

넓은 가상공간에서의 충돌검출

이승욱 ⁰ 박경환

동아대학교 컴퓨터공학과

sunlee@mail.donga.ac.kr ⁰ khpark@daunet.donga.ac.kr

Collision Detection for Large-Scaled Environments

Sung-ug Lee ⁰ Kyung-hwan Park

Dept. of Computer Engineering, Dong-A University

요약

본 논문에서는 실시간으로 상호작용이 가능한 광범위한 3차원의 넓은 외부영역을 렌더링하기 위해 공간분할 방법으로 제한적 옥트리를 이용하고, 충돌검출 방법으로 계층적 경계상자를 사용하여 효과적으로 충돌검출을 하는 방법을 소개한다. 넓은 외부 영역을 동적으로 빠르게 처리하기 위해 공간을 분할시켜 처리영역을 축소하고, 3차원 공간을 고정볼륨 바운드를 구축한 후 이를 처리 값으로 이용한 계층적 경계상자를 이용하여 충돌처리를 한다. 옥트리는 정적인 물체에 대한 정확한 표현 시 주로 사용되나 본 논문에서는 이를 동적인 환경에 이용하기 위하여 제한적 OSF를 사용한다. 또한 폴리곤의 충돌검사를 모두 하지 않고 빠른 시간에 충돌검출을 판단할 수 있는 방법과 경계상자의 충돌검출에 대한 비용이 3차원 개체 개수에 대해 비례하여 증가하는 데에 대한 개선 방법을 제안 한다.

1. 서론

게임은 크게 게임엔진과 게임 컨텐츠로 구성된다. 게임 엔진은 특정 게임 컨텐츠 포맷을 처리하는 컴포넌트의 집합으로 대개 게임의 장르에 맞는 게임 컨텐츠의 틀을 가지고 있다. 따라서 게임 엔진은 레벨포맷, 오디오처리, 이벤트 핸들러, 입력처리 등의 핵심 로직을 모두 포함한다. 다음 그림 1은 게임엔진의 중요한 세부 요소를 도시하고 있다.

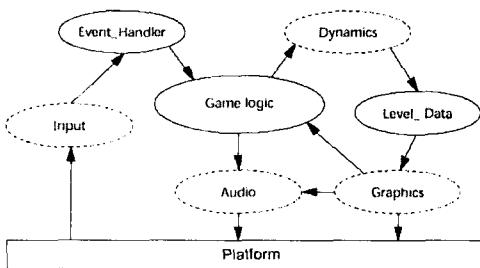


그림 1. 게임의 구조

최근 하드웨어의 급속한 발전으로 인하여 3차원 가속기가 내장된 그래픽 카드를 이용한 그래픽엔진이 장면관리를 위하여 도입되었지만, 3차원 공간 상에 사용되는 거대한 지형 데이터와 개체를 실시간으로 처리하기 위해서는 많은 어려움이 따른다. 이러한 3차원 지형의 실시간 처리에 있어서 고려해야 사항으로 다음과 같은 것이 있다.

3차원 게임에 사용되는 모델의 폴리곤 수를 최대한 줄여야 한다. 본 논문은 실시간으로 상호작용이 가능한 광범위한 3차원의 넓은 외부영역을 처리하기 위한 공간분할 방법으로 제한적 옥트리를 이용하고, 충돌검출 방법으로 계층적 경계상자를 이용한다. 넓은 외부 영역을 동적으로 빠르게 처리하기 위해서는 공간을 분할시켜 처리영역을 축소하고, 3차원 공간을 고정볼륨 바운드를 구축한 후 이를 처리 값으로 이용한 계층적 경계상자를 이용하여 충돌검출을 한다. 삼각형의 충돌검출을 하지 않고서도 빠른 시간에 충돌검출을 판단할 수 있으며, 경계상자들끼리의 충돌 시 드는 비용이 3차원 개체의 개수에 대하여 비례하여 증가하는 데에 대한 개선 방법을 제시하여, 게임엔진 등과 같은 시스템 설계 시 유용하게 사용할 수 있는 처리기법을 제안 한다.^{[1][2][3][6]}

2. 충돌검출기법

2.1 제한적 옥트리의 공간분할 방법

OSF(Octree Space Partitioning)의 알고리즘은 3차원의 정적인 공간을 정확히 표현을 위해 사용되는 자료구조이다. 그러나 본 논문에서는 계층적 제한 경계상자(Hierarchical Bounding Volumes)을 이용하여 공간을 모두 옥트리로 분할하지 않고 일정한 부분을 제한적 OSF를 이용하여 분할한 후 계층적 경계상자(Hierarchical grid Bounding Box)을 이용하여 충돌검출을 처리한다. 거대지형을 제한적 OSF를 이용하여 구축할 경우 동적인 처리를 하기 위하여 3차원개체의 움직임에 따라 빈번하게 발생되는 트리의 재구성 회수를 줄일 수 있다. 트리의 재구성은 충돌검출 광선이 계층적 경계상자를 벗어나다

른 노드로 이동 할 경우에 충돌대상 개체에 대한 체크를 통하여 제한적 OSF를 재구성한다. 제한적 OSF는 게임과 같은 환경에서 MMORPG와 같은 게임환경에서 처리되는 화면을 최종 분할단계로 제한하여 육트리의 분할 구조를 단순화 시킨다. 이렇게 제한적 OSF는 그림2와 같은 5단계로 육트리 구조는 분할되는 것을 볼 수 있다. 일반적인 육트리의 자료구조는 8개의 자식 노드를 갖는 트리 구조를 이용하여, 3차원 공간을 8개의 동일한 작은 입방체로 재귀 분할한다. 이렇게 분할된 육트리를 정의 각도 구해서 화면 시야 여부를 결정하고, 8개의 점이 모두 다 시야 안이라면 그 안의 육트리를 더 이상의 클리핑도 필요없이 렌더링에 필요한 노드로 처리되어 검출한다. 8개의 점이 모두 밖에 있다면 그 안의 육트리는 화면에 보이지 않는 것이 확실하므로 노드가 검출되지 않고 처리에서 제외시켜버린다. 앞의 2가지의 경우가 아니라면 다시 하위 육트리로 내려가 다시 앞의 2가지 분할이 끝나면 카메라의 viewing frustum과 육트리는 모든 노드와 포함 연산을 통해 보여지는 노드를 검출한다.

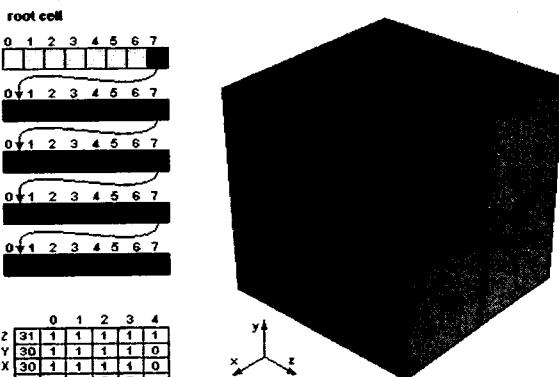


그림 2 육트리 노드의 탐색 원칙

하지만, 여기서 제시된 제한된 OSF는 시야 여부 결정보다는 3차원의 넓은 공간을 제한된 최종단계 단계까지 강제적으로 재구분할 한 후 최종단계의 노드를 처리대상으로 삼는다. 기존의 육트리를 이용하여 충돌검출을 하지 않고 계층적 경계상을 이용하여 충돌검출을 하기 위한 제한된 육트리를 사용한다. [7][8][9][10]

2.2 3차원 개체의 충돌검출

제한적 육트리의 검출과정을 통하여 처리에 필요한 카메라의 영역이 결정된다. 처리에 필요한 영역을 대상으로 여기에서 제시된 경계상자끼리의 충돌을 검출하기 위하여, 제한적 육트리에 의하여 선택된 노드를 경계상자로 구분하여 가까이 있는 개체들을 그림 3과 같은 형식으로 한데 묶은 경계상자로 만든다. 이 경계상자를 통하여 수시로 객체의 정보가 삽입과 삭제를 통하여 현시점의 충돌에 관한 판단을 하게 된다. [7][9]

2.3 경계상자를 이용한 충돌검출 방법

본 논문에서는 경계상자끼리의 충돌 시 드는 시간이 3차원 개체의 개수에 대하여 비례하여 증가하는 데에

대한 개선이다. 경계상자를 각각의 하나씩의 개체들에 대하여 검사하는 것이 아니라, 일정한 경계 기준으로 묶어 있는 것들끼리 차례로 둘씩 묶어서 여러 개의 개체를 한데 묶은 경계 상자들도 만들어 검사하는 것이다. 즉 경계 상자를 단위 그룹으로 묶은 경계 상자를 계층적으로 구성한다. 계층적인 경계상자를 이용하면 모든 경계상자들을 충돌 검사 대상으로 삼지 않아도 되고, 결과적으로 시간 복잡도는 $O(n \log n)$ 으로 떨어진다.

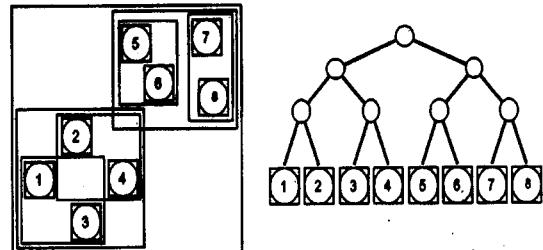


그림 3. 경계상자의 계층적 구조

그림 3에서는 여덟 개의 개체들을 일단 공모양의 경계상자로 간주하고 묶은 뒤, 다시 계층적으로 육면체 모양의 경계상자들을 적용한 예를 보여준다. [3][5][9]

2.4 충돌검출을 위한 처리방법

객체들의 정확한 충돌검출을 위해서는 광선(ray) 검출에 쓰이는 방법을 사용하여 처리할 수 있다. 광선과 벡터를 이용해 표현된다. 벡터의 시작점과 방향을 표시한다.

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

P: 광선 위의 임의의 한 점(point)

S: 광선의 시작점(start point)

t: 변수, 광선 위에서의 상대적 위치를 표시한다. ($0 \leq t < \text{무한대}$)

V: 방향벡터 $V(x,y,z)$

($t > 0$) 식으로 나타내면 ray는 위와 같이 나타낼 수 있다. t가 0이면 ray의 시작점이고, t에 따라서 대응되는 ray 위의 점을 얻을 수 있다. P, S, V는 모두 3차원 벡터이다. 이것들을 이용해서 관선과 평면과의 교차점, 광선과 개체와의 교차점을 계산할 수 있다. 움직이는 구와 평면의 교차점 찾기 위해서는 평면은 다음과 같이 나타낼 수 있다.

Xn 내적 $X = X$: 벡터, 평면 위의 한 점을 나타낸다.

Xn : 평면의 법선 벡터 d : 좌표원점에서부터 평면까지의 최단 거리 $Ax + By + Cz + D = 0$

이제 광선이 평면 위의 임의의 점을 뚫고 지나간다고 했을 때, (1),(2)식을 이용하여, 만나는 점에서 다음과 같은 식이 성립하는 것을 알 수 있다.

$$Xn \text{ 내적 } P = d \quad \dots \dots \dots (2)$$

광선이 평면 위의 임의의 점을 뚫고 지나간다고 했을 때,

(1),(2)식을 이용하여, 만나는 점에서 다음과 같은 식이 성립하는 것을 알 수 있다.

$$Xn \text{ 내적 } P = d, P\text{대신에 } (1)\text{식을 대입하면},$$

대한 정확한 표현 시 주로 사용되나 본 논문에서는 이를

X_n 내적 $(S + t \cdot V) = d$, $(X_n$ 내적 $S) + t \cdot (X_n$ 내적 $V) = d$
 t 에 대해서 전개하면, $t = (d - X_n$ 내적 $S) / (X_n$ 내적 $V)$
 d 를 치환하면 $\rightarrow t = (X_n$ 내적 $P - X_n$ 내적 $S) / (X_n$ 내적 $V)$
 다시 기술하면, $t = (X_n$ 내적 $(P - S)) / (X_n$ 내적 $V)$

여기서 t 는 광선의 시작점 S 부터 광선의 방향을 따라서 평면과 교차하는 점까지의 거리를 나타나게 된다. 여기서 t 는 시작점과 교점간의 거리를 $= 0$ 이 되는 경우가 있을 수도 있는데 그러면 광선과 평면이 평행인 경우이다. 또 t 가 0보다 작은 수를 나타낸다. 그러므로 ray를 나타내는 식에 이 t 를 대입하면 교점을 구할 수 있다. X_n 내적 $X = X$ 가 나오는 경우는 광선의 뒤쪽에서 평면과 만나는 것을 의미하게 되므로 결국 충돌이 발생하지 않는다. 함수의 인자를 통하여 평면과 관선의 시작점과 방향 벡터간의 충돌이 일어난 곳까지의 거리를 저장할 인자 (lambda), 충돌지점에서의 법선 값을 저장할 인자 가 있다는 것을 알 수 있다. 이를 통하여 실제적인 충돌을 검출한다. [4][6][8][9][10]

3. 결론

본 논문은 3D 게임엔진에 실시간으로 상호작용이 가능하고 3D MMORPG(Massive Multiplay Online Role Playing Game) 게임에 적합한 가상공간을 표현하기 위해 필요한 기술을 분석한다. 처리의 첫 번째 단계로 넓은 외부 영역을 동적으로 빠르게 처리하기 위해서는 공간을 분할시켜 처리영역을 축소해야 한다. 제한적 OSF는 시야여부 결정보다는 3차원의 넓은 공간을 제한된 최종도달 단계까지 강제적으로 재귀분할 한 후 최종단계의 노드를 처리대상으로 삼는다. 다음 단계로 3차원 공간을 고정볼륨 바운드를 구축한 후 이를 처리 값으로 이용한 계층적 경계상자를 이용하여 충돌처리를 한다. 삼각형의 충돌검출을 모두 하지 않고서도 빠른 시간에 충돌검출을 할 수 있으며, 경계상자들끼리의 충돌 시드는 비용이 3차원 개체의 개수에 대하여 비례하여 증가하는 데에 대한 개선 방법을 제시하였다. 시간 복잡도를 살펴보면 다음과 같다. 3차원 개체가 n 개이고, 각각의 3차원개체에 대해 면을 이루는 삼각형의 개수가 평균 m 이라면, 경계상자를 이용하지 않을 경우 충돌검출에 필요한 시간복잡도는 $O(n^2m^2)$ 가 되지만, 경계상자를 이용하는 경우 $O(n^2+\alpha)$ 가 된다. 여기서 α 는 경계상자들끼리 충돌하는 경우 다시 자세하게 삼각형을 중심으로 충돌검출을 행하는 시간을 말한다. 그리고 계층적 경계상자를 이용하면 모든 경계상자들을 충돌검출 대상으로 삼지 않아도 되고, 결과적으로 시간 복잡도는 $O(n/\log n)$ 으로 떨어진다. 실시간으로 상호작용이 가능한 광범위한 3차원의 넓은 외부영역을 렌더링하기 위해 공간분할 방법으로 제한적 옥트리를 이용하고, 충돌검출 방법으로 계층적 경계상자를 사용하여 효과적으로 충돌검출을 하는 방법을 소개하였다. 넓은 외부 영역을 동적으로 빠르게 처리하기 위해 공간을 분할시켜 처리영역을 축소하고, 3차원 공간을 고정볼륨 바운드를 구축한 후 이를 처리 값으로 이용한 계층적 경계상자를 이용하여 충돌검출을 한다. 옥트리는 정적인 물체에

동적인 환경에 이용하기 위하여 제한적 OSF를 사용한다. 또한 풀리곤의 충돌검출을 모두 하지 않고 빠른 시간에 충돌검출을 할 수 있는 방법과 경계상자의 충돌검출에 대한 충돌검출 시드는 비용이 3차원 개체 개수에 대해 비례하여 증가하는 데에 대한 개선 방법을 제안 했다. 여기에 제시된 이론은 계층상자 간의 쌍으로 충돌검출등과 같은 경우에는 비효율적이다. 본 논문에서 제시된 방법은 3D MMORPG 등과 같은 게임 엔진 설계등과 같은 시스템 개발에 이용한다면 효율적인 처리능력을 발휘할 수 있다고 본다.

[참고문헌]

1. J. Rohlfs and James Helman, " IRIS Performer: A High-Performance Multiprocessing Toolkit for Real-Time 3D Graphics ", Proc, Siggraph 94, ACM Press, New York, pp.381-394, 1994
2. M. Shantz, Building Online Virtual Worlds, Proc. Graphicon 96, Grafo Computer Graphics Society, St. Petersburg, Russia, Vol. 2, pp. 12-17, 1996.7
3. Gottschalk, M.C. Lin, D. Manocha, " A Hierarchical Structure for Rapid Interference Detection ", Proc Siggraph 96 ACM Press New York, pp 171-180, 1996
4. Bishop, L.; Eberly, D.; Whitted, T.; Finch, M.; Shantz, M. Computer Graphics and Applications, IEEE, Volume 18, pp. 46-539. 1998
5. Young J. Kim, Ming C. Lin, and Dinesh Manocha Workshop on Algorithmic Foundations of Robotics (WAFR), 2002.12
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, 1995.12
7. J. CohenM, LinD. 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
8. J. Cohen, M. Lin, D. Manocha, K. Ponamgi, Proceedings of the 1995 ACM International 3D Graphics Conference, pp. 189-196, 1995
9. Mike Kelleghan. Octree Partitioning Techniques. Game Developer magazine, pp.30-33. 1997. 7 <http://www.gamasutra.com/features/programming/080197/octree.htm>
10. H.J. Haverkort, M. de Berg, and J. Gudmundsson. Box-Trees for Collision Checking in Industrial Installations. Proc. 18th ACM Symp. on Computational Geometry, pages 53-62, 2002.