경계상자를 MMORPG와 같은 게임엔진 설계시 고려되어야 하는 문제점에 관하여 분석하고 이를 효과적으로 적용시키는 방법을 제시하려 한다.

2. 충돌 검출에 대한 이론적 배경

2.1 3차원 가상공간의 처리

3차원 가상도시를 배경으로 하는 3D MMORPG(Massive Multi-play Online Role Playing Game)와 같은 게임을 설계하기 위하여 필요한 세부 기술은 다양한 이론이 병합되어야 한다. 이러한 게임엔진의 설계에서 중요하게 다루어지는 것은 처리 속도이다. 3D MMORPG의 주 처리 대상이 되는 넓은 외부 영역을 동적으로 빠르게 처리하기 위해서는 공간분할이 필요하다. 일반적인 옥트리의 자료 구조는 화면을 단순히 분할하는 것이 아니라 폴리곤의 개수를 기준으로 균등 분할함으로써, 메모리의 오용이 적고 클리핑 및 충돌검출시 폴리곤을 정확하고 빠르게 검출해 낼 수 있다. 이렇게 분할된 옥트리를 점의 각도를 구해서 화면 시야 여부를 결정하고, 8개의 점이 모두 다 시야 안이라면 그 안의 옥트리를 더 이상의 클리핑도 필요 없이 렌더링에 필요한 노드로 처리되어 검출한다. 8개의 점이 모두 밖에 있다면 그 안의 옥트리는 화면에 보이지 않는 것이 확실하므로 노드가 검출되지 않고 처리에서 제외시켜버린다. 앞의 2가지의 경우가 아니라면 다시 하위 옥트리로 내려가 다시 앞의 2가지 분할이 끝나면 옥트리는 모든 노드와 포함 연산을 통해 보여지는 노드를 검출한다.



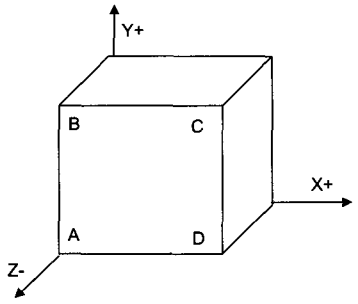
(그림 2) 주위의 입방체로부터 세부수준으로 분할된 모습

일반적인 옥트리의 경우 정적인 입체나 지형처리에 적합한 자료 구조이다. 실시간 적으로 발생되는 동적인 처리시 빈번하게 발생되는 옥트리의 재 구축 회수는 게임엔진의 처리속도를 느리게 만들고 넓은 영역을 옥트리를 이용하여 트리를 구축해야 하기 때문에 많은 메모리를 필요로 한다. 옥트리는 게임과 같은 분야에 적용하기 위해서는 동적인 객체의 가시성 판단이나 각 개체의 부착목록을 저장하기 위한 자료 구조로 사용된다. 여기에 제시된 공간분할 방법인 OSP는 분할된 3차원 공간을 고정볼륨 바운드를 구축하기 위한 분할 자료로 사용하게 된다. 기존의 옥트리를 동적인 환경에 이용하기 위해서는 옥트리의 처리 개념을 변경시킬 필요가 있다. 제한적 옥트리 분할 방법으로 사용되는 제한적 OSP는 복잡한 3D 공간상의 물체를 정형적으로 분류하기 위한 분할 방법이다. 넓은 3차원 공간을 네모난 구역으로 나누기 위해 사용되는 방법이다. 광범위한 외부 지형 자료를 최소한의 처리에 필요한 데이터로 국한 시켜 처리를 단순화 시키는 방법으로 사용되고 있다[1, 6-8].

2.2 3차원 게임에서의 충돌검출

충돌검출을 하기 위한 MMORPG 외에도 일반적인 3차원 게임에서 많이 사용되고 있는 방법이 경계상자를 이용한 충돌검출 방법이다. 경계상자를 구하기 위한 처리 방법은 객체 전체의 최대, 최소 x, y, z 값들을 찾아야 한다. 객체의 각 정점들을 돌아가면서 비교하면 최대, 최소값들을 찾을 수 있다. 상자의 꼭 지점을 A, B, C, D, E, F, G라고 하면 (그림 3)과 같다, 다음과 같이 최대/최소값들로 얻은 값들을 각 꼭 지점들의 좌표로 배정할 수 있다.

A = (minx, miny, minz)
B = (minx, maxy, minz)
C = (maxx, maxy, minz)
D = (maxx, miny, minz)
E = (minx, miny, maxz)
F = (minx, maxy, maxz)
G = (maxx, maxy, maxz)
H = (maxx, miny, maxz)



(그림 3) 경계상자

이를 이용해서 공간 대각선들의 중점, 즉 상자의 중점을 구하는 코드는 다음과 같다. 중점 공식은 주어진 두 점 A(x1, y1, z1)과 B(x2, y2, z2)에 대해, A와 B를 잇는 선분의 중점의 좌표는 다음과 같다.

$$[(x1 + x2) / 2, (y1 + y2) / 2, (z1 + z2) / 2]$$

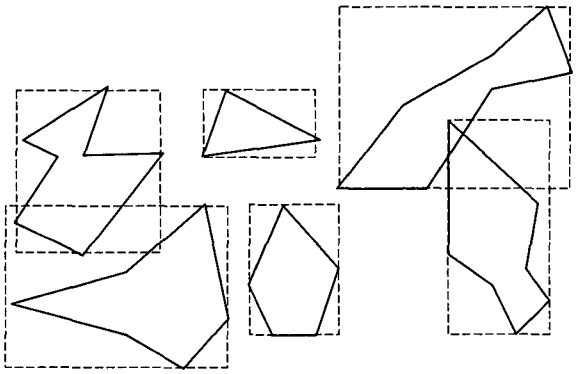
```
center.x = (A.x + G.x)/2 ;
center.y = (A.y + G.y)/2 ;
center.z = (A.z + G.z)/2 ;
```

경계 구의 반지름은 중점과 가장 먼 정점과의 거리를 사용해서 구한다. 반복적으로 객체의 정점들을 차례로 탐색하고, 각 정점에 대해 중점과 그 정점 사이의 거리를 구한 다음, 그것이 현재의 최대 거리보다 크면 현재의 최대 거리를 그 거리로 갱신한다. 루프가 끝났을 때의 최대 거리가 바로 경계 구의 반지름이 되는 것이다 경계 구의 중점과 반지름이 결정된다면 3D MMORPG 게임엔진 설계에 필요한 충돌검출과 같은 분야에 활용할 수 있다. 충돌검출 형태는 두 경계 구를 통하여 충돌 여부를 매우 빠르고 쉽게 판단할 수 있다[2, 3].

2.3 충돌검출의 처리 절차

2.3.1 3차원 게임에서의 충돌검출

충돌검출은 하나의 개체를 이루는 다각형 덩어리들에 대해 개별적인 경계구들을 들어 충돌검출을 할 수 있다. 3차원 게임환경에서는 3차원 개체의 충돌이 동시 다발적으로 일어 나지 않는다면 경계상자를 이용하는 방법은 충분한 효과를 얻을 수 있다. 3차원 대상들의 경우에 대부분의 경우 서로 충돌을 하는 경우가 그리 많지 않고, 충돌하는 경우에 있어서도 거의 동시다발적으로 일어나지 않으며, 3차원 개체의 충돌에 있어서 극히 일부의 개체들간의 충돌이 일어나는 경우가 많다. 이러한 점을 이용하여 모든 3차원 개체들에 대해 각각의 면을 구성하는 삼각형끼리의 충돌검사를 모두 하지 않고서도 빠른 시간에 충돌 체크를 할 수 있다. 3차원 각각의 개체에 대하여 경계상자를 설정하고, 경계상자끼리의 충돌을 검출한다. 경계상자끼리 충돌을 하지 않은 경우에는 그 개체들은 충돌하지 않는다고 볼 수 있다. 그러나 경계상자끼리 충돌 했다면, 이것은 충돌할 가능성이 있는 것이다. 경계상자를 이용하면 필요이상으로 자세하게 충돌검사를 행하기 전에 비교적 멀리 떨어져 있는 경계 상자간의 충돌검출을 피할 수 있고, 따라서 충돌을 감지하는데 소요되는 시간을 크게 줄일 수 있다. 경계상자의 모양은 경계 상자간의 충돌여부를 쉽게 검사할 수 있는 것으로 선택하여 사용하게 된다. 경계상자의 모양은 직육면체와 공 모양의 상자를 이용할 수 있다.



(그림 4) 경계상자의 이용

직육면체의 경우 물체가 회전할 때마다 경계상자의 위치가 변하는 단점이 있지만, 경계 상자간의 위치 순서로 정렬을 쉽고 빠르게 충돌을 감지할 수 있다는 장점이 있다. 경우에 따라서는 직육면체와 공 모양의 경계상자를 모두 사용하여 이중으로 충돌검출을 행할 수도 있다. 경계상자를 이용하여 충돌을 감지하는데 걸리는 시간 복잡도를 살펴보면 다음과 같다. 3차원 개체 n개이고, 각각의 3차원개체에 대해 면을 이루는 삼각형의 개수가 평균 m이라면, 경계상자를 이용하지 않을 경우 충돌검출에 필요한 시간복잡도는 $O(n^2m^2)$이 되지만, 경계상자를 이용하는 경우 $O(n^2 + \alpha)$가 된다. 여기서 α는 경계 상자간의 충돌했을 경우 다시 자세하게 삼각형을 중심으로 충돌검출을 행하는 시간을 말한다[6, 12, 13].

2.3.2 실질적인 충돌검출을 위한 처리

객체들의 정확한 충돌검출방법으로 광선 검출방법을 사용한다. 광선과 벡터를 이용해 표현한다. 벡터는 시작점과 방향을 표시한다.

$$P = S + t \times V \quad (1)$$

P : 광선 위의 임의의 한 점(point)
S : 광선의 시작점(start point)
t : 변수, 광선 위에서의 상대적 위치를 표시한다.
(0 <= t < 무한대)
V : 방향벡터 V(x, y, z)

(t > 0) 식으로 광선은 위와 같다. t가 0이면 광선의 시작점이 된다. t에 따라서 대응되는 광선 위의 점을 얻는다. P, S, V은 모두 3차원 벡터이다. 이것들을 이용해서 광선과 평면과의 교차점, 광선과 개체와의 교차점을 계산한다. 움직이는 구와 평면의 교차점 찾는다.

Xn 내적 X = x : 벡터. 평면 위의 한 점이다.
Xn : 평면의 법선 벡터
d : 좌표원점에서부터 평면까지의 최단 거리 나타내는 식이다.

$$Ax + By + Cz + D = 0$$
$$Xn \text{ 내적 } P = d \quad (2)$$

광선이 평면 위의 임의의 점을 뚫고 지나간다면, 식 (1), 식 (2)를 이용하여 만나는 점에서 다음과 같은 식이 성립한다.

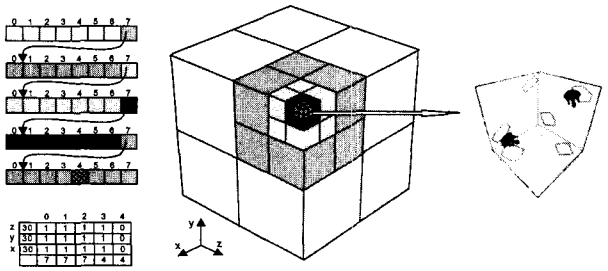
Xn 내적 P = d, P 대신에 식 (1)을 대입하면, 교점을 구할 수 있다.

Xn 내적 X = x가 나오는 경우는 광선의 뒤쪽에서 평면과 만나는 것을 말한다. 함수의 인자를 통하여 평면과 광선의 시작점과 방향 벡터간의 충돌이 일어난 곳까지의 거리를 저장할 인자, 즉 충돌지점에서의 법선 값을 저장할 인자가 있다. 인자를 통하여 충돌을 검출한다[2, 3, 5, 12].

3. 동적인 환경에서 제한적 OSP의 적용과 구현

3.1 제한적 OSP의 공간분할 방법

옥트리의 공간분할 방법인 OSP(Octree Space Partitioning)는 일반적으로 3차원의 정적인 공간을 정확히 표현을 위해 사용되는 자료 구조이다. OSP의 분할 형태는 3차원 개체를 기준으로 그 개체의 다각형이 주어진 노드의 경계 입방체 안에 들어가는지를 판단하는 기준으로 사용한다. 그러나 제한적 OSP는 3D MMORPG 게임엔진 설계에 있어 주처리 대상이 되는 3차원 개체 및 지형을 실시간적으로 처리하기 위한 자료처리 방법이다. 실시간적으로 발생되는 동적인 처리시 빈번하게 발생되는 옥트리의 재구축 회수는 게임엔진의 처리속도를 느리게 만들고 넓은 영역을 옥트리를 이용하여 트리를 구축해야 하기 때문에 많은 메모리를 필요로 한다. 옥트리를 게임과 같은 분야에 적용하기 위해서는 동적인 객체의 가시성 판단이나 각 개체의 부착목록을 저장하기 위한 자료 구조로 사용되기도 한다. 그러나 옥트리를 동적인 객체의 처리 즉 MMOPRPG와 같은 게임에 사용하기에는 부적합하다. 이에 대해서 제한적 OSP는 분할된 3차원 공간을 고정볼륨 바운드를 구축하기 위한 분할 자료로 사용하게 된다. 기존의 옥트리를 동적인 환경에 이용하기 위해서는 옥트리의 처리 개념을 변경시킬 필요가 있다. 제한적 옥트리 분할 방법으로 사용되는 제한적 OSP는 복잡한 3D 공간상의 물체를 정형적으로 분류하기 위한 분할 방법이다. 넓은 3차원 공간을 네모난 구역으로 나누기 위해 사용되는 방법으로 광범위한 외부 지형 자료를 최소한의 처리에 필요한 데이터를 생성시킨다. 실 시간으로 처리될 단위 노드의 값은 게임의 페이지 화면 단위가 된다 제한적 OSP를 이용하여 분할된 노드를 계층적 경계상자를 이용하여 처리 대상을 트리로 구축해야 한다. 분할된 공간에 대한 개체에 관한 정보는 신 그래프로 주어져 기하학 데이터 셋을 통하여 얻을 수 있다. 이것을 이용하여 렌드링에 필요한 3차원 개체를 계층적 경계상자(Hierarchical Grid Bounding Box)를 적용하여 충돌검출을 처리한다. 신 그래프의 정보도 또한 트리구조 이루어져 있어 개체의 위치 정보를 가지고 있다. 거대한 3차원 공간을 제한적 OSP를 사용함으로 3차원개체의 움직임에 따라 빈번하게 발생되는 옥트리의 재구성 회수 및 많은 양의 옥트리 데이터 처리량을 줄일 수 있다. 그리고 뷰 카메라가 OSP의 현재 노드를 벗어나 다른 노드로 이동할 경우에 충돌대상 개체에 대한 체크를 통하여 제한적 OSP를 재구성한다. 제한적 OSP는 게임과 같은 환경에서 처리되는 화면을 분할단계로 제한하여 OSP의 분할 구조를 단순화 시킨다. 입 노드에 해당되는 자식 노드가 실재적인 처리 단위가 된다. 제한적 OSP는 (그림 3)과 같은 5단계로 OSP 구조로 분할되는 것을 볼 수 있다. 일반적인 OSP의 자료구조는 8개의 자식 노드을 갖는 트리 구조를 이용하여, 3차원 가상공간을 8개의 동일한 작은 입방체로 재귀 분할한다[1, 7, 10].



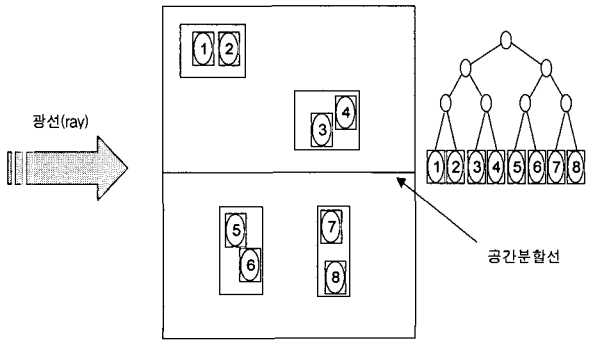
(그림 5) 제한적 OSP 노드의 탐색 원칙와 노드의 페이지 화면

하지만, 여기서 제시된 제한된 OSP는 시야 여부 결정보다는 3차원의 넓은 공간을 제한된 최종도달 단계까지 강제적으로 재귀분할 한 후 최종단계의 노드를 처리대상으로 삼는다. 따라서 제한적 OSP로 공간 분할을 한 후 계층적 경계 상자를 이용하여 충돌 검출을 처리한다.

3.2 계층적 경계상자를 이용한 충돌검출 처리속도의 개선

충돌 시스템 설계에서 고려하여야 하는 중요한 부분 중의 하나는 충돌검출에 사용되는 개체의 수가 많다는 것이다. MMORPG와 같은 유형의 게임은 거대한 지형에서 충돌에 관하여 처리하여야 하고 개체들을 충돌 그룹으로 충돌을 처리한다고 해도 많은 비용이 소요된다. 그러나 계층적 경계상자를 이용하여 처리를 단순화 시킨다면 충돌검출에 소요되는 비용을 감소시킬 것이다. 계층적 경계상자의 적용을 위해서는 제한적 OSP의 분할과정을 통하여 처리에 필요한 페이지가 결정된다. 처리에 필요한 영역을 대상으로 계층적 경계상자끼리의 충돌을 검출하기 위하여, 제한적 OSP에 의하여 선택된 노드를 계층적 경계상자로 구분하여 가까이 있는 개체들을 (그림 5)와 같은 형식으로 한데 묶은 계층적 경계상자로 만든다. 이 때 고려되는 부분으로는 첫째, 계층적 구조의 구축방향 둘째, 경계 볼륨의 구축 방향 셋째, 충돌정보의 처리방법 넷째, 게임과 같은 동적인 환경에서 충돌 예측에 관한 데이터의 유지방법 등이 고려되어야 한다 이 계층적 경계상자는 신 그래프의 기하학 데이터 셋을 통하여 얻을 수 있다. 신 그래프의 객체 정보는 객체가 이동하게 되면 수시로 삽입과 삭제에 의하여 자료 값이 변화하게 된다. 객체가 움직임이 발생할 때 마다 현시점의 충돌에 관한 판단을 하게 된다. OSP에 의하여 분할 처리된 노드의 공간은 BSP(Binary Space Partition)트리의 경우 공간분할 형태는 옥트리의 공간분할과는 차이를 가진다. BSP 트리는 분할을 위한 형태로 정확히 공간을 이등분하지 않는다. 이 방법은 이진트리의 자료를 구축하지만 단지 왼쪽과 오른쪽을 구분하기 위한 형태의 공간분할을 취한다. 계층적 경계상자의 경우는 공간을 정확히 이등분을 통하여 이진트리를 구축한다. 이렇게 분할된 공간을 재귀적으로 분할 시켜 나감으로써 이진트리를 구축하게 된다. 분할된 공간의 단위는 경계 쌍을 이루는 단위가 된다. 경계 쌍은 일정한 경계 기준 이루게 되므로 인접한 개체들끼리 묶어서 여러 개의 개체를 한데 묶은 경계상자들로 만들어 검사하는 것이 된다. 즉 경계상자를 단위 그룹으로 묶은 경계상자를 계층적으로 구성한다. 계층적인 경계상자를 이용하면 모든 경계상자들을 충돌검출 대상으로 삼지 않아도 되고, 결과적으로 시간 복잡도는 O(n*log*n)으로 떨어진다.

(그림 6)에서는 여덟 개의 개체들을 경계상자로 간주하고 묶은 뒤 다시 계층적으로 육면체 모양의 경계상자들을 적용한 예를 보여준다.



(그림 6) 경계상자의 계층적 구조

3.3 구 현

3D MMORPG 게임엔진은 거대한 3차원 도시의 수많은 빌딩과 개체를 실시간으로 빠르게 표현하기 위해서는 그래픽 렌드링의 빠른 처리가 필요하며 특히 충돌과 같은 처리는 게임엔진의 성능에 많은 영향을 미친다. 이에 관한 효율적인 처리 기법이 필요하다. 1단계로 제한적 OSP를 이용한 거대한 지형에 대한 공간의 분할이다. 제한적 OSP는 폴리곤의 개수를 기준으로 균등 분할 하지 않고 화면을 단순 분할 함으로써 처리 대상을 제한하기 위한 자료구조로 사용된다.

(그림 3)에서 거대한 지형 데이터를 세부 단계인 5단계로 탐색해 가는 과정을 볼 수 있다. 제 1단계는 주 처리 대상이 되는 넓은 외부 영역을 동적으로 빠르게 처리하기 위해서는 제한적 OSP를 이용한 공간분할이다. 최종 노드는 게임에서의 처리 단계의 페이지 화면이 된다.

제 2단계는 거대한 3차원 지형을 고정 볼륨 바운드를 이용하여 3차원 영역을 3차원 타일 맵을 구성한다. 이는 3차원의 개체의 위치와 값을 세부 적인 점으로 표현하지 않고 영역인 타일로 표현 함으로서 처리를 단순화 할 수 있다. 일반 적인 2차원 게임에서 적용하는 방식과 동일한 방법으로 3차원의 적용이 가능하다.

제 3단계는 신 그래프를 사용한 주어진 기하학 데이터 셋을 통하여 렌드링에 필요한 3차원 개체를 트리로 구축한다.

```
class Limit_octree {
        int iType ;
        int Depth ;
        FACE3D * flist ;
        VECTOR3D min ;
        VECTOR3D max ;
        VECTOR3D vCenter ;
} ;
 class octree_node : public Limit_octree {
        limit_octree * child[8] ;
        } ;
 class octree_leaf : public limit_octree {
        polygon_list polygons ;
        object_list objects ;
        ...
        } ;
```

(그림 7) 제한적 옥트리의 구축에 관한 코드

제 4단계는 계층적 경계상자를 구축한다. 3단계의 신 그래프의 정보를 이용하여 인접한 개체의 위치 정보를 얻을 수 있다. 이 정보를 이용하여 인접한 개체의 묶음을 2.2에서 제

시한 경계상자를 구할 수 있다. 이 경계상자의 각각을 계층적 경계상자를 적용하여 트리로 구축한다.

여기에 제시된 처리 방법에서 계층적 경계상자를 이용하는 것과 BSP트리를 사용하여 적용하는 방법의 차이점은 다음과 같다. BSP에 있어서는 분할을 위한 형태로 정확히 공간을 이등분하지 않는다. BSP 트리는 이진트리의 자료를 구축하지만 단지 왼쪽과 오른쪽을 구분하기 위한 형태의 공간분할을 취한다.

```
octree_node :: Limited_BuildOctree (FACE3D * flist) {
        LoadOctreeParameter () ;
        g_iMaxTriangle = NumOfVertsInFace (flist) /
        g_iOctreeParameter [1] ;
         CTREE * octree = AllocNewOctree () ;
         octree → flist = flist ;
         octree → min = MinimalOfVector (octree → flist) ;
         octree → max = MaximumOfVector (octree → flist) ;
         BeginFromRoot (octree) ;
         return octree ;
}
```

(그림 8) 제한적 OSP의Limited_BuildOctree()에 관한 코드

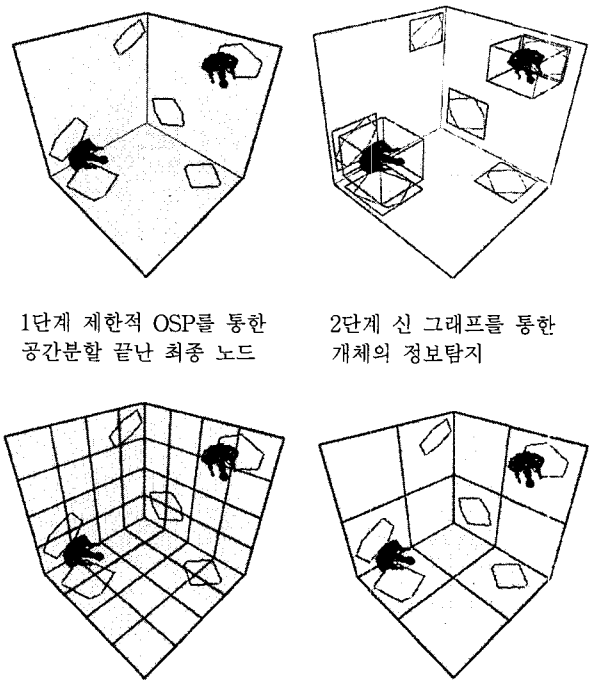
여기에 제시된 계층적 경계상자의 경우에는 공간을 정확히 이등분을 통하여 이진트리를 구축한다. 이렇게 분할된 공간을 재귀적으로 분할 시켜 나감으로써 이진트리를 구축한다. 분할된 공간의 단위는 경계 쌍을 이루는 단위가 된다. 분할된 공간 단위는 일정한 경계 기준을 이루게 되므로 인접한 개체들끼리 묶은 경계상자들도 만들어 검사를 하는 것이 된다. 계층적 경계상자를 이용함으로써 충돌검출에 필요한 개체를 검색하고 이를 이용함으로써 충돌검출 속도를 증가시킬 수 있다.

```
void octree_node :: hbb_ftb (const Point & point) {
 int near = (point dot node.plane_normal >= 0.0) ;
 if (child[near]) child[near] → hbb_ftb(point) ;
     for each polygon facing the near node in this plane {
        collision_detection() ;
           }
 if (child[near^1]) child[near^1] → planar_ftb (point) ;
        }
```

(그림 9) 계층적 바운드박스의 검출을 위한 의사 처리 코드

제한적 OSP처리의 장점은 폴리곤의 충돌검사를 모두 하지 않고 빠른 시간에 충돌검출을 판단할 수 있는 방법이 된다.

(그림 7)은 이러한 기본적인 처리 단계를 도시화한 것이다. 여기에 제시된 단계를 통하여 계층적 경계상자와의 충돌이 발생한 경우에만 최종 단계인 충돌검출 단계를 수행하게 된다. 실질적인 충돌검출 처리는 세 가지 단계로 처리작업이 수행된다. 첫째, 경계상자를 둘러 싸는 비교적 간단한 모양의 경계상자를 설정하여 광선과 경계상자간의 충돌검출을 행한다. 이렇게 해서 광선과의 충돌여부를 통하여 충돌 가능성을 체크한다.



1단계 제한적 OSP를 통한 공간분할 끝난 최종 노드

2단계 신 그래프를 통한 개체의 정보탐지

3단계 고정볼륨 바운드가 적용된 OSP

4단계 계층적 경계상자의 적용

(그림 7) 각 단계별 처리 화면

둘째, 광선과 경계상자간의 충돌이 발생한 경우에만 자세하게 충돌검사를 행한다. 이렇게 함으로써 비교적 멀리 떨어져있는 상자와의 충돌 체크는 피할 수 있다.

셋째, 경계상자간의 충돌이 발생한 경우에 한하여 정확한 충돌 체크를 위한 충돌검출 알고리즘을 이용하여 정확한 충돌여부를 검출한다.

처리 단계를 단계적으로 살펴보면 다음과 같다. 제 1단계는 공간을 분할하기 위하여 제한적 OSP를 사용하여 공간을 8개로 분할 및 재귀적으로 반복한다. 이렇게 분할된 공간의 자료 값을 이용하여 고정볼륨 바운드를 구축한 후 격자구조로 공간을 구축한다. 격자 구조로 공간을 구축하는 이유는 차 후 경계상자를 계층적으로 구성하기 위한 처리 값을 얻기 위 함이다. 그러나 공간분할은 광선이 이동됨에 따라 트리의 삽입과 삭제가 일어나게 되는데 경계를 벗어 날 때만 삽입과 삭제가 일어난다.

제 2단계는 광선과 경계상자간의 충돌을 판별하기 위한 충돌검출이 필요하다.

충돌검출은 다음과 같은 과정이 필요하다. 먼저 광선과 경계상자의 위치가 평행하거나 혹은 반대편에 존재한다면 충돌 할 가능성은 존재하지 않는다. 광선의 이동 경로에 따라 경계상자의 계층 노드를 탐색하며 충돌을 검출할 수 있다.

옥트리의 경계상자의 계층적 구조의 개념은 광선과 경계상자 간의 충돌이 검출을 위하여 8개의 부분으로 공간을 세분하고 각 상자에 객체들의 리스트를 저장할 것이다. 종료기준은 트리의 최단 노드 단계에 설정되어있다.

이에 대한 처리 효율성은 다음과 같다. 3차원 개체가 n개이고, 각각의 3차원 개체에 대해 면을 이루는 삼각형의 개수가 평균 m이라면, 시간 복잡도는 제한적 OSP를 활용한 계층적 경계상자 충돌검출의 시간은 o(n*log*n)이며, 경계상자간의 충돌검출을 하는 경우에는 o(n*log*n + α)으로 기존의 방식에 비하여 효율성이 뛰어나다. 3D MMORPG(Massive Multi-play Online Role Playing Game) 게임엔진 설계에 이러한 방식을 이용하여 가상공간을 표현 한다면 어느 정도의 속도 향상을 달성할 수 있다고 할 수 있다.

4. 결 론

3D MMORPG 게임엔진은 거대한 3차원 도시의 수많은 빌딩과 개체를 실시간으로 빠르게 효과적으로 표현하기 때문에 렌드링의 처리뿐만 아니라 속도에 영향을 미치는 많은 요소를 가지고 있다.

하드웨어의 급속한 발전에도 불구하고 게임상에 사용되는 거대한 지형 데이터와 3차원 개체를 실시간으로 처리하려면 아직까지 많은 어려움이 따른다. 본 논문은 게임엔진 설계에 필요한 세부 기술 중 엔진의 처리속도 향상을 위한 처리 방법을 제시하였다.

3D MMORPG의 주 처리 대상이 되는 넓은 외부 영역을 동적으로 빠르게 처리하기 위해서 공간분할을 사용하였다. 분할된 3차원 공간을 고정볼륨 바운드를 구축한 후 신 그래프를 통하여 주어진 기하학 데이터 셋을 이용하여 충돌검출에 필요한 3차원 개체를 트리로 구축하였다. 이를 처리값으로 이용한 계층적 경계상자를 이용함으로써 충돌검출에 필요한 개체를 검색하고 이를 통하여 충돌검출 속도를 증가시킬 수 있었다. 공간분할 방법으로 사용한 옥트리는 정적인 물체에 대한 정확한 표현시 주로 사용하나 본 논문에서는 이를 동적인 환경에 이용하기 위하여 제한적 옥트리 공간분할구조로 제한적 OSP를 사용했다.

충돌검출을 위하여 계층적 경계상자를 사용했고 이러한 처리의 장점은 게임엔진 처리의 효율성을 개선할 수 있다는 것이다. 왜냐하면 폴리곤의 충돌검사를 모두 하지 않고 빠른 시간에 충돌검출을 판단할 수 있는 방법을 제시하고 경계상자의 충돌검출에 대한 비용이 3차원 개체 개수에 대해 비례하여 증가하는 데에 대한 속도를 개선하기 때문이다.

본 논문에서 사용된 처리 형태는 첫 번째 단계로 넓은 외부 영역을 동적으로 빠르게 처리하기 위해서는 공간을 분할시켜 처리영역을 축소하였다. 두 번째 단계로 제한적 OSP로 3차원의 넓은 공간을 단순히 처리대상을 제한하기 위한 공간분할 구조로 사용하였다. 세 번째 단계로 3차원 공간을 고정볼륨 바운드를 구축한 후 신 그래프를 통하여 주어진 기하학 데이터 셋을 이용하여 처리에 필요한 3차원 개체를 트리로 구축한다. 이를 처리 값으로 이용한 계층적 경계상자를 이용하여 충돌검출을 한다.삼각형의 충돌검출을 모두 하지 않고서도 빠른 시간에 충돌검출을 할 수 있으며, 경계상자들간의 충돌검출시 드는 비용이 3차원 개체의 개수에 대하여 비례하여 증가하는 데에 대한 효율적인 처리를 수행하였다.

시간 복잡도를 살펴보면 다음과 같다. 3차원 개체가 n개이고, 각각의 3차원개체에 대해 면을 이루는 삼각형의 개수가 평균 m이라면, 경계상자를 이용하지 않을 경우 충돌검출에 필요한 시간복잡도는 $O(n^2m^2)$가 되지만, 경계상자를 이용하는 경우 $O(n^2 + \alpha)$가 된다. 여기서 α는 경계상자들간의 충돌하는 경우 다시 자세하게 삼각형을 중심으로 충돌검출을 행하는 시간을 말한다. 반면 제한적 OSP를 활용한 계층적 경계상자 충돌검출의 시간은 o(n*log*n)되며, 경계상자간의 충돌검출을 하는 경우에는 o(n*log*n+ α)으로 기존의 방식에 비하여 효율성이 뛰어나다. 3D MMORPG(Massive Multi-play Online Role Playing Game) 게임엔진 설계에 이용한다면 속도 향상을 달성할 수 있다고 할 수 있다. 3차원 게임 엔진설계에서의 충돌검출은 아주 중요한 부분으로 다루어지고 있으며, 이 방법은 충돌검출뿐만 아니라 렌드링 처리에도 효과적으로 사용할 수 있을 것이다.

참 고 문 헌

[1] 이승욱, 박경환, "CLOD를 활용한 충돌검출과 옥트리 분할기법", 한국정보처리학회 가을학술발표논문집, 제8권 제2호, pp.615-618, 2001.

[2] Bobic, Nick, "Advanced Collision Detection Techniques," http://www.gamasutra.com/features/20000330/bobic_pfv.htm, March, 2000.

[3] Eberly, David H., 3D Game Engine Design, Academic Press, pp.189-265, 2001, Blow, Jonathan, "Practical Collision Detection," Proceedings, Computer Game Developers Conference, 1997, http://142.104.104.232/eCOW/projects/Resources/practical_collision_detection.html.

[4] H. J. Haverkort, M. de Berg and J. Gudmundsson, "Box-Trees for Collision Checking in Industrial Installations," Proc. 18th ACM Symp. on Computational Geometry, pp. 53-62, 2002.

[5] J. Cohen, M. Lin, D. Manocha, K. Ponamgi, "I-COLLIDE : An Interactive and Exact Collision Detection System for Large-Scaled Environments," Proceedings of the 1995 ACM International 3D Graphics Conference, pp.189-196, 1995.

[6] Kyu-Young Whang, Ju-Won Song, Ji-Woong Chang, Ji-Yun Kim, Wan-Sup Cho, Chong-Mok Park, Il-Yeol Song,

"Octree-R : An Adaptive Octree for Efficient Ray Tracing," IEEE Trans. On Visualization and Computer Graphics, 1995.

[7] M. de Berg and K. Dobrindt, On levels of detail in terrains. Graphical Models and Image Detection 60, pp.1-12, 1998, [technical report].

[8] M. Kelleghan, "Octree Partitioning Techniques," Game Developer magazine, pp.30-33, 1997, http://www.gamasutra.com/features/programming/080197/octree.htm.

[9] M. de Berg, M. van Kreveld, M. Overmars and O. Schwarzkopf, Computational Geometry : Algorithms and Applications. Springer-Verlag, Heidelberg, 1997.

[10] M. de Berg, M. de Groot and M. Overmars, New results on binary space partitions in the plane, Computational Geometry : Theory and Applications 8, pp.317-333, 1997, [draft].

[11] M. de Berg. Ray Shooting, Depth Orders and Hidden Surface Removal, Lecture Notes in Computer Science 703, Springer-Verlag, Berlin, 1993.

[12] P. Agarwal, M. de Berg, J. Matousek and O. Schwarz- kopf. Constructing levels in arrangements and higher order Voronoi diagrams. SIAM J. Comput. 27, pp.654-667, 1998, [technical report].

[13] S. Gottschalk, M. C. Lin, D. Manocha, "A Hierarchical Structure for Rapid Interference Detection," Proc Siggraph 96 ACM Press New York, pp.171-180, 1996.

이 승 욱

e-mail : sunlee@donga.ac.kr

1989년 동아대학교 경제학과(학사)

1991년 동아대학교 산업대학원 컴퓨터공학(공학석사)

2001년 동아대학교 컴퓨터공학과(박사과정 수료)

1992년~1997년 보스컴퓨터학원원장, 시스템 사업부 이사

1996년~2000년 부산여자대학 겸임교수

2000년~현재 동아대학교 전기전자컴퓨터 공학부 BK21 계약교수

관심분야 : 멀티미디어,데이터베이스, 3D 그래픽, 게임 등