제18회 한국정보처리학회 추계학술발표대회 논문집 제9권 제2호 (2002.11)

공간분할트리를 이용한 효율적인 충돌탐지 방법에 관한 연구

남승우*, 정연철**
*호남대학교 산업대학원
**호남대학교 미디어학부
e-mail:{namter, ycjeong}@honam.ac.kr

A Study on Method for Effective Collision Detection Using a Spatial Partition Tree

Seung-Woo Nam*, Yeon-Chul Jeong**
*Dept of ****, Honam University
**Division of Media, Honam University

요 약

게임에서 충돌탐지는 게임의 성능향상을 위해 중요하다. 본 논문에서는 효율적인 충돌탐지를 위해 BSP 트리를 사용한다. 공격에 사용되는 스프라이트와 공격의 대상이 되는 스프라이트를 트리로 구성하여 빠른 시간내에 충돌탐지를 행한다. 또한 스프라이트의 모양에 따라 경계 볼륨(bounding volume)을 구와 박스(box)를 선택적으로 사용하여 충돌탐지에서 발생하는 문제점을 해결한다.

1. 서론

게임은 가상 세계를 형성하고 다양한 전략 전술을 이용하여 적을 물리쳐가는 일종의 가상 체험이다. 물론 게임은 다양한 분야로 구분된다. 그렇지만 대부분의 게임에서는 적과 주인공과의 대치 상황으로 구분되고 전투를 통해 진행되는 방법이 일반적으로 쓰인다. 그런데 이와 같은 전투에서 필요한 요소가 충돌탐지이다.

게임을 구성하는 등장물체(이하 객체)들은 계속해서 상태가 변화한다. 물론 고정된 객체도 존재하지만 이동하여 새로운 상황에 직면하게 되는 경우가 많다. 따라서 게임에서는 다양한 물리적 현상이 발생하게 되고 게임 개발자들은 현실 세계의 물리적 기법을 기초하여 게임에 구현하기 위한 많은 노력을 기울였다. 그렇지만 게임에서는 모든 물리적 현상들을 그대로 표현하기 보다는 간단한 수학적 방법을 통해 구현하고자 한다.

게임에서 객체는 단일체(예를 들어, 비행기등)로 구성되는 경우도 있고 인체와 같이 복잡한 형태로 구성되기도 한다. 여러개의 구성체로 만들어진 객체의 경우 애니메이션 구현에서 물체의 이동은 트리를 이용한다. 객체는 이동과정에서 객체끼리 충돌이 발생하는데 이를 탐지하고 게임에 반영해 주어야 한다. 또한 객체 충돌했을 때 피해를 입는 부분과 공격을 가하는 부분으로 구분하여 게임에 반영해 주어야 한다.

2. 물체의 충돌탐지

물체와 물체가 부딪칠 때 충돌이 발생하고 이를 탐지하여 게임에 적용하여야 한다. 그런데 대부분의 복잡한 물체는 다각형(polygon)으로 모델링(modeling) 된다. 현재 가장 널리 사용되고 있는 모델링 방법은 삼각형 메쉬(mesh)를 이용한 방법이다. 그런데 다각형으로 표현된 물체의 충돌탐지는 많은 계산 시간이 필요하기 때문에 실시간(real-time)을 필요로 하는 게임에서 적용은 많은 문제를 일으킨다.

게임 내에서 객체들의 충돌은 2차원 객체의 충돌과 3차원 객체의 충돌에 따라 방법이 구분된다. 2차원 개체의 충돌탐지는 평면에서 문제이기 때문에 크게 어려운 문제는 아니다. 그러나 3차원 물체의 충돌을 처리하기 위해서는 공간상에 물체의 위치를 파

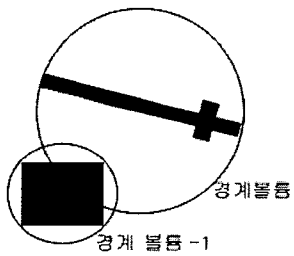
악하여 위치에 따르는 충돌여부를 검사여야 한다. 그래서 충돌탐지를 위해 경계볼륨이 많이 사용된다.

2.1경계 볼륨(bounding Volume)

경계 볼륨은 물체의 충돌을 위해 사용되는 것으로 물체를 포함하는 최대 원(3차원에서는 구)이거나 사각형(3D에서는 box)이다. 게임에서 표현되는 객체들은 이러한 경계볼륨을 이용한 충돌탐지를 통해 단순화된 문제로 해결된다. 본 논문에서는 2D와 3D에 대해 각각 경계 볼륨으로 구분하지 않고 표현하기로 한다.

2.1.1 기존의 충돌 탐지 방법

경계볼륨은 물체의 특성에 따라 사각형과 원이 모두 사용될 수 있다. 그런데 경계 볼륨을 무엇을 사용하는가에 따라 충돌 탐지 여부가 잘못된 경우가 발생할 수 있다는 것이다. 다음 그림의 예는 물체가 서로 충돌하지 않지만 경계 볼륨이 충돌하는 경우로 탐지된다.



(그림 1) 경계원을 사용한 충돌탐지

특히 2차원 게임에서 움직이는 객체를 스프라이트(sprite)라고 하는데 스프라이트의 경계 볼륨을 이용하여 충돌 탐지를 실행한다. 물론 대부분의 프로그래머들은 스프라이트 경계를 사각형을 사용한다. 그렇다 해도 스프라이트 경계 볼륨이 실제 물체보다 크게 정의되어(그림 1) 잘못된 충돌탐지로 인해 게임에서 잘못된 결과를 얻는 경우도 발생한다.

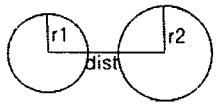
그래서 스프라이트 내의 모든 픽셀을 반복적으로 동일 픽셀 안에 다른 스프라이트 객체가 들어오는지 검사하는 억지 기법을 사용하거나 스프라이트가 화면에 그려질 때 경계볼륨을 나타내지 않기 위해 투명 픽셀 정보를 사용하는데 이 투명영역을 제외하고 충돌을 검사하는 방법 비트 배열 방법이 있다. 그러나 이러한 방법도 결과적으로 충돌탐지에 오류가 포함되어 사용이 어렵다. 따라서 본 논문에서는 경계볼륨을 객체의 형태에 따라 원 또는 사각형을 선택적으로 사용하고 이를 이용한 효율적 충돌 탐지를 위해 공간분할 트리를 사용하여 빠른 충돌 탐지 방법을 제안한다.

2.2 경계볼륨 충돌탐지

먼저 경계 볼륨으로 사용하는 원과 사각형에 대한 간단한 충돌 탐지에 관한 수학적 표현을 설명한다.

2.2.1 원과 원의 충돌 탐지

원과 원의 충돌 탐지는 두 원의 반지름의 합과 거리의 값으로 계산한다.



(그림 2) 원과 원 충돌

임의의 두원을 각각 c1, c2라 하고 반지름을 r1과 r2라고 가정하자. 또한 두 원의 중심간의 거리를 dist라고 했을 때 원의 충돌여부는 다음과 같다.

$$dist = \sqrt{(c1.x - c2.x)^2 + (c1.y - c2.y)^2}$$

```
bool Sphere_cross_Test() {
if(r1 + r2 < dist ) return false;
else true;
}
```

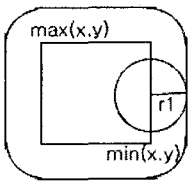
원에 대한 충돌 검사는 3차원에서는 z축에 대하여 새롭게 공간상의 거리를 계산하면 위의 방법으로 쉽게 얻을 수 있다.

2.2.2 원과 사각형 충돌탐지

원과 사각형의 충돌 탐지는 임의의 사각형과 원이 만나는 면적으로 계산할 수 있다.

```
bool Box_sphere_cross_Test() {
if(exit Cross) return true;
else false;
}
```

원이 사각형의 주위를 그리며 이동한 면적이 그림과 같이 되었을 때 원과 사각형은 서로 교차한다.



(그림 3) 원과 사각형
충돌

3. 트리(Tree)를 이용한 물체의 충돌탐지 제안

경계 볼륨을 사용하여 충돌 여부를 검사하는 경우 단순한 물체의 경우에는 물체의 형태에 따라서 사용할 경계볼륨을 결정하면 된다. 그렇지만 복잡한 물체인 경우 게임에서는 각 부분(part)에 따라서 담당하는 역할이 다르다. 예를 들어, 객체가 칼을 들고 공격하는 경우 칼에 충돌이 탐지 되는 경우 상대방은 피해(damage)를 입지만 반대인 경우는 주인공이 피해를 입도록 충돌탐지에 대한 세부적인 분류가 필요하다.

본 논문에서는 이러한 복잡한 객체의 충돌탐지를 위해 객체의 부분별 경계 볼륨을 트리로 구성한다. 이를 객체 트리로 정의한다. 또한 객체가 얻을 수 있는 공격부분과 방어부분에 대한 객체의 경계 볼륨이 미리 정의된다. 각 객체의 정의는 다음과 같다.

```
struct CharacterNode {
  int   charID;
  int   linkID;
  int   volType;
  POINT   min, max;
  int   CharType;
};
```

volType은 경계볼륨이 원이거나 사각형을 나타내기 위해 사용되는 필드이고 min, max는 사각 영역을 나타내고 만약 경계볼륨 타입이 원이라면 내접하는 영역으로 정의 된다. 그리고 CharType은 공격용 또는 방어용, 일반용으로 타입이 분류된다.

복잡한 객체를 구성하는 요소들은 서로 링크 되어 있고 트리의 상위 부분에 있는 노드일수록 공격을 받을 때 피해가 많이 가해지는 노드로 구성된다. charID는 게임에 참여하는 객체에 순서적으로 부여되는 값이다.

3.1 객체 공간분할 트리

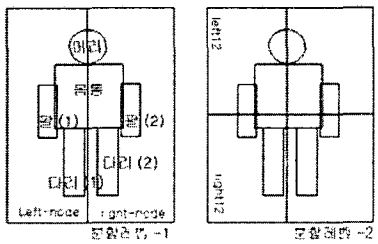
객체 공간 분할 트리는 객체의 중심을 기준으로 공간분할 이진트리로 작성된다. 트리는 객체 트리를 전체로 하는 경계볼륨을 사각형으로 하여 가장 긴 부분을 분할 판(cutting plane)으로 분할된다.

```
struct ObjectTreeNode {
  CharacterNode     id;
  POINT     min, max;  // 경계볼륨
  int       cut_plane;  // 분할판
  ObjectTreeNode  *lnode;
  ObjectTreeNode  *rnode;
};
```

```
Build_Object_Tree() {
1. if( depth >= N ) return;
2. Determine Cutting Plane
3. Split  box1, box2
   Make_Tree_Node(box1);
   Make_Tree_Node(box2);
```

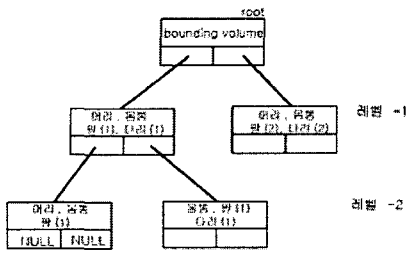
N은 트리 깊이(depth)가 너무 깊어지는 것을 방지하기 위해 사용되는 상수값이다.

예를 들어, 사람 형태를 객체 트리를 이용하여 객체 공간 분할 트리로 생성하는 경우를 살펴보자. 몸통 부분과 머리 두 팔과 그리고 두 다리에 대해 각각 CharacterNode가 전처리과정에서 생성되고 객체 트리가 생성된다.



(그림 4) 트리분할 예

(그림 4)에서 모두 5개의 CharacterNode가 객체 트리를 구성한다. 각각의 노드에 대해 linkID를 부여할 수 있는데 몸통부분이 캐릭터의 중요 기능을 하므로 linkID = 1로 부여된다. 그리고 팔과 다리 부분, 그리고 머리부분은 몸통에 연결되어 있기 때문에 하위 노드로 생성된다. linkID는 모두 2가 부여된다.



(그림 5) 객체 공간 분할 트리

3.2 트리를 이용한 충돌 탐지

충돌 검사를 위해 두 개의 공간분할 트리의 루트 노드의 경계 볼륨간의 충돌탐지를 실행한다. 만약 루트노드가 서로 충돌이 발생하지 않는면 두 개는 트리는 서로 충돌이 발생하지 않는다. 또한 충돌이 발생된 트리 노드는 필요에 따라서 공격, 피해에 관한 데이터 관리를 위해 해당 노드의 식별자를 구분하여 데이터에 적용한다. 그리고 노드의 특성에 따라 주요 노드에 대한 피해는 전체적인 가중치에 따라 다른 값으로 사용할 수 있다.

```
First_Collision_test(node, node1) {
    if( !Bounding_Cross_test(Node, Node)
        return false;
    if ( node is a leaf) {
        reutrn (chrID);
    } else {
    if(Bounding_Cross_test(left, Node) )
      return(chrID_left);
    else if(Bounding_Cross_test(right, Node))
      return(charID);
      else return false;
    }
}
```

객체의 충돌 탐지는 객체의 위치가 게임 진행됨에 따라서 상대적인 위치가 달라지기 때문에 트리의 공간의 위치에 많은 영향을 받는다. 그러므로 공간 분할 방식으로 생성된 트리는 중복된 노드를 갖는다. 충돌이 발생했을 때 각 트리의 노드는 이진트리 형태로 순회되기 때문에 충돌 여부 탐지 속도가 빠르다.

```
if( Bounding_Cross_test(Node1, Node2) ) {
case  Node1.volType, Node2,volType == Box
    Box_Cross_test()
case Node1.volType, Node2.volType == Sphere
    Sphere_Cross_test();
default :
    Box_Sphere_test();
```

4. 결론 및 향후 연구과제

충돌탐지에 사용되는 경계볼륨은 게임 객체 정의 과정에서 물체의 형태에 따라 결정된다. 객체 트리는 게임 내의 각각의 객체에 대해 형성될 수 있고 트리의 상위 노드는 하위 노드를 지배하는 구조로 정의된다. 각각의 객체 트리는 새로운 노드(게임에서 캐릭터)를 받아 들여 동적으로 트리의 노드가 변화할 수 있다. 또한 각각의 트리 노드는 공격과 방어용 노드로 분류되고 상대방이 가진 노드와의 충돌이 발생했을 때 객체에 미치는 영향을 쉽게 판단할 수 있다. 그리고 각각의 충돌탐지를 공간 분할 트리를 이용하기 때문에 물체와의 충돌 탐지를 실시간 내에 처리할 수 있기 때문에 효율적이다.

그렇지만 객체마다 트리가 생성되어 트리의 수가 너무 많아지기 때문에 메모리의 효율적 사용에 대한 문제를 해결해야 한다.

참고문헌

[1] Wolfgang F. Engel. Amir Gava, Andre LaMothe Beginning Direct3D Game Programming with DirectX.

[2] Lan Parberry Lean Computer Game Programming with DirectX.

[3] Foley, etc, Computer Graphics Principle and Practice, Addison-Wesley

[4] Edward Angel, interactive Computer Graphics A top-down approach with OpenGL, Addison-Wesley

[5] Hearn & Baker, Computer Graphics 2nd Edition, Prentice Hall.

[6] 류광역, Game Programming Gems2, 정보문화사

[7] 김도원, 최성, "게임역학과 스트라이트 객체간의 충돌탐지 방법에 관한 연구", 2002, 한국 게임학회 학술대회, pp. 127- 131.