한국컴퓨터정보학회 동계학술대회 논문집 제25권 제1호 (2017. 1)

Unity3D엔진에서 투사체 충돌처리 방법의 비교분석

서종석*, 강명주O
O*청강문화산업대학교 게임콘텐츠스쿨
e-mail: slowgg@naver.com*, mjkkang@ck.ac.krO

Comparative Analysis of Projectile Collision Detection Methods in Unity3D

Jong-Seok Seo*, Myung-Ju KangO
O*School of Game, Chungkang College of Cultural Industries

● 요 약 ●

게임에서 캐릭터와 몬스터 간, 캐릭터와 캐릭터 간, 피격처리 등을 위해서는 충돌처리가 필요하다. 본 논문에서는 Unity3D 엔진에서 지원하는 충돌처리 방법을 비교분석하였다. Unity3D 엔진에서 지원하는 충돌처리 방법은 Collider 방식과 Raycast 방식이 있으며, Collider 방식은 정적이거나 속도가 아주 느린 오브젝트에 사용하고, Raycast방식은 속도가 빠른 투사체를 구현하는데 적합하다.

키워드: Unity3D, Collision Detection, Raycast

I. Introduction

게임개발에서 캐릭터와 오브젝트 간, 캐릭터와 몬스터 간, 피격 등의 충돌처리는 반드시 구현되어야 한다. 본 논문에서는 Unity3D 엔진에서 제공하는 충돌처리 방식인 Collider 방식과 Raycast 방식의 장단점을 비교분석하였다.

II. Projectile Collision Detection Methods in Unity3D

1. Collider 방식

Unity3D엔진에서는 물체의 충돌체크를 위해 많은 API를 제공하고 있다. 그 중 대표적으로 물체에 Collider컴포넌트와 RigidBody컴포넌트를 추가하여 오브젝트간의 물리적인 충돌체크를 가능하게 해준다. 이 방식은 물체의 모양에 따라 Collider의 모양이 정해지는 MeshCollider[1]기능이 있기 때문에 물체의 Mesh와 완벽하게 일치한 충돌체크를 가능하게 해준다. 그러나, Mesh가 복잡할수록 퍼포먼스가 떨어지는 단점이 있다. Mesh가 복잡할 경우 사용자가 직접 MeshCollider를 수정하여 완벽하게 일치하지는 않지만 충돌처리 부분에서는 문제가 되지 않을 정도로 간략화하여 사용할 수 있다.

2. Raycast 방식

Raycast방식은 투사체의 위치에서부터 시작되는 Ray를 생성하여 투사체가 이동하기 전 Ray에 충돌하는 오브젝트가 있는지 여부를 먼저 확인하는 방식이다. Raycast 방식은 오브젝트에 컴포넌트를 붙이는 방식이 아니기 때문에 LayerMask를 매개변수로 받을 수 있게 하여 특정 Layer만, 혹은 특정 Layer만 제외하고 충돌이 가능하게 할 수 있다. 또한 거리 조정이 가능하기 때문에 보통 투사체의 속도에 deltaTime을 곱하여 한 프레임 동안 이동 될 투사체의 거리만큼만 Ray를 체크하도록 한다.

III. Comparative Analysis of Projectile Collision Detection

Collider 방식으로 투사체를 구현하는 경우 Collider가 투사체에 완벽하게 일치한 상태로 이동 후 판정이 되어버리기 때문에 투사체가 장애물에 충돌한 지점이 겹쳐지거나 투사체의 속도가 빠를 경우 장애물을 넘어가버리는 상황이 발생할 수 있다.

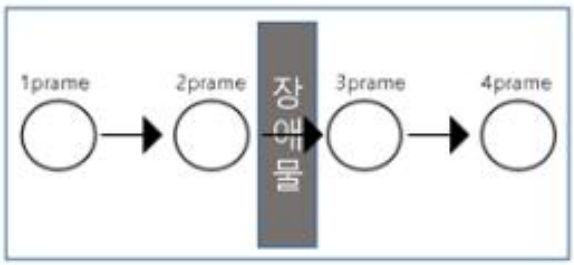

Fig. 1. Disadvantages of Collider Method

또한 Trigger속성인 경우 Unity가 지원하는 API에서는 부딪힌 장애물의 정확한 접촉 지점을 알려주는 'contects' 속성을 사용 할 수 없기 때문에 불편한 점이 있다[2].

Raycast 방식으로 투사체를 구현하는 경우 Collider 방식의 문제점을 해결할 수가 있다. 투사체가 이동한 후 충돌판정을 하는 것이 아니라 현재 위치에서 생성한 Ray를 가지고 충돌판정을 하기 때문에 투사체가 다음프레임에서의 충돌을 미리 체크할 수 있다.

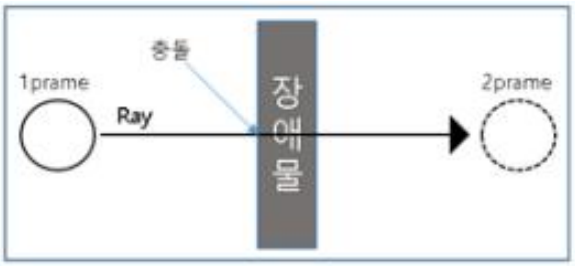

Fig. 2. Advantage of Raycast Method

또한 Raycast 방식을 사용하면 Ray에 충돌한 오브젝트의 정보를 저장하는 'RaycastHit' 속성을 사용 할 수 있다[3]. 하지만 Raycast방식의 단점은 투사체의 표면체크를 정밀하게 하기 어렵다는 점이다. 만약 투사체의 Ray를 하나만 생성한다면 아래 [그림 3]과 같은 상황이 발생할 수 있다.

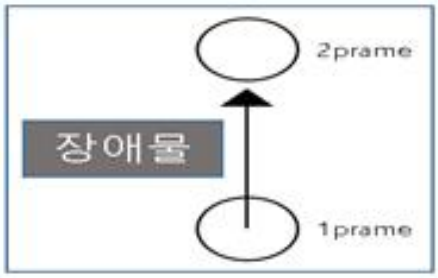

Fig. 3. Disadvantage of Raycast Method

그러나 이를 해결하기 위해 물체의 상, 하, 좌, 우 등 여러 개의 Ray를 추가로 생성하여 체크할 수 있지만 물체의 Mesh가 복잡한 경우에는 정확도가 떨어질 수 있다.

IV. Conclusions

본 논문에서는 Unity3D엔진의 대표적인 두 가지 충돌처리 방식에 대해서 비교하였다. Collider 방식은 Mesh와 정확하게 일치하는 충돌체를 만들 수 있지만 충돌 시 부딪힌 장애물 사이에 끼거나 속도가 빠를 경우 장애물을 넘어가는 단점이 있고, Raycast방식은 Collider 방식처럼 정확한 표면처리는 불가능 하지만 장애물에 끼거나 넘어가는 상황은 일어나지 않게 된다. 또한 Collider 방식의 경우 Trigger속성이 체크되면 'contects' 속성을 사용 할 수 없게 되는데 이는 상당히 큰 단점이 된다. 보통 투사체는 총알이나 미사일 등을 구현하는데 사용하는데 투사체가 부딪힌 정확한 접촉지점을 알 수가 없다면 해당 지점에 추가 작업을 할 수가 없게 되기 때문이다. 따라서 Collider 방식은 정적이거나 속도가 아주 느린 오브젝트에 사용하고, Raycast방식은 속도가 빠른 투사체를 구현하는데 적합하다.

References

[1] https://docs.unity3d.com/Manual/class-MeshCollider.html
[2] https://unity3d.com/kr/learn/tutorials/topics/physics/colliders
[3] https://docs.unity3d.com/ScriptReference/RaycastHit.html