제 18 회 한국정보처리학회 추계학술발표대회 논문집 제 9 권 제 2 호 (2002.11)

샘플링 트리를 이용한 효과적인 충돌탐지

여은선*, 한정현*
*성균관대학교 정보통신공학부
e-mail : ishia0518@hotmail.com

Efficient Collision Detection using Sampling Tree

EunSun Yea*, JungHyun Han*
*School of Information & Communications Engineering, SungKyunKwan University

요 약

3 차원 게임, 컴퓨터 시뮬레이션 등과 같은 대화적 가상환경의 사실성을 높이기 위해 충돌탐지는 절대 필요한 요소가 된다. 하지만 기존의 바운딩 볼륨 기반 기법들은 볼륨 계산의 복잡성 때문에 실시간 어플리케이션에서 정교한 검사를 수행하는데 어려움이 있었다. 본 논문에서는 바운딩 구와 꼭지점 샘플링을 이용한 정교하고도 효과적인 충돌탐지 방법을 제안한다. 한편, 제안된 기법에서는, 반드시 필요한 부분에서만 꼭지점을 이용한 충돌검사를 시행해서 충돌검사의 효율성을 높였고, 이를 위한 샘플링 툴을 제공하였다.

1. 서론

컴퓨터 그래픽스 응용 분야에서 많이 쓰이는 충돌 연산을 위한 단계는 물체의 충돌을 탐지하고 충돌부위를 찾으며 충돌 후의 반응까지 처리하는 일련의 작업으로 이루어진다. 이 중 충돌 탐지(collision detection)는 상황의 특성에 관계없이 공통적으로 적용될 수 있는 부분으로 가상세계의 물체들을 보다 사실적으로 표현하기 위해 활발히 연구되고 있다.

충돌 탐지는 가상환경의 대표적인 예인 게임, 시뮬레이션, 로봇공학의 충돌 회피 길 찾기 알고리즘 등 다양한 분야에서 사용되고 있다.

현재 충돌탐지 방법에서 보편적인 것은 바운딩 볼륨의 계층적인 구조를 사용하는 것이다. 하지만 최근 주목 받고 있는 물체의 형태에 보다 가까운 바운딩 볼륨은 그 볼륨 자체의 충돌탐지 계산이 복잡한 것이 단점이다.

본 논문에서는 샘플링 트리를 이용해 보다 정확하면서도 빠른 충돌탐지 방법을 제안하고자 한다. 샘플링 트리는 노드 범위의 바운딩 구와 리프노드에 샘플링 된 꼭지점의 인덱스를 가지고 있는 트리를 말하며, 각 노드의 바운딩 구와 꼭지점 좌표를 이용해서 표면을 샘플링한 구를 이용하는 방식을 채택하고 있다. 바운딩 구를 이용한 충돌탐지는 단순하면서도 정교한 연산을 수행할 수 있는 장점이 있다. 이는 게임 같은 실시간 어플리케이션에서 정확하면서도 빠른 검사를 수행하는데 적합할 것이다.

본 논문은 2 절에서 주요 관련연구에 대해 기술하고, 3 절에서 본 논문에서 사용된 기법들을 설명한다. 그리고 4 절에서 구현 결과를 보여준 후, 5 절에서 결론을 내리고 향후 과제를 제시하며 마친다.

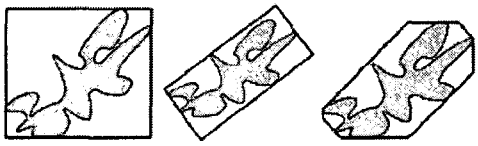
2. 기존연구

2-1. 기존연구의 경향

가상환경에 대한 관심이 높아지면서, 충돌탐지 또한 사실감을 높일 목적으로 계속 연구되어 왔다. 또한 가상환경을 구성하는 물체의 수가 많아지고 복잡해짐에 따라 여러 개의 바운딩 볼륨으로 물체를 근사화하여 계층 구조로 검사하는 방법이 보편화 되었다.

기존 논문들의 대부분은 좀더 정확하고 효과적인 충돌탐지를 수행하기 위한 바운딩 볼륨의 변형이나, 바운딩 볼륨 계층구조를 구성하는 방법, 또는 바운딩 볼륨 검사방법을 바꾸어 좀 더 간단한 계산 방법을 만들어 내는 것이었다. 바운딩 볼륨의 종류에는 물체를 둘러싸는 가장 긴 축이 반지름이 되는 바운딩 구[1,8], 세계 좌표계와 평행한 육면체인 AABB(Axis Aligned Bounding Box)[4], 물체의 축에 나란한 육면체인 OBB(Object Oriented Box) [2,6], k 개의 불연속면으로 표현되는 K-DOP(K-Discrete Orientation Prototype) [3]등이 있다. 전체적인 충돌탐지 계산 복잡도로 볼 때, 바운

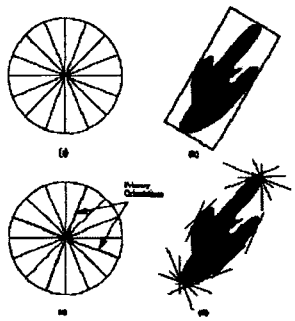
딩 볼륨 자체의 계산복잡도가 더 큰 볼륨, 즉 바운딩 볼륨에 보다 빈 공간이 없는 볼륨이 계산복잡도가 더 낮으며 대표적인 바운딩 볼륨으로는 OBB 와 K-DOP 가 있다.



(그림 1) AABB 와 OBB 와 K-DOP 의 비교

(그림 1)에서는 좌측에서부터 AABB 와 OBB, K-DOP 의 볼륨에 꽉 차는 정도를 보여준다.

또한 QuoSPO[5]는 이러한 두 볼륨의 장점인 물체에 나란한 축을 가지는 것과 정해진 방향으로 나뉘는 것을 결합한 형태라고 할 수 있다. OBB 에서 Primary axis 를 얻고 공간을 정해진 방향으로 분할하여 바운딩 볼륨을 얻는다.



(그림 2) QuOSPO Tree 의 구성

(그림 2)는 이와 같은 QuoSPO Tree 를 만드는 과정을 보여준다.

하지만 변형 가능한 물체를 대상으로 한 충돌탐지에서는 실시간으로 복잡한 바운딩 볼륨을 업데이트 하기 힘들기 때문에 AABB 를 사용하기도 한다[4]. 또한 SWIFT[7] 같은 알고리즘에서는 보로노이 영역의 개념을 이용해 가까이 있는 대상을 중심으로 충돌 탐지를 수행하며, 대상물체를 간략화한 볼륨들을 가지고 간략화된 모델의 꼭지점 사이에 관련성을 두어 모델의 충돌탐지의 단계를 달리했다.

2-2. 비용 함수 (Cost Function)

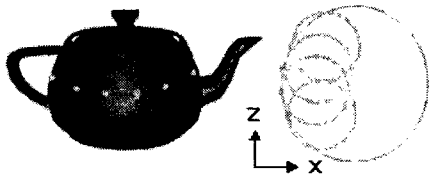
$$t = nv \times cv + np \times cp + nu \times cu$$

비용 함수란, 충돌탐지의 복잡도를 정하는 기준이 되는 식이다. nv 는 계산하는 바운딩 볼륨의 개수, np 는 계산하는 삼각형 수, nu 는 바운딩 볼륨의 갱신 횟수에 해당된다. c 에 해당하는 항목들은 각각에 드는 비용을 말한다. 일반적으로 바운딩 볼륨 자체의 계산 비용이 싼 볼륨들은 바운딩 볼륨 검사 횟수나, 복잡한 삼각형 단위의 검사가 늘어나게 되므로 오히려 계산 복잡도가 더 높다고 할 수 있다.

3. 샘플링 방법

3-1. 구 샘플링

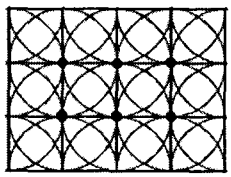
꼭지점을 이용한 구 샘플링이란, 가상환경에서 사용되는 물체의 정보를 나타내는 꼭지점을 이용해 가장 계산이 간단한 구로 표면을 근사화 하는 작업이다. 이는 보다 근사화 된 형태로 충돌검사를 수행하면서, 바운딩 볼륨의 충돌탐지나 업데이트 계산을 간략화 하고자 하는 것이다.



(그림 3) 구 샘플링(XZ)

(그림 3)은 XZ 좌표계에서 본 구 샘플링의 한 예를 보여주고 있다

이것을 XY 좌표계에서 보면 각 꼭지점을 중심으로 한 구가 중복되므로, (그림 4)와 같은 모양이 된다.



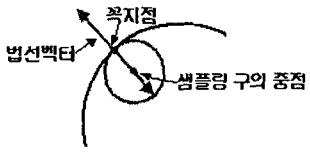
(그림 4) 구 샘플링 (XY)

구 샘플링은 꼭지점을 기준으로 해서 중심점이 해당 꼭지점의 법선 벡터의 반대방향에 존재 하게 하기 때문에 구의 중복되는 부분들로 충분히 표면을 근사할 수 있다.

구 샘플링을 이용하여 얻는 이득을 계산 복잡도 식에서 알아보자.

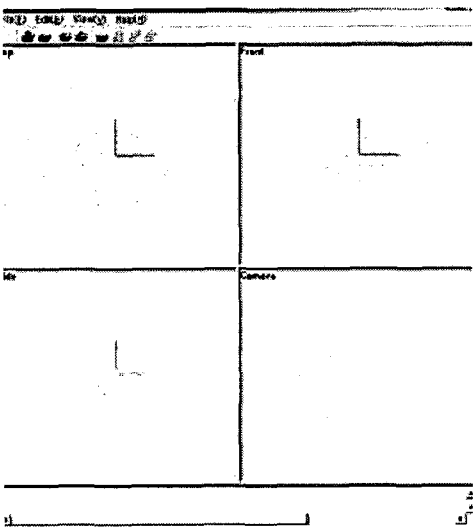
$$t = nv \times cv + np \times cp + nu \times cu$$

바운딩 볼륨의 계산비용(cv)은 구를 이용한 것으로 반지름의 합과 중심의 거리 비교이므로 감소하고, 삼각형 단위(np, cp)의 검사는 없으므로 감소하게 된다. 또한 바운딩 볼륨의 업데이트(nu, cu)는 각 바운딩 구의 중심점의 이동이고 샘플링 구의 경우, 꼭지점 검사가 필요할 때 (그림 5)와 같이 중심점을 꼭지점의 법선 벡터 반대방향으로 반지름 길이만큼 이동 해주면 되므로, 감소한 값을 가지게 된다.



(그림 5) 샘플링 구의 중점 업데이트

3-2. 꼭지점 샘플링



(그림 6) 샘플링 툴

앞서 언급한 구 샘플링 방법을 사용하려면 꼭지점을 샘플링 해 두어야 한다. (그림 6)는 샘플링 툴을 보여주고 있다. API 로는 OpenGL 이 쓰였고, view 는 top, front, side, camera 로 구성되어 있다. 꼭지점을 샘플링하는 인터페이스로는 마우스의 x, y 좌표를 이용한다. 마우스로 화면을 클릭하여 드래그 하면 화면에 사각형이 나타나는 데, 이 구간의 꼭지점을 선택하는 것이다. 좌표값들은 하나의 좌표계를 기준으로 비교해야 하므로, 세계좌표를 기준으로 검사하였으며, x, y 각 좌표는 퀵소트로 정렬 되어있어 범위를 찾기 쉽다.

3-3. 샘플링 트리 구성

샘플링 트리를 구성하기 위해 샘플링 툴을 사용한다. 먼저 전체 볼륨을 포함하는 가장 큰 상위노드를 만든다. 다음 번의 영역을 선택했을 때 왼쪽 노드에는 부모 노드의 영역과 선택한 영역의 교집합에 해당하는 꼭지점들이 선택된다. 그리고 같은 트리 레벨의 오른쪽 노드에는 부모 노드 중에서 왼쪽 노드에 포함되지 않은 부분을 넣는다.

샘플링 된 트리는 충돌탐지에 사용하기 위해 *. spt 파일로 저장한다.

샘플링을 할 때, 충돌이 자주 일어날 부분은 예측 가능하다. 특히 게임 같은 어플리케이션에서 캐릭터에는 많은 동작들이 존재한다. 이런 경우, 그 동작의 특성에 따라 먼저 충돌 가능한 부분이 존재한다.

예를 들어 사람 캐릭터에는 걷는 동작이 가장 많고, 걷고 있는 경우라면, 발 부분이 충돌 가능성이 가장 크다. 따라서 이런 충돌 가능성이 높은 범위를 트리의 상위 노드에 존재하도록 샘플링 순서를 정해서 자주 충돌하는 부분의 빠른 탐지를 할 수 있다. 또한 어플리케이션에서 동작하는 범위, 이동방향, 다른 물체의 동작 등을 고려해서 같은 오브젝트라도 다른 트리로 관리할 수도 있다.

3-4. 샘플링 트리를 이용한 충돌탐지

3-4-1. 트리를 이루는 노드의 구성

```
class CNode
{
public:
        int id;
        CVertex center;
        float bv_radius;
        float ss_radius;
        bool active;
        int num_uni_idx;
        int *Ver_uni_Idx;
        int GetLChild();
        int GetRChild();
        int GetParent();
};
```

위 코드는 트리의 노드를 구성하는 클래스이다. 트리는 평균적으로 빠른 순회 속도를 가진 이진 트리를 사용한다. 트리의 저장 형태는 속도 향상을 위해 하나의 배열 형태로 저장되고 아이디를 이용한 쉬프트 연산에 의해 자식 노드나 부모 노드를 찾을 수 있는 맴버 함수를 가지고 있다.

CNode 클래스는 리스트에서의 식별 아이디, 노드의 바운딩 구 정보, 또한 acitive 라는 실제 샘플링 구의 검사가 필요한 경우에는 이에 필요할 정보를 가진다.

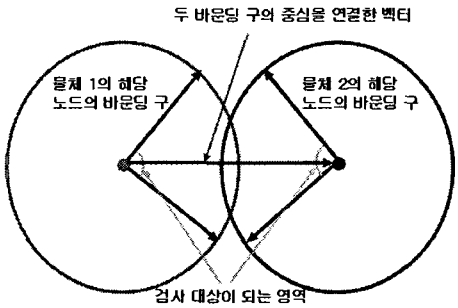
꼭지점을 이용한 테스트는 아무리 가벼운 비용의 검사라도 꼭지점의 수에 종속적이기 때문에, 검사할 꼭지점 대상을 제거할 방법이 필요하다. 따라서 해당 노드 구간의 x, y, z 중 가장 긴 축을 기준으로 한 바운딩 구를 구성해서 상위 노드에서는 대략적인 충돌 대상을 선정하는 것이다. 실제적으로 active 노드가 아닌 노드는 꼭지점 인덱스의 리스트도 갖지 않는다.

3-4-2. 충돌한 샘플링 구의 쌍 찾기

충돌 탐지를 위해 트리를 순회할 때는 충돌하는 모든 쌍을 찾아내기 위해서 왼쪽 자식 노드와 오른쪽 자식 노드 모두를 순회하게 된다. 먼저 바운딩 구로 충돌 가능성을 검사해서 상위 노드에서 충돌 가능성

이 없는 것으로 판정되면 하위 노드 모두 검사 대상에서 제외된다. 검사하는 노드가 액티브 노드이면 바운딩 구의 검사 후에 샘플링 구를 쌍을 이루어 검사를 수행한다.

샘플링 구를 검사하는 경우, (그림 7)과 같이 그 검사 대상을 전부로 두는 것이 아니라 법선 벡터의 범위에 따른 검사를 수행하므로 검사대상의 수를 줄일 수 있다.



(그림 7) 법선 벡터에 따른 검사 영역 선정

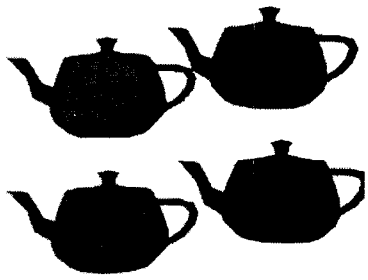
3-4-3. 충돌 후의 반응

법선 벡터의 방향이 주어진 범위 내에 포함되는 방향인 경우, 모든 샘플링 구를 대상으로 충돌 탐지 검사를 수행한다. 이때, 충돌 깊이를 비교하여 가장 깊이 들어간 샘플링 구의 깊이를 기준으로 각 물체를 현재 이동 방향의 반대방향으로 이동 시키므로, 잘못된 충돌 탐지를 수행하지 않는다.

4. 구현 결과

구현 결과는 간단히 두 개의 물체를 위치시킨 후 임의의 이동방향을 주어 움직여 충돌 반응을 검사하는 것으로, 정확한 충돌검사의 가능성을 확인할 수 있었다. 분할된 노드의 개수는 총 31 개 이지만, 자식 노드의 아이디를 맞추기 위한 노드를 제거하면, 실제 노드의 개수는 9 가 된다. 노드의 개수가 증가할수록, 즉 하나의 리프 노드가 포함하는 꼭지점의 인덱스의 개수가 감소할수록, 검사비용은 더 줄어들 것이다. 따라서 효과적인 공간분할 알고리즘을 사용하거나, 노드의 분할이 세분화 된다면 더욱 빠른 속도를 얻을 수 있을 것이다. 속도 면에서 비교한다면, 모든 연산이 구 연산인 샘플링 트리 방법이 매번 이동 할 때마다 복잡한 바운딩 볼륨을 업데이트 해야 하고, 충돌 검출 시에도 복잡한 연산을 해야 하는 다른 바운딩 볼륨 계층구조방법에 비해 빠른 것은 자명하다.

(그림 8)는 충돌 탐지 결과 화면으로, 정확한 충돌 검사가 이루어 지고 있음을 보여준다.



(그림 8) 충돌 시점에서의 물체
(위: 충돌 직후, 아래: 충돌 후의 반응)

5. 결론 및 향후 연구과제

본 논문에서 제시한 방법은 꼭지점 샘플링 트리를 이용하여 가벼운 계산으로 효율적인 충돌탐지를 수행하는 것이다. 샘플링 트리를 이용하면 외곽의 꼭지점과 구를 이용한 샘플링으로 보다 정확한 결과를 얻을 수 있다. 이러한 효율적인 충돌 탐지는 실시간 실행이 엄격히 요구되는 게임 같은 어플리케이션에 더 큰 사실감을 부여해, 사용자의 몰입감을 증대시킬 것이다.

여기에 덧붙여 공간이 넓은 가상 환경 어플리케이션 같은 경우, 효과적인 공간분할을 더한다면 여러 물체를 대상으로 하는 충돌 탐지에서 실제 충돌 가능성이 있는 물체를 선정하는데 더욱 효율적일 것이다.

참고문헌

[1] P.M Hubbard, "Collision detection for interactive graphics applications," *IEEE Transactions on Visualization and Computer Graphics*, 1995, pp.218-230.

[2] S. Gottschalk, M. C. Lin and D. Manocha, "OBB Tree : A Hierarchical Structure for rapid Interference Detection," *ACM Siggraph96*, 1996, pp.171-180.

[3] J.T. Klosowski, M. Held, J.S.B. Mitchell, H. Sowizral, and K. Zikan, "Efficient Collision detection Using Bounding Volumes Hierarchies of k-DOPs," *IEEE Transactions on Visualization and Computer Graphics, 1998*, pp.21-36.

[4] Gino V.D. Bergen, "Efficient Collision detection of complex Deformable models using AABB trees," *ACM's Journal of Graphics Tools*, 1997, pp.1-14.

[5] Taosong He, "Fast collision detection using QuOSPO trees," *Proceedings of the 1999 symposium on Interactive 3D graphics*, 1999, pp.55-62.

[6] A. Gregory, M.C. Lin, S. Gottschalk and Taylor, "A Framework for fast and Accurate Collision detection for haptic Interaction," *IEEE Transactions on Virtual Reality*, 1999, pp.38-45.

[7] Stephen A. Ehmann, Ming C. Lin, "SWIFT: Accelerated Proximity Queries Using Multi-Level Voronoi Marching," *Intelligent Robot and Systems*, 2000, pp. 2101-2106.

[8] S. Redon, A. Khedder, S. Coquilart, "Contact: Arbitrary in – between motions for collision detection," *IEEE International workshop on Robot and Human Interactive Communication*, 2001, pp.106-111.