1인칭 충돌회피 게임 개발

안성옥*, 이용운*, 김수균*

요약

대부분의 충돌회피게임은 3인칭 시점에 2D 그래픽을 사용하고 있다. 본 논문에서는 이러한 방식에서 탈피한, 3D 그래픽을 이용한 1인칭 시점의 충돌회피 게임을 제작하는 방법에 대해 제안한다. 기존의 3인칭 시점에 상하좌우 네 방향으로만 움직이는 방식을 벗어나, 시점을 1인칭으로 변환하고 FPS(First person shooter)와 같은 시점과 이동방식을 제공하며, 기존의 2D 게임에서 사용되던 축이 고정된 오브젝트의 충돌인 AABB(Axis Aligned Bounding Box)가 아닌 축이 수시로 변하는 OBB(Oriented Bounding Box) 방식을 사용함으로 써, 3D그래픽에서도 2D그래픽에서처럼 정교한 충돌 검출 기능이 가능하도록 제작하는 것을 목표로 한다. 본 논문에서는 저수준 PC에서도 사용 할 수 있도록 개발한 것이 장점이다.

Development of First Person Collision Avoidance Game

Syungog An*, Yong-Un Lee*, SooKyun Kim*

ABSTRACT

Most collision avoidance games are using two dimensional graphics in third person point of view. This thesis proposes an unconventional developmental method for first person collision avoidance games using three dimensional graphics. This method breaks away from the up, down, left and right movements of the traditional third person point of view. The point of view is changed to first person and the movements are much like that of First Person Shooters (FPS). Also, the new method uses an Oriented Bounding Box (OBB) in which the axis is frequently changing rather than the conventional Axis Aligned Bounding Box (AABB) used two dimensional games in which the axis is fixed in one position. The objective of the OBB is to allow elaborate collision detection skills in three dimensional graphics, just as in two dimensional graphics. The advantage of this method is that it was developed to be compatible even in low quality PCs.

Key Words : OBB, AABB, FPS, Game

* 배재대학교 게임공학과(✉sungohk@pcu.ac.kr)
· 제1저자(First Author) : 안성옥 · 교신저자(Correspondent Author) : 김수균
· 접수일(2012년 1월 2일), 수정일(1차 : 2012년 2월 1일), 게재확정일(2012년 2월 3일)

Ⅰ. 서 론

회피 미니게임은 게임이라는 것이 생기면서부터 존재해온 매우 오래된 게임으로, 쉬운 조작성과 강한 중독성을 가지고 있는 게임이다. 본 논문에서는 이러한 회피게임을 3D로 제작하여 원래 게임과는 다른 재미를 선사한다. 본 논문에서는 게임엔진에서 제공하는 그래픽의 특정화된 기술을 사용하지 않고 DirectX 라이브러리를 이용하여 게임의 세세한 부분까지도 직접 구현한다. 특히 OBB[1]를 이용한 충돌 감지 구현을 통해, 3D 그래픽 상에서도 세밀하고 빠른 충돌감지 기능을 제공한다.

Ⅱ. 미니 회피게임 설계 및 제작

본 절은 DirectX를 이용한 1인칭 회피게임이 구현되는 부분에 대한 내용으로 게임에 사용된 개발기술의 구현 방법에 대해 설명 할 것이다.

본 논문에서 게임 개발 흐름도는 다음과 같다.

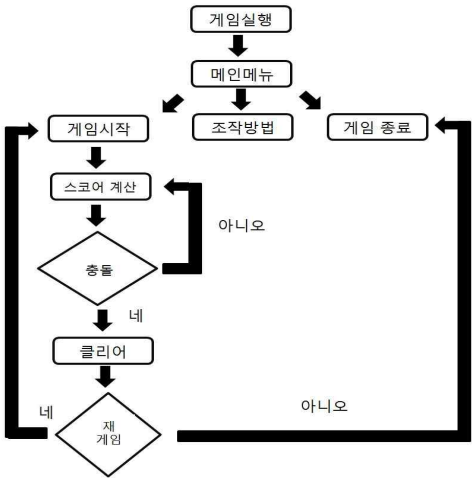

그림 1. 전체 흐름도
Fig 1. System flowchart

게임을 실행하고, 메인메뉴에서 게임의 조작 방법이나 게임 밖으로 나갈지를 사용자들이 선택하게 한다. 만약 게임을 진행하게 된다면, 간단하게 미사일과 충돌 했는지를 검사하여 충돌되게 되면 게임이 끝나고, 최종 점수를 사용자에게 보여주게 된다. 이러한 단순한 게임은 보통 기록 경신을 통해 사용자들의 몰입감을 높이게 된다.

2.1 충돌 감지

충돌 감지는 두 물체가 서로 맞부딪치거나 움직이는 물체를 접촉할시 사물의 상태를 검사하는 것이다. 회피게임을 구현하기 위해서는 필수 요소이며 가장 중요한 부분이다.

충돌감지를 통해서 게임세계를 현 세계와 동일한 것을 개발할 수 있으며 많은 것을 이룰 수가 있다. 캐릭터와 미사일 사이에 충돌 감지를 통해 게임의 완성도를 높이는데 크게 작용한다.

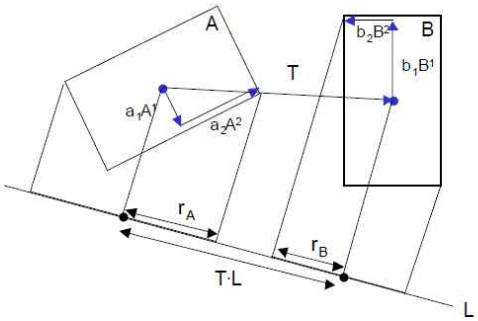

그림 2 분리축 이론을 이용한 알고리즘
Fig 2. Algorithm using Separating Axi Theory

그림 2는 분리축 이론(Separating Axi Theory)[2]을 이용한 알고리즘으로 본 논문에 제시하는 방법이다. 분리축 이론이란 '임의의 두 볼록다면체 A,B에 대해 어떤 축이 존재해서 그 축으로 다면체들의 투영들의 구간이 서로 겹치지 않는다면, A, B는 서로 분리 되어있다'라는 이론으로, A의 면 중 하나의 벡터, B의 면

중 하나의 벡터, A의 변과 B의 변에 동시에 수직인 축들 중 하나를 분리축으로 지정하고, 그림 1과 같이 분리축에 두 볼록다면체를 투영시켜 충돌이 되었는지 유무를 감지한다.

본 방법을 사용하게 되면 기존의 축이 고정된 AABB를 이용한 충돌 감지 보다 더욱 다양한 환경과 상황에서 사용이 가능하며, 세밀한 충돌 감지가 가능하다.

충돌 감지에 사용되는 분리축은 A의 면법선벡터 6개, B의 면법선벡터 6개, A의 변 12개와 B의 변 12개의 조합으로 만들 수 있는 동시에 수직인 축 144개를 모두 더한 156개 이지만, 볼록다면체는 육면체 오브젝트이기 때문에 서로 평행한 선들을 제외하면 A의 면법선벡터 3개, B의 면법선벡터 3개, A의 변 3개와 B의 변 3개의 조합으로 만들 수 있는 동시에 수직인 축 9개를 더한 15개로 압축 할 수 있다.

이렇게 구해진 15개의 분리축과 각 오브젝트의 중심축, 육면체의 크기를 통해 A와 B 사이의 충돌 감지 유무를 확인한다.

우선 OBB B를 OBB A의 중심을 원점으로 하는 좌표축으로 변환한다.

$$B|_{objectA} = (T_A R_A)^{-1} T_B R_B B|_{objectB} \quad (식\ 1)$$

직선의 방향과 크기만 보존되면, 두 벡터의 내적값은 보존되므로, r_A와 r_B를 계산하는 데는 이동변환은 필요없고 object B좌표의 B의 좌표값에 대한 회전변환만으로도 충분하다. 그러므로, 회전변환 r_A, r_B 계산 시 사용되는 회전변환 행렬 R은

$$R = R_A^{-1} R_B \quad (식\ 2)$$

그리고 두 OBB를 분리축에 투영했을 때의 두 원점의 거리에 사용되는 T는 이동변환의 행렬이 아니라, 그림 2와 같이 두 중심사이의 벡터를 나타낸다. 이때, object A의 기준좌표축이 A_1, A_2, A_3이고, object B좌표계의 기준좌표축이 B_1, B_2, B_3 라 하고, OBB A의 크기가 각 좌표축 기준으로 $2a_1, 2a_2, 2a_2$, OBB B의 크기가 각 좌표축 기준으로 $2b_1, 2b_2, 2b_2$라 할 때,

식 3과 같은 수식이 만들어 지고,

$$r_A = \frac{|a_1 A_1 \bullet L|}{|L|} + \frac{|a_2 A_2 \bullet L|}{|L|} + \frac{|a_3 A_3 \bullet L|}{|L|}$$
$$r_B = \frac{|b_1 R B_1 \bullet L|}{|L|} + \frac{|b_2 R B_2 \bullet L|}{|L|} + \frac{|b_3 R B_3 \bullet L|}{|L|} \quad (식\ 3)$$

식 4와 같을 경우 두 OBB는 충돌한 상태이다.

$$\frac{|T \bullet L|}{|L|} < r_A + r_B \quad (식\ 4)$$

그림 3은 메쉬를 둘러싼 경계상자를 표현한 것으로 세밀한 충돌 감지를 구현한다.

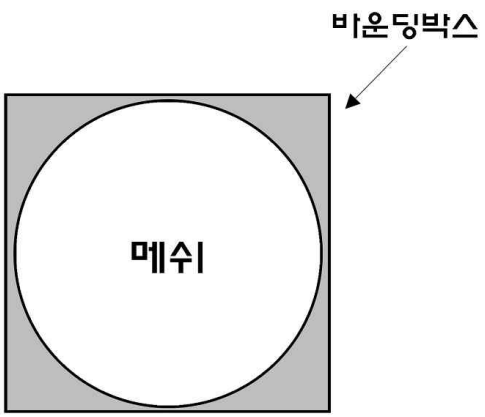

그림 3 바운딩 박스와 메쉬 관계
Fig 3. Relationship between Bounding box and mesh

2.2 유연한 카메라 클래스

유연한 카메라 클래스[3,4]는 기존의 카메라 클래스가 가지고 있던 x,y,z 중 한 축으로만 이동이 가능하다는 한계를 극복하고자 만들어진 클래스이다. 기존의 카메라 클래스는 x,y,z축 중 한 방향으로만 이동이 가능하였지만, 유연한 카메라 클래스에서는 x,y,z 각 축으로의 단위벡터를 생성하고, 단위벡터들의 회전을 통해, 자유로운 회전과 이동이 가능하도록 만들어졌다. 본 논문에서는 이러한 카메라 클래스와 마우스의 이동값을 통해 자유로운 회전과 이동이 가능한 유연한 카메라 클래스를 구현하였다.

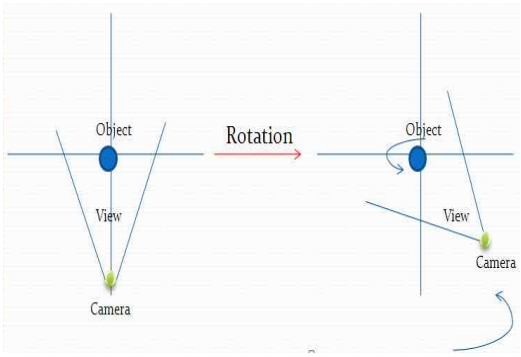

그림 4. 카메라와 오브젝트의 연동
Fig 4. Link between camera and object

그림 4는 유연한 카메라 클래스와 오브젝트 간의 연동을 그림으로 표현한 것이다. 카메라의 회전은 마우스 좌표값의 변화를 통해 조절하게 된다. 이전 마우스의 좌표값과 현재 마우스의 좌표값의 차를 통해 마우스가 이동한 거리를 구하고 이동한 거리를 각도로 환산하여 회전하게 된다.

이렇게 구한 각도만큼 오브젝트를 회전시키고, 회전한 각도와 오브젝트의 좌표, 카메라의 좌표를 통해 피타고라스의 정의를 이용하여 카메라가 회전하여 위치할 좌표를 구하게 된다. 공식으로 표현하면 다음과 같다.

식 5와 같은 방법을 통해 카메라를 오브젝트 와 함께 회전시킨다.

(ob:오브젝트, cmr:카메라, r:각도)

$$l = \sqrt{(ob.x - cmr.x)^2 + (ob.z - cmr.z)^2}$$
$$\text{cmr.x} = ob.x + l * \cos(r)$$
$$\text{cmr.z} = ob.z - l * \sin(r) \qquad \text{(식 5)}$$

2.3 폭발 이펙트와 입자 파티클

본 절은 폭발 이펙트와 입자 파티클 구현부분으로 파티클 시스템[5,6,7,8,9]을 사용하여 개발한다. 파티클 시스템은 신비로운 자연 현상의 많은 부분을 비슷하게 구현 가능하고 무수히 작은 입자들로 표현가능 하다. 파티클 입자들은 눈이 내리는 장면을 연출 하거나 폭발 이펙트와 같은 것들을 모델링 한다.

파티클 시스템에서 기본의 되는 것은 원하는 개수와 텍스처 파일을 초기화하는 것이다. 매 프레임 업데이트를 통해 새로운 좌표를 계산하는 방식으로 구현한다.

그림 5는 파티클 입자로 표현한 폭발을 구현한 그림으로 폭발과 같이 게임 상에서 특수한 효과를 구현하기 위해 점단위로 표현한다[10].

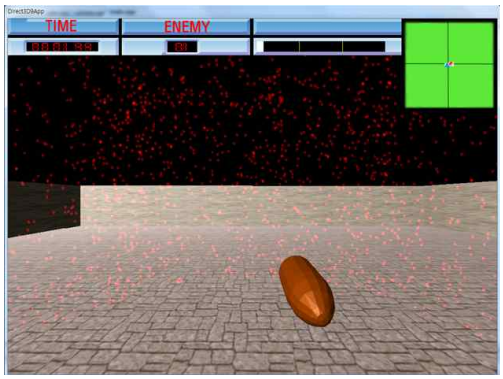

그림 5. 파티클을 이용한 폭발
Fig 5. Explosion using particle system

III. 개발환경

본 논문에서 구현된 개발환경은 윈도우 7 환경에 Visual Studio2008을 이용하였고, DirectX 9.0c라이브러리를 이용하여 구현하였다.

게임의 동작은 인텔 Quad Core 3.00Ghz와 주 메모리 4GB 사양의 PC에서 수행하였다.

그림 6은 게임을 플레이하는 화면으로써 자신을 향해 날라오는 미사일을 피하는 장면으로, 상단에 좌측부터 경과 시간, 날라오는 미사일의 수, 미니맵 등이 표시된다.

그림 6 미사일을 피하는 화면
Fig 6. Collision avoidance

그림 7은 피하지 못한 미사일과 충돌하여 폭발하는 장면을 보여준다.

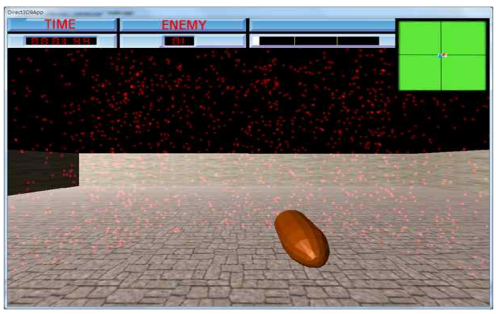

그림 7. 미사일과 충돌하여 폭발하는 화면
Fig 7. Collision effect

그림 8은 게임오버 시 보여지는 장면으로 자신의 기록과 최고 기록이 표시된다. 만약 자신의 기록이 최고기록을 넘어서게 되면 최고 기록은 자신의 기록으로 갱신되어 표시된다.

그림 8. 게임이 끝난 화면
Fig 8. Ending of Game

V. 결론

본 논문은 DirectX 라이브러리를 이용하여 1인칭 회피게임을 제작함으로써 기존의 2D 그래픽에 3인칭 시점의 회피게임과 다른, 3D 그래픽에 1인칭 시점의 회피게임을 제작 하였다.

본 논문은 게임엔진으로는 접근 할 수 없는 세세한 부분까지도 직접 구현 가능하다는 점과 충돌감지, 유연한 카메라 클래스, 파티클 시스템 등을 직접 구현함으로써 비주얼 이펙트를 극대화하고 그래픽을 최적화하여 메모리측면에서 효율적인 장점이 있다.

참고문헌

[1] David H. Ebrly, "3D game engine design", 2001.
[2] Christer Ericson, “Real-Time Collision Detection(The Morgan Kaufmann Series in Interactive 3-D Technology)”, 2005.

[3] Frank D. Luna," DirectX9를 이용한 3D GAME 프로그래밍 입문", pp.319-342, 2003.
[4] Frank D. Luna," Introduction to 3D Game Programming with DirectX 9.c A Shader Approach ", 2006.
[5] Frank D. Luna,"Introduction to 3D Game Programming with DirectX 10", 2008.
[6] Mark Deloura,"Game Programming Gems", Charles River Media, 2000.
[7] 김용준," 3D 게임 프로그래밍 ", 2010.
[8] Peter Walsh,"Advanced 3D Game Programming with DirectX 9.0", Wordware Publishing, 2003.
[9] Todd Barron저, 최현호 역, "DirectX9를 이용한 전략 게임 프로그래밍", 정보문화사, 2008.
[10] Andre Lamothe,"Tricks of the 3D Game Programming Gurus: Advanced 3D Graphics and Rasterization", Sams, 2003.

저자소개

안성옥(Sung-Og An)

1983 고려대학교 수학교육과(이학사)
1985 고려대학교 컴퓨터학과(이학석사)
1989 고려대학교 컴퓨터학과(이학박사)
1991년~현재 배재대학교 게임공학과 교수
※ 관심분야: 가상현실, 데이터베이스

이용운(YongUn Lee)

2011년 배재대학교 게임공학과

김수균(SooKyun Kim)

2006년 고려대학교 컴퓨터학과(이학박사)
2006년~2008년 삼성전자 통신연구소
책임 연구원
2008년~현재 배재대학교 게임공학과 조교수
※ 관심분야: 기하모델링, 게임그래픽, 실감미디어