물리 엔진을 이용한 3차원 게임 객체의 충돌 처리

강정훈*

*동명대학교

Using the Physics engine to collision detection of 3D Game Object

Jung-Hun Kang*

*Tong Myong University

E-mail : blackhydra@tu.ac.kr

요 약

본 논문은 3차원 게임 내에서 물리 엔진을 이용하여 다양한 객체들의 충돌 처리를 구현하는 방법과 그에 따른 문제점에 대해 설명한다. 특히, 3D 게임콘텐츠에서 실시간으로 유용하게 사용될 수 있는 알고리즘을 제안한다.

ABSTRACT

This paper proposes the collision detection of objects of variety how to implement method and corresponding problems in 3D game engine. Specially, I propose the algorithm using a game engine technique to produce 3D game contents.

키워드

Physics Engine, 3D game Programming, Collision Detection

Ⅰ. 서 론

지금까지 게임 분야에 있어서 하드웨어의 성능 향상은 대부분 그래픽이나 서버 처리에 초점이 맞추어져 왔으나 이제는 게임들의 완성도를 평가하는 데 있어 사용자가 실제와 같이 느낄 수 있는 게임 플레이가 가능한가에 게임 기술이 맞추어지고 있다.**[1]**

3D 게임의 비중이 큰 현재의 게임 산업에서는 더욱 현실에 가까운 애니메이션과 특수 효과를 구현하기 위해 물리적인 처리를 요구하고 있다. 게임 내에 물리 엔진을 직접 구현 또는 포함하였을 경우 사용자들의 흥미를 높일 수 있는 사실성 있는 게임을 제작할 수 있다.

본문을 통해 물리 엔진을 이용한 충돌처리 방법을 알아보고, 이를 구현하여 객체 간의 충돌을 처리, 사용자가 조작 가능한 Player 객체를 통한 충돌 처리, 3D 렌더링 엔진을 통한 PhysX의 구현 방법에 대해 알아보도록 한다.

Ⅱ. 물리 엔진을 이용한 충돌 처리

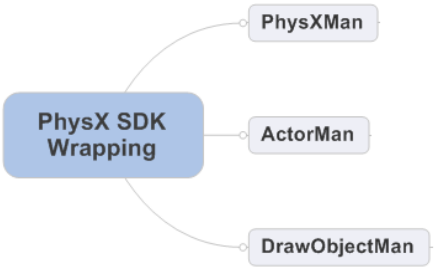
물리 엔진은 크게 3가지로 구분할 수 있다. 완전 상용 엔진, 공개 형 상용 엔진 그리고 완전 공개 엔진으로 분류된다.

물리 엔진에서의 충돌처리를 구현하기 위해서 먼저 엔진의 역할 및 기능에 대해서 알아야 한다. 물리 세계에서는 크게 **강체(Rigid Body)**, **유체(Fluid)**로 구분된다.

본 논문에서는 물리 엔진을 이용한 충돌 처리를 위해 두 가지의 물리 엔진을 사용하였다. 또한 렌더링 및 3D 객체 모델 데이터를 쉽게 사용하기 위해 **Ogre(Object-Oriented Graphics Rendering Engine)[1]**을 사용하였다. 물리 엔진으로는 앞서 소개한 완전 공개 엔진인 ODE와 공개 형 상용 엔진인 **PhysX[2]**를 사용하였다. 각각 약간의 용어와 기능상의 차이는 있지만 물리를 기반한다는 내용에 대해서는 다를 것이 없다.

Ogre3D Rendering 엔진에서 PhysX를 사용하

기 위해서 가장 먼저 PhysX SDK를 Wrapping하여 사용하기 쉽도록 PhysXMan, ActorMan, DrawObjectMan의 세 가지 클래스를 직접 구현했다.



[그림 1] PhysX SDK Wrapping Class

SetupMatrix() 함수를 이용하여 Ogre3D와 PhysX 간의 정보를 교환한다. Box, Sphere, Capsule 등 형태가 정해진 객체의 경우 위의 방법을 통해 간단히 구현할 수 있지만 TriangleMesh의 경우는 vertex 및 index의 정보가 필요해서 좀 더 복잡한 방법을 통해 구현해야 한다.

여러 다양한 Scene Model을 손쉽게 제어하기 위해서 XML 정의 데이터를 만들어서 화면 내에 출력될 mesh 데이터들의 정보를 정의함으로써 손쉽게 Scene을 구성, 편집할 수 있다.

사용자가 조작할 수 있는 Player Actor를 위해서는 PhysX에서 제공하는 CharacterControl 클래스를 사용해야 한다. 먼저 캐릭터를 제외한 Actor(Object)들의 그룹을 지정해야 한다. 충돌 처리가 되지 않을 객체(GROUP_NON_COLLIDABLE), 충돌 처리가 되며 이동 되지 않는 객체(GROUP_COLLIDABLE_NON_PUSHABLE), 충돌 처리가 되며 이동되는 객체(GROUP_COLLIDABLE_PUSHABLE)의 세 개의 그룹으로 구분해야 한다. 기본적으로 제공하는 CharacterControl 클래스를 약간의 수정을 통해 사용하며 특별하게 변환한 내용은 없다. PhysX에서 지원되는 CharcaterControl은 Box 형태와 Capsule 형태로 정확한 TriangleMesh 형태는 지원하고 있지 않다. CharacterControl에 관한 것은 PhysX Tutorial에서 자세히 확인할 수 있다. 다만 Player Actor의 크기에 대해서는 직접적인 처리가 필요하다.

II. 적용 결과

PhysX는 CharacterControl 클래스를 사용하기 전과 사용한 후의 결과가 다르다. 사용하기 전에는 다양한 물리적인 문제가 발생했다.

CharacterClass는 Box와 Capsule의 충돌 처리만을 구현해 두었다. 개발자가 원하는 플레이어 객체에 Box나 Capsule을 둘러싸 충돌 처리는 하는 방법을 제공한다. 움직임에 대해서는 개발자의 세부 설정이 필요한 부분이 있지만 대부분의 충돌이 원활하게 작동했으며 점프 및 높이의 제한과 같은 방법이 구현되어 있어 적용하는 데 있어 편리한 이점이 있었다. 약간의 문제점이라고 하면 PhysX와 Ogre3D 렌더링 엔진을 결합하는 부분에서의 시간 비용이 든다는 점이다.

[그림 2] Player Actor를 추가한 모습

V. 결 론

충돌 처리를 위해서 물리 엔진을 사용하는 것은 매우 효율적인 방법이었다. 직접적인 충돌 처리 구현에 드는 시간적인 비용을 줄여주었으며, 직접 구현한 것보다 더 뛰어난 충돌 효과를 보여주었다. 또한 충돌 이외의 다양한 효과를 얻을 수 있다.

약간의 문제점으로는 특성에 맞는 충돌 처리를 적용하기 위해서 세부 설정들이 필요하다. 또한 모든 강체들에 대해 충돌이 처리되지 않는다는 단점이 있다.

전반적으로 직접 충돌처리를 구현하는 것보다 효과적이었으며 약간의 설정을 통해 보다 좋은 물리적인 충돌 효과를 얻을 수 있었다.

참고문헌

[1] 이기석 외 4명, “게임 물리 기술 동향”, ETRI(2007)

[2] Ogre 공식 사이트 : http://www.ogre3d.org

[3] PhysX 공식 사이트 : http://www.nvidia.com/object/physx_new.html