3차원 종이 모델 게임의 설계 및 구현

이용운*, 이근호**, 김수균*

요약

아케이드 게임은 1970년대 후반에서 1990년대 초반 사이에 많은 인기를 누렸던 게임이며, 현재까지도 많은 관심을 받고 있는 게임 장르이다. 아케이드 게임은 작동하기 쉬우면서도 아기자기한 그래픽을 사용하여 볼거리를 제공하는 특징이 있다. 이러한 특성을 살려 본 논문에서는 3차원 종이 모델 게임을 설계하고 구현하는데 목적이 있다. 특히 게임엔진을 사용하지 않고 전 과정을 저수준의 DirectX 라이브러리만을 이용한 방법으로 제작하여 다양한 PC에서도 사용 할 수 있는 장점과 메모리 측면에서 효율적인 장점이 있다. 또한 본 논문에서 설계 및 구현한 게임은 30, 40대의 중장년들도 쉽게 즐길 수 있는 게임 구현을 목표로 한다.

Design and Development of 3D Paper Model Game

Yong-Un Lee*, Keun-Ho Lee** and SooKyun Kim*

ABSTRACT

The golden age of arcade games was from the late 1970s to the late 1980s, arcade games were still relatively popular during the first half of the 1990s, but the genre maintain in popularity. Arcade games are easy to handle, simple and intuitive control schemes, and rapidly increasing difficulty. This paper propose a 3D paper model game using DirectX library instead of game engines. Memory efficiency and use of various PC is the advantage of this game. Middle age like to comfortably this game.

Keywords : Collision detection, particle system, battle city, online, arcade game

* 배재대학교 게임공학과(✉mp3112@msn.com)
· 제1저자(First Author) : 이용운 · 교신저자(Correspondent Author) : 김수균
· 접수일(2011년 6월 10일), 수정일(1차 : 2011년 7월 8일), 게재확정일(2011년 7월 11일)

Ⅰ. 서 론

아케이드 게임은, 게임이라는 것이 생기면서부터 존재해온 매우 오래된 게임으로 쉬운 조작성과 강한 중독성을 가지고 있는 게임이다. 특히 순발력과 민첩성이 요구되는 전자 게임의 하나이고, 우리나라에서는 보통 전자오락실이라고 불리는 곳에서 흔히 접할 수 있다. 특히 우리나라에서는 아케이드 게임은 온라인 게임에 비해서 큰 인기를 끌고 있지는 못하다 [1]. 그러나 꾸준히 남녀 차이 없이 인기를 끌고 있는 분야이다.

본 논문에서는 이러한 아케이드 게임을 종이질감의 3D로 제작하여 다른 아케이드 게임과는 차별된 게임을 선보인다. 본 논문에서는 게임엔진에서 제공하는 그래픽의 특정화된 기술을 사용하지 않고 DirectX 라이브러리를 이용하여 게임의 세세한 부분까지도 직접 구현하는 것을 특징으로 하고 있다.

Ⅱ. 게임의 기초 세계 구축

2.1 하늘 상자를 이용한 하늘표현

3차원에서 원경을 구현 할 때 주로 쓰이는 것이 하늘 상자이다[2]. 실외형 게임구현에서 가상의 공간표현 할 때 중요한 부분이 하늘 표현이다. 하늘을 표현하는 것이다. 하늘 표현은 정육면체의 상자로 모델링된 모델에 텍스처를 입혀 구현한다.

2.2 뼈대를 이용한 애니메이션

게임 캐릭터 제작 디자이너들이 만들어낸 게임 캐릭터에 생명을 불어 넣기 위해서는 그 캐릭터가 다양한 행동들을 보이도록 애니메이션 시켜야 한다.

캐릭터 애니메이션[3, 4, 5]은 게임에서 가장 비중있는 분야 중에 하나로써 하드웨어의 비약적인 성장으로 인해 자연스러운 캐릭터 표현이 점차 증대되고 있다. 본 절에서는 캐릭터 애니메이션 처리 기법에 대해 설명할 것이다.

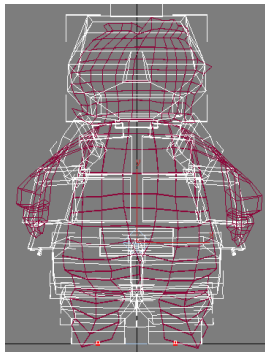


그림 1. 계층구조 형태로 서로 연결된 뼈대 제작
Fig 1. Hierarchy structure of game character

그림 1은 계층구조 형태를 가지는 서로 연결된 뼈대 셋으로 뼈대를 이동시키거나 회전시키면 메시의 표면도 이동되거나 회전된다. 메시 표면은 사람의 피부처럼 보이는데 메시 표면의 버텍스들 각각이 여러 개의 뼈대와 연결하여 구현한다. 본 모델은 3D MAX를 이용하여 손쉽게 제작 할 수 있다.

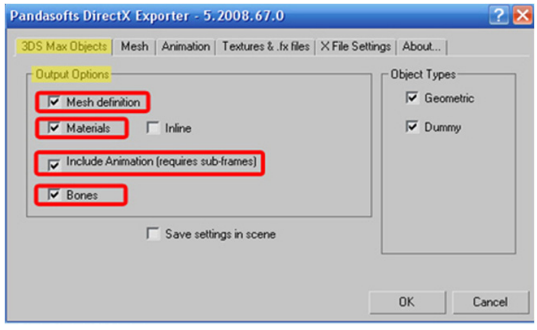


그림 2. 3DS Max 물체 옵션 설정
Fig 2. 3DS Max option setting for DirectX Exporter

이렇게 만들어진 게임 캐릭터는 Panda Export plug-in[6]을 통해 X파일로 추출한다. 추출하기 위해서는 3D Max에서 몇 가지 옵션 설정을 해줄 필요가 있다. 옵션 설정에 올바르게 되지 않으면, DirectX에서

캐릭터가 뒤틀려 보일 수 있다.

그림 2는 Export 프로그램 상의 옵션 값이다. 이러한 옵션에 의해 모델링한 캐릭터를 DirectX에서 사용 가능한 X파일로 변환 시킨다.

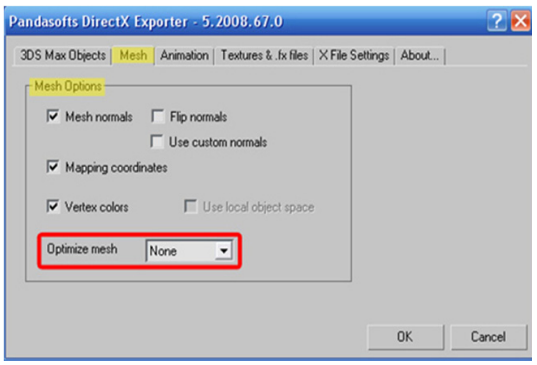


그림 3. 3DS Max에서의 메쉬 설정
Fig 3. 3DS Max option setting for Mesh

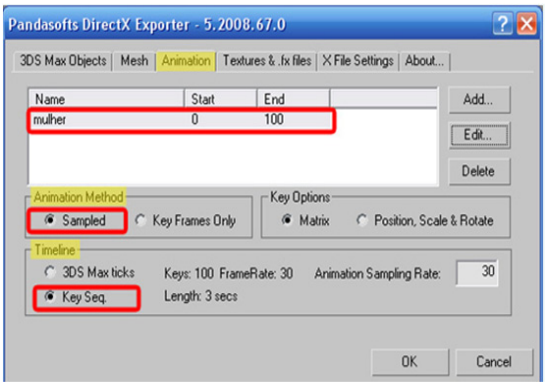


그림 4. 3DS Max에서의 애니메이션 옵션 설정
Fig 4. 3DS Max option setting for animation

그림 3은 게임 캐릭터 메쉬를 설정하는 부분이다. 그림 4는 애니메이션에 필요한 설정을 하는 부분으로, 특히 이부분은 애니메이션은 제작 애니메이션 별로 설정이 달라지기 때문에 설정을 할 때 옵션을 잘 선택해야 한다. 그림 5는 게임 캐릭터에 덮어씌울 텍스처 파일을 설정하는 부분으로 여러 가지 이미지를 입력받아 매핑 시킬 수 있도록 한다.

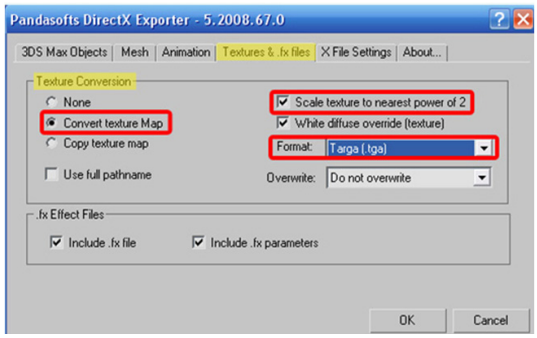


그림 5. 3DS Max에서의 텍스쳐 파일 설정
Fig 5. 3DS Max option setting for texture file

그림 6은 3DS MAX에서 제작된 모델을 x파일로 변환하기 위한 마지막 설정 단계이다.

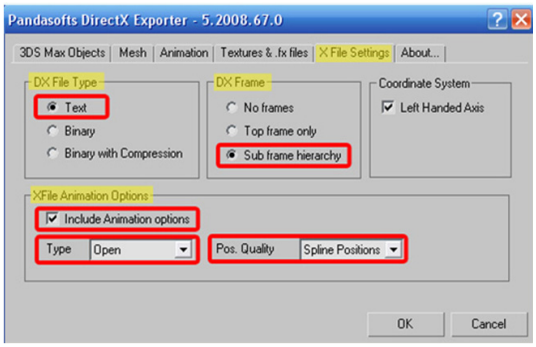
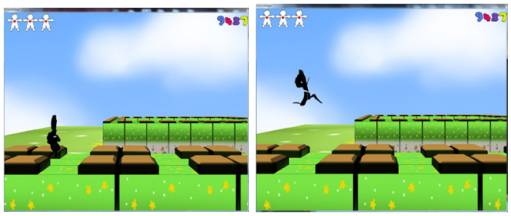


그림 6. 3DS Max에서의 x파일 설정
Fig 6. 3DS Max option setting for X file



(a) 걷기 애니메이션 (b) 뛰기 애니메이션
그림 7. x파일을 이용한 애니메이션
Fig 7. Animation using x file in DirectX

그림 7은 제작된 캐릭터 모델을 DirectX 환경에서 구현하여 애니메이션을 한 결과를 보여준다. 그림 7(a)

는 사용자가 걷기 동작을 시켰을 때를 보여주며, 그림 7(b)는 뛰는 동작을 보여준다.

III. 3차원 종이 모델 게임 구현

3.1 충돌 감지 구현

충돌 감지[2,3,7]는 두 물체가 서로 맞부딪치거나 움직이는 물체를 접촉할 때, 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.

특히 충돌 감지 할 때의 가장 좋은 전략 중의 하나는 계통적으로 범위를 좁혀 나가면서 충돌을 검출하는 것이다. 즉 객체의 경계구와 객체의 경계구를 비교하고, 다각형 경계구와 다각형 경계구를 비교하고, 마지막으로 삼각형 단위로 비교하는 것이다.

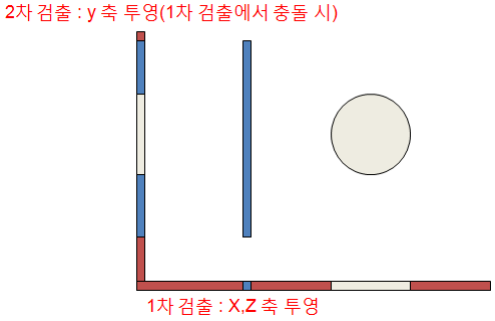


그림 8. 삼각형 평면화를 이용한 충돌 검출
Fig 8. Collision detection

이러한 충돌 감지 방법은 경계 상자와 경계 구를 이용하는 알고리즘이 가장 많이 사용되고 있다. 계산식도 간단하고 충돌효과도 만족할 만하다. 그러나 본 논문에서는 위의 충돌 판정식을 사용하지 않고, 조금 더 빠른 계산과 간단한 게임 제작을 위해 삼각형 평면화[2]를 이용한 충돌 검출을 사용한다. 이 방법은 하나의 삼각형을 삼각형의 방향에 기반해서 좌표 평면들 중 하나에 투영한다고 해보자. y축에 대해 평편화 시킨다면, y좌표는 사라지고 x와 z 좌표들만 남을 것이다. 이를 통해 평면화 된 삼각형에 대해 점이 삼각형 안에 있는지를 판단하여 충돌 여부를 결정하게 된다.

그림 8에서와 같이, 캐릭터와 장애물을 x와 z축으로 투영 시킨 후, 거리를 비교하여 투영축이 겹쳐져 있을 경우, y축에 대한 투영을 통해 최종적으로 캐릭터와 장애물이 충돌하였는지 아닌지를 판단한다.

```
bool cannon::ck_bound(D3DXVECTOR3 pos)
{
    ck_len = sqrt(pow(pos.x-bt_pos.x,2) + pow(pos.z-bt_pos.z,2));

    if(ck_len <= _ridus)
    {
        ck_len = pos.y - bt_pos.y;
        if(ck_len <= 5.0f && ck_len >=-8.0f)
            return true;
    }

    return false;
}
```

그림 9. 충돌 검출 알고리즘
Fig 9. Collision detection algorithm

그림 9는 본 논문에서 구현한 충돌 검출 함수를 나타낸다. 변수 ck_len은 캐릭터와 물체의 x와 z 좌표간의 거리를 측정하는 것이다. 만약 거리가 물체의 반지름보다 작을 경우 다음과 같은 기술을 사용하게 된다.

즉 캐릭터와 물체의 y좌표간의 거리를 비교(이때, 캐릭터의 y좌표는 발밑)하게 되고, y좌표간의 거리가 일정 거리(-8∽5 사이)내 일 경우 충돌한 것으로 간주하게 된다.

3.2 포탄 이벤트 효과

본 논문에서는 포탄이 날아가고, 주인공이 포탄과 충돌하고, 다른 한편으로는 이를 피하는 이벤트에 대해 설명한다. 주인공이 포탄과 충돌하는 것은 위에 설

명한 충돌 검출 알고리즘으로 해결 할 수 있다.

그러나 주인공이 포탄을 피하기 위해 점프하는 동작이 필요한데, 이를 사인(sin) 그래프를 이용하여 처리한다. 그림 10에서와 같이, 점프가 참일 때, 변수 temp의 값을 0.003씩 증가하고 sin(temp)의 값에 100을 곱한 후, 정수형으로 변경한다.

```
c_motion = true;
temp += 0.003f;
float temp2 = sin(temp)*100.0f;
int temp3 = (int)temp2;
jump = temp3*0.01f*10.0f;

if(temp >=3.14f)
{
     jump_mt = false;
     c_motion = false;
     c_move = CHAR_NO_MOVE;
     temp = 0.0f;

     jumpsound = true;
}
```

그림 10. 날라오는 포탄을 피해 점프하는 알고리즘
Fig 10. Jumping algorithm

정수형으로 변경한 sin(temp)의 값에 다시 0.01을 곱하여 소수점 둘째자리 실수로 변경하고, 변경된 값에 임의의 값(여기서는 10)을 곱하여 점프시의 높이를 조절한다. 변수 temp의 값이 1.57(180°)보다 커질 때까지 위의 내용 반복하면 점프가 가능하다.

그림 11은 실제 구현된 게임에서 포탄을 피하는 장면을 보여준다.

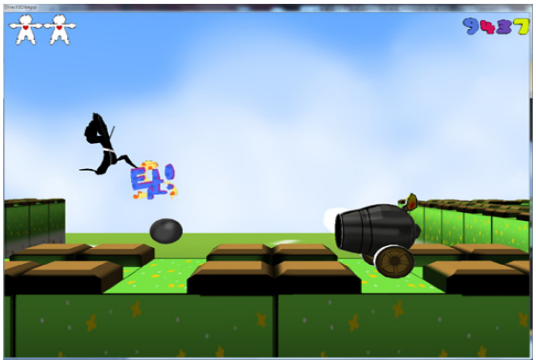


그림 11. 날라오는 포탄을 피하는 장면
Fig 11. Escape scene from canonball

3.3 입자 파티클

폭발 효과는 게임에서 현실감을 증대하기 위해 많이 쓰이고 있다. 이러한 폭발 효과를 나타내기 위해 파티클 시스템을 사용하는데, 여기서 파티클은 작은 입자 알갱이의 기본 원리를 이용하여 현란한 특수 효과를 표현하는데 많이 사용한다. 본 절은 입자 파티클 구현부분으로 파티클 시스템 [8, 9]을 사용하여 개발한다.

파티클 시스템은 신비로운 자연 현상의 많은 부분을 비슷하게 구현 가능하고 무수히 작은 입자들로 표현이 가능하다. 파티클 입자들은 눈이 내리는 장면을 연출 하거나 기관총에서 발사되는 총알과 같은 것들을 모델링한다. 파티클 시스템에서 기본의 되는 것은 원하는 개수와 텍스처 파일을 초기화하는 것이다. 매 프레임 업데이트를 통해 새로운 좌표를 계산하는 방식으로 구현한다.

그림 12는 파티클 입자로 표현한 연기를 나타낸다. 그림에서와 같이 게임 상에서 특수한 효과를 구현하기 위해 점단위로 표현한다.

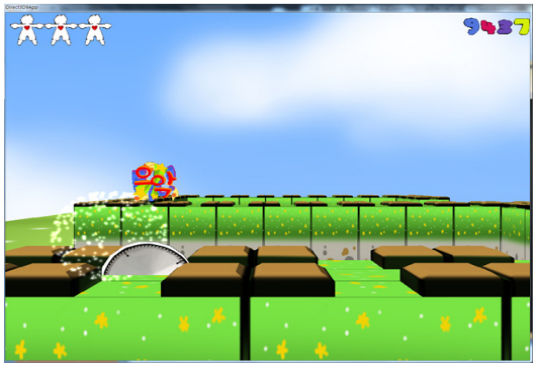


그림 12. 파티클 입자로 표현한 연기
Fig 12. Implemetation of Smoke using particle system

IV. 개발환경

본 논문에서 구현된 개발환경은 Quad Core

3.00Gh, 4Gb의 PC에서 윈도우 7 환경에서 Visual Studio2008을 이용하여 C++와 DirectX9.0 라이브러리를 사용하여 구현한다.

그림 13과14는 게임 플레이 화면으로서 맵에 설치된 장애물을 피하면서 전진하는 장면이다. 특히 그림 14는 맵의 코너에 닿았을 때, 카메라의 시점이 회전하면서 변화하는 장면이다.

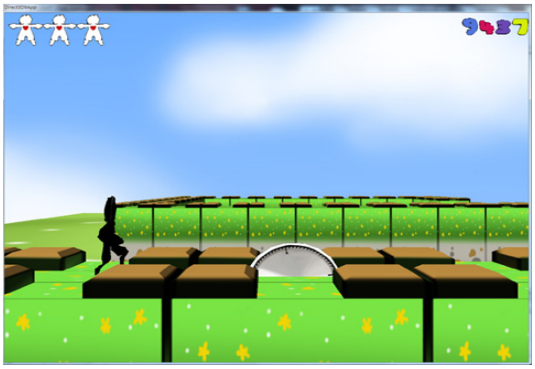


그림13. 함정을 통과하는 장면
Fig 13. Trap scene

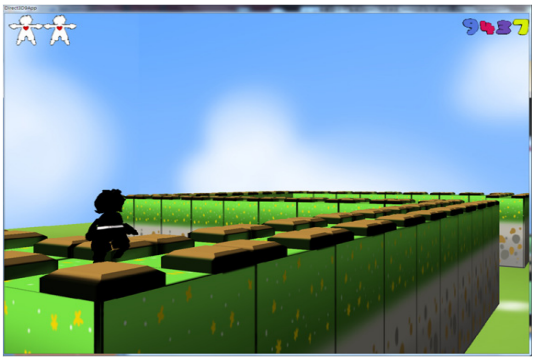


그림14. 코너에서 카메라 시점이 바뀌는 장면
Fig 14. Camera view transform from corner

V. 결론

본 논문은 DirectX 라이브러리를 이용하여 3차원 종이 모델 게임을 구현 하였다. 본 논문은 게임엔진으로는 접근 할 수 없는 세세한 부분까지도 구현 가능하다는 점과 충돌감지, 캐릭터 애니메이션, 파티클 시스템 등을 직접 구현함으로써 비주얼 이펙트를 극대화하고 그래픽을 최적화하여 메모리측면에서 효율적인 장점을 가지고 있으며, 다양한 연령층에서 즐길 수 있도록 만든 게임이다.

참고문헌

[1] *대한민국 게임백서, 한국게임산업진흥원, 2009*

[2] Mark Deloura, "*Game Programming Gems*", *Charles* River Media, 2000

[3] 김용준, "*3D 게임 프로그래밍*", 2010

[4] Mark Deloura, "*Game Programming Gems 2*", *Charles* River Media, 2001

[5] Dante Treglia, "*GAME Programming Gems 3*", Charles River Media, 2002

[6]http://www.andytather.co.uk/Panda/directxmax.aspx

[7] Andrew Kirmse, "*GAME Programming Gems 4*", Charles River Media, 2004

[8] Frank D. Luna, "*Introduction to 3D Game Programming with DirectX 9.c* A Shader Approach ", 2006

[9] Frank D. Luna, "*Introduction to 3D Game Programming with DirectX 10* ", 2008

저자소개

이용운(YongUn Lee)

2011년 배재대학교 게임공학과

이근호(Keun-Ho Lee)

2006년 고려대학교 컴퓨터학과(이학박사)
2006년~2010년 (주)삼성전자 DMC연구소
2010년~현재 백석대학교 쿰인성개발원 팀장

2010년~현재 백석대학교 정보통신학부 전임강사
※ 관심분야 : M2M 보안, 이동통신보안, 융합 보안, 개인정보보호

김수균(SooKyun Kim)

2006년 고려대학교 컴퓨터학과(이학박사)
2006년~2008년 삼성전자 통신연구소
책임 연구원

2008년~현재 배재대학교 게임공학과 조교수
※ 관심분야: 기하모델링, 게임그래픽, 실감미디어