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

Inverse BSP 트리와 구와의 효과적인 충돌검사

이동춘*, 김혜선*, 박찬용*, 장병태*
*한국전자통신연구원 증강현실연구팀
e-mail : bluepine,hsukim,cypark,jbt@etri.re.kr

An efficient collision detection between inverse BSP tree and bounding sphere

Dong-Chun Lee*, Hye-Sun Kim *, Chan-Yong Park, Byung-Tae Jang *
*Augmented Reality Research Team,
Electronics and Telecommunications Research Institute

요 약

충돌검사는 사실감 있는 3 차원 공간을 시뮬레이션 하기 위해서 필수적인 요소이다. 본 논문에서는 BSP 로 표현될 수 있는 정적 게임환경과 Bounding Sphere 로 표현될 수 있는 움직이는 물체와의 효과적인 충돌검사 방법에 대해서 기술하고 있다. 본 논문에서 제안하는 충돌검사 알고리즘은 단 하나의 BSP 트리를 사용하여 임의의 크기의 bounding sphere 와 충돌검사를 할 수 있다는 장점을 가진다.

1. 서론

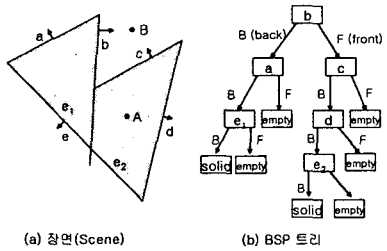
충돌검사는 사실감 있는 3 차원 공간을 시뮬레이션 하기 위해서 필수적인 요소이며, 움직이는 물체의 수가 증가함에 따라 충돌검사의 속도 향상 문제는 더욱 중요한 문제이기에 지금까지 이에 대한 연구 또한 많이 이루어졌다[5,7,8]. Cohen[4]과 Mirtich[10]는 움직이는 convex polyhedra 간의 빠른 충돌 검사 알고리즘을 발표하였으나 이는 물체들이 convex 라는 조건을 만족해야 한다는 단점이 있으며, 또한 이들은 정적인 물체와 동적인 물체의 구분하지 않는 충돌검사 알고리즘이다. Gottschalk[6]은 OBB 트리를 이용한 임의의 모양의 물체의 정확한 충돌 검사 알고리즘을 제시하고 있으나 수행시간이 많이 걸린다는 단점이 있다[9].

본 논문에서는 정적인 환경하에 있는 동적인 물체의 충돌검사를 위해 정적인 환경에 대해서 BSP 트리를 구성하여 충돌검사를 수행하는 방법을 사용한다. BSP 트리를 이용한 충돌검사는 지금까지 많은 연구가 되어져 왔으며[1, 11, 12, 13, 14], 여러 응용 프로그램에서 사용되어져 왔다[2,3]. 그러나 BSP 트리를 이용한 대부분의 충돌 검사 방법은 사용될 Bounding Sphere 의 크기에 따라 정적 환경을 재구성하는 방법을 사용하기 때문에 크기가 다양한 Bounding Sphere 를 사용 할 경우 여러 개의 BSP 트리를 구성해야 해야 한다는 단점이 있다. Stan 은 하나의 BSP 트리를 이용하여 다양한 크기의 Bounding Sphere 와의 충돌 검사 방법에 대하여 제시하였으나, 이 방법은 BSP 트리와 Bounding Sphere 와의 충돌 검사를 BSP 트리와 선분(line segment)과의 충돌 검사 알고리즘으로 변형한 방법으로[9] 본 논문에서 제시하는 BSP 트리와 점과의 충돌 검사 알고리즘에 비해 속도면에서 비효율적이다 할 수 있다.

본 논문의 구성은 다음과 같다. 2 장과 3 장에서는 BSP 트리와 BSP 트리를 이용한 점의 충돌 검사 방법에 대하여 기술하고 있으며, 4 장과 5 장에서는 Inverse BSP 트리와 Inverse BSP 트리를 이용한 구의 충돌 검사 방법에 대하여 기술하고 있으며 마지막으로 6 장에서 실험결과에 대하여 기술하고 있다.

2. BSP 트리

BSP 트리란 장면을 구성하는 모든 면들에 대해서 기준면을 선택하고, 그 기준면을 기준으로 다른 면들이 기준면의 앞에 있는지 혹은 뒤에 있는지를 나타내는 이진 트리(binary tree)를 말한다. BSP 트리에서 각 노드는 기준면의 정보가 들어가 있으며, 오른쪽 자식 Tree 에는 기준면의 앞쪽에 있거나 기준면과 동일한 평면상에 있는 면(기준면과 법선벡터(Normal)가 같으며, 거리(distance)가 0 인 면)들의 정보가 들어 있으며, 왼쪽 자식의 트리에는 기준면의 뒤쪽에 있는 면들의 정보가 들어 있다.



(a) 장면(Scene) (b) BSP 트리

그림 1. Scene과 BSP트리

그림 1 의 (a)와 같이 구성된 Scene 의 BSP 트리는 그림 1 의 (b)와 같다. 여기서 면 e 는 기준면 b 에 의해 e1, e2 2 개의 면으로 나뉘어진다.

3. BSP 트리와 점과의 충돌검사

Scene 상에 존재하는 임의의 점이 Scene 을 구성하는 면과의 충돌되었는지 여부는 그 Scene 에 대응하는 BSP 트리의 각 노드를 따라가면서 수행하게 된다. 그림 1 의 (a) 에서 보면 a,b,c,d,e 5 개의 면으로 둘러싸여진 부분은 solid 영역으로 이곳은 갈 수 없는 곳이다. 그림 1 에서 점 A 의 경우 면 b 의 앞쪽에 있으므로 BSP 트리상의 node c 로 가게 된다. 그리고 A 는 면 c 의 뒤쪽에 있으므로 node d 로 가게 된다. 이와 같은 방법으로 node e2 를 거쳐 solid node 에 도달하게 되므로 점 A 는 solid 영역 안에 있다고 판단하게 된다. 또한 점 B 의 경우는 node b, c 를 거쳐 node empty 에 도달하게 되어 움직일 수 있는 빈 공간에 있다는 것을 알 수 있다.

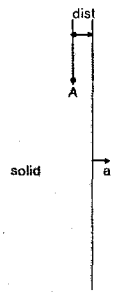
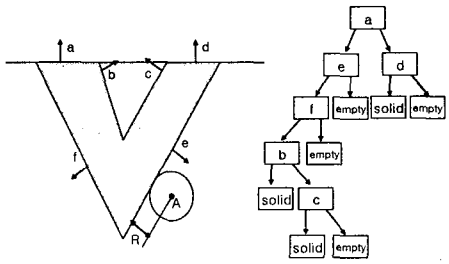


그림 2. 임계값

이렇게 충돌여부를 판단할 점이 BSP 트리 상의 각 node 를 따라가기 위해선 그 점이 면의 앞면에 있는지 혹은 뒷면에 있는지를 판단할 수 있어야 하며 이는 점과 평면과의 거리를 구함으로써 가능하다. 그런데 일반적으로 Scene 을 구성하는 면의 점이나 법선 벡터들은 실수값을 가진다. 그리고 실수인 두 값의 비교에는 항상 임계값(thresh hold)값을 수반하게 된다. 즉 점이 평면의 앞쪽에 있느냐, 뒤쪽에 있느냐 혹은 평면상에 있느냐는 것은 점과 평면과의 거리(distance)가 임계값을 넘어서느냐 혹은 임계값 안에 존재하느냐에 의해 결정된다.

그림 2 에서 점 A 가 면 a 의 안쪽으로 dist 거리만큼 들어가서 solid 영역에 있다고 가정하자. 만약 dist 값이 임계값인 th 값보다 크게 되면 점 A 는 solid 영역에 있는 것이지만 dist 값이 th 보다 작거나 같다면 점 A 는 면 a 상에 있는 것과 같으며 이는 점 A 가 빈 공간에 있다는 말이 되므로 Scene 과 점 a 는 충돌 되지 않은 것이 된다.

4. Inverse BSP 트리



(a) 장면 (b) BSP트리

그림 3. 상이한 Inverse BSP트리 구조의 Scene

Inverse BSP 트리란 장면을 구성하는 모든 면들의 법선 벡터에 음의 값을 취하여 이를 바탕으로 BSP 트리를 구성한 것을 말한다. 대부분의 Inverse BSP 트리의 모양은 BSP 트리를 좌우 바꾼 모양과 같다. 그렇지만 그림 3 와 같이 비교 대상이 되는 면(d)이 기준면(a)과 동일한 평면상에 있고 법선 벡터가 같을 경우는 Inverse BSP 트리는 그림 4 와 같이 완전히 다른 모양을 하게 된다.

5. Inverse BSP Tree 와 구의 충돌 검사

본 논문에서 제시하는 장면과 구와의 충돌검사는 다음과 같은 과정을 거친다.

단계 1. Inverse BSP 트리를 구성한다.

단계 2. 실수계산을 위한 임계값(th)을 구의 반지름(R)값으로 설정한다.

단계 3. BSP 트리의 각 node 를 따라가면서 다음의 과정을 거친다.

```
CollisionCheck(node n,vector pos)
 if (n is an empty leaf)   return 0 // 충돌 되지 않았음
 if (n is a solid leaf)    return 1 //충돌 되었음
 if(pos 값이 node 의 앞쪽에 있다면)
    return CollisionCheck(node->front,pos)
else {
    if (pos 값이 node 의 뒤쪽에 있다면)
      return CollisionCheck(node->back,pos)
    else{
      if (pos 값이 node 의 임계값 안에 있다면) {
```

```
        hit = CollisionCheck(node->front,pos)
        if(hit==1)
                return CollisionCheck(node->back,pos)
        else    return 0
    }
  }
}
```

단계 4. 단계 3 에서 구한 값을 반전시켜(벽과 충돌하지 않았다면 벽과 충돌한 것으로, 벽과 충돌하였다면 충돌하지 않은 것으로) 결과값을 돌려준다.

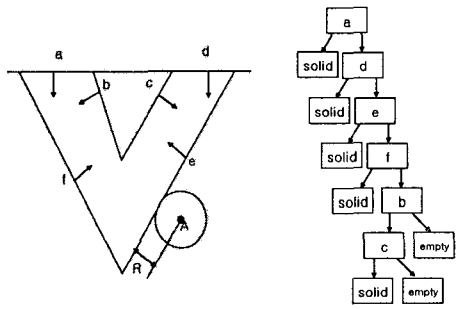


그림 4. Inverse BSP 트리

위의 방법을 토대로 해서 그림 3 과 같이 구성된 장면과 반지름이 R 인 구와의 충돌검사를 수행 해 보면 다음과 같다. 먼저 그림 3 에 대응하는 Inverse BSP 트리인 그림 4 를 구성한 다음, 실수값의 비교를 위한 임계값을 반지름값인 R 로 설정한다. 그리고 구의 중심점의 위치값을 가지고 Inverse BSP 트리를 따라가게 되면 다음과 같은 순으로 node 를 따라가게 된다. 양의 작은 ε 에 대해 만약 node e 와 점 A 와의 거리가 R+ε 이라면 node a, node d, node e, node solid 로서 충돌되었다고 판단되어지지만, node e 와 점 A 와의 거리가 R-ε 이라면 node a, node d, node e, node f, node b, node c, node empty 로서 빈 공간상에 점이 있는 것으로 판단되어진다. 따라서 단계 4 에 의해 node e 와 점 A 와의 거리가 R+ε 이라면 구가 충돌되지 않았다고 판달 할 수 있으며, node e 와 점 A 와의 거리가 R-ε 이라면 장면과 구는 충돌되었다고 판단할 수 있다.

6.실험 결과

본 논문에서는 Inverse BSP 트리를 이용한 움직이는 물체와 충돌검사 알고리즘에 대해서 기술했다. 앞에서 기술한 바와 같이 상기 알고리즘의 수행 시간 및 잉여 메모리를 사용하지 않는다는 장점을 지니고 있다. 그림 5, 6 은 상기 알고리즘에 바탕을 하여 충돌검사는 한 결과로 흰색의 구가 위쪽 방향으로 움직이는 물체이며 충돌에 의해 더 이상 움직이지 못하는 상태이다. 실험결과에서 단일 면으로 구성된 벽에 대한 움직이는 물체의 충돌 검사는 정확하게 이루어지나, edge 부분(두 면이 만나는 부분)에서의 충돌검사는 약간의 에러를 가진다. 이러한 에러는 두 면이 이루는 각이 작을수록 크지며 각이 클수록 작아진다. Stan Melax 의 논문에서는 이러한 에러에 대한 해결책으로 edge 부분에 부가의 면을 첨가하는 방법을 제시하였으며, 본 논문의 방법에도 쉽게 적용이 가능할 것이다 [9].

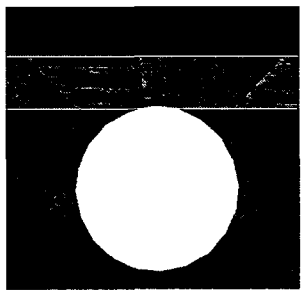


그림 5 단일면에서의 충돌 검사

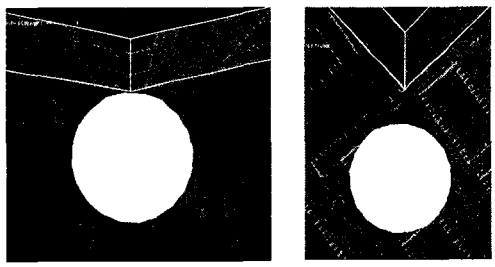


그림 6 Edge면에서의 충돌 검사

참고문헌

[1] M. de Berg, M. de Groot, M. Overmars. *Perfect Binary Space Partitions*. Canadian Conference on Computational Geometry, 1993.

[2] Bioware, Shiny, and Interplay: *MDK2*. Sega and PC Video Game, 2000.

[3] John Carmack, M. Abrash. *Quake's BSP compiler*. Id Software.

[4] Jonathan D Cohen, Ming C Lin, Dinesh Manocha, Madhav Ponamgi. *I-Collide, An Interactive and Exact Collision Detection System for Large-Scale Environments*. ACM Symposium on Interactive 3D Graphics (I3DG), pp 189-196, 1995.

[5] D. Dobkin, D. Kirkpatrick.: *A Linear Algorithm for Determining the Separation of Convex Polyhedra*. Journal of Algorithms 6, pp 381-392, 1985.

[6] S. Gottschalk, M. C. Lin, D. Manocha. *OBB-Tree: A Hierarchical Structure for Rapid Interference Detection*. SIGGRAPH, pp. 171-180, 1996.

[7] J. Klosowski, M. Held, J.S.B. Mitchell, H. Sowizral, K. Zikan: *Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs*. SIGGRAPH Visual Proceedings, 1996.

[8] S. Krishnan, M.Gopi, M. Lin, D. Manocha, and A. Pattekar. *Rapid and accurate contact determination between spline models using ShellTrees.* In Proceedings of Eurographics'98, 1998.

[9] S. Melax. *Dynamic Plane Shifting BSP Traversal.* In Proceedings of Graphics Interface, pp 213-220, 2000.

[10] B. Mirtich: *V-Clip.* Technical Report 97-05, Mitsubishi Electric Research Laboratory, 1997.

[11] Bruce F. Naylor, John Amanatides, William Thibault. *Merging BSP Trees Yields Polyhedral SetOperations.* SIGGRAPH, pp 115-123, 1990.

[12] Bruce F. Naylor. *Interactive Solid Modeling via Partitioning Trees.* Graphics Interface, pp 11-18, 1992.

[13] Bruce F. Naylor. *A Tutorial On Binary Space Partitioning Trees.* Computer Games Developer Conference Proceedings, pp 433-457, 1998.

[14] Michael S. Paterson and F. Frances Yao, *Efficient Binary Space Partitions for Hidden-Surface Removal and Solid Modeling.* Discrete and Computational Geometry, vol 5, pp 485-503, 1990.