3차원 온라인 아케이드 탱크 게임 설계 및 구현

김수균*, 이찬섭**, 안성옥***

요약

현재는 다양한 도구들을 이용한 게임들이 구현되고 있다. 본 논문에서는 '배틀시티'라는 고전게임을 주제로 하여 기존 게임에서 보여 주었던 2차원 요소를 3차원으로 확장하여 고기능의 아케이드 게임 구현을 제안한다. 아케이드 게임은 동전으로 작동하는 오락 게임으로 공공장소에 비치된 비디오 게임 종류를 말하며, 본 논문은 게임 엔진을 사용하지 않고 DirectX 라이브러리를 이용하여 비용을 절감하고 유연한 설계에 대한 높은 확장성을 제공한다. 특히 본 논문은 다양한 연령층의 사용자들이 쉽게 즐길 수 있는 게임 구현을 목표로 한다.

Design and Development of 3D Online Arcade Tank Game

Soo-Kyun Kim*, Chan-Seob Lee**, Syung-Og An***

ABSTRACT

Many games are developed by use various toolkit. This paper proposes a development of battle city, which is a famous classic arcade game. An arcade game is a coin-operated entertainment machine, usually installed in public businesses such as public houses, and video arcades. Most arcade games are redemption games, merchandisers (such as claw crane), video games, or pinball machines. However, the objective of this paper is to produce a tank game using directX library instead of the specialized engines. The advantage is that the overall cost of the production would be minimized since there is no need to invest in making game engine. Users can enjoy to easily the proposed game.

Key Words : Collision detection, particle system, battle city, online, arcade game

* 배재대학교 게임공학과(✉kimsk@pcu.ac.kr)
** 혜천대학 물류유통정보과
*** 배재대학교 게임공학과
· 제1저자(First Author) : 김수균 · 교신저자(Correspondent Author) : 안성옥
· 접수일(2010년 12월 27일), 수정일(1차 : 2011년 1월 26일), 게재확정일(2011년 1월 31일)

Ⅰ. 서 론

오늘날 많은 게임들은 온라인 기반 롤플레잉 게임 형식으로 구현 되고 있으며, 유사한 종류의 게임들로 넘쳐 나고 있다. 이러한 고기능 게임들이 출시되고 있지만, 고전 게임을 주제로 한 게임들은 아직도 사용자에게 상당한 영향력을 행사하고 있다. 특히 고전게임을 즐겨했던 20~30대들에게 현대적이면서도 고전적인 요소를 갖춘 게임을 제공한다면 사용자들에게 더욱더 친숙하게 접근 할 수 있다. 특히 청소년들에게는 새로운 장르의 게임, 일반 성인들에게는 추억을 회상하는 게임이 될 수 있다.

기존 아케이드 게임은 주로 높은 비용의 3차원 엔진을 사용하여 구현 되었지만, 본 논문에서는 엔진을 사용하지 않고 DirectX 라이브러리[1, 2, 3] 를 기반으로 하여 제작 비용을 절감하도록 하였다. 또한 고품질의 3차원 컨텐츠 서비스를 위한 인터페이스 솔루션 기술을 제공하고 유연할 설계를 통해 다양한 플랫폼에 적용 할 수 있도록 한다.

본 논문에서는 DirectX 라이브러리를 이용하여 기존에 출시된 고전게임의 2차원적 요소를 컴퓨터 그래픽스 기법을 이용하여 3차원으로 확장하여 설계 및 구현하였다.

Ⅱ. 게임 시스템 설계

2.1 아케이드 게임 시나리오 설계

본 절에서는 사용 사례와 액터 등이 포함된 사용 사례 다이어그램[4, 5]을 통해 시스템 설계에 대해 기술한다. 사용 사례 다이어그램에서는 사용 이름, 관련되는 액터, 전제 조건, 성공 또는 실패 결과 조건, 사건의 흐름, 확장 등이 표현되어야 한다. 여기서는 모든 실패 케이스를 기술할 필요는 없겠으나 예외 사건의 흐름은 표현 할 필요가 있다.

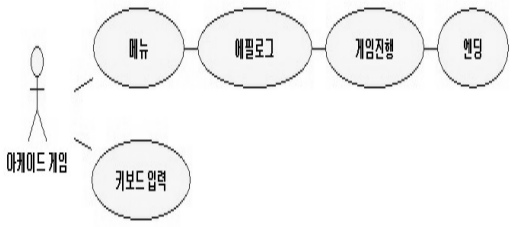

그림 1. 사용 사례 다이어그램
Fig 1. Use Cases Diagram

그림 1은 사용 사례 다이아그램을 나타내며, 사용 사례 다이어그램에서의 액터는 아케이드 게임으로 표현된다. 즉 사용자를 의미하고, 이에 대한 자세한 설명은 표 1의 사용 사례 시나리오를 통해 설명한다.

표 1. 사용 사례 시나리오
Table 1. Use Cases Scenario

사례 이름	메뉴
참여 액터	아케이드게임
시작 조건	1. 아케이드게임이 시작되었을 경우
사건 흐름	2. 게임시작, 도움말, 게임종료의 2D이미지 메뉴가 표시 3. 키보드 좌우 방향 키를 이용하여 메뉴 선택 4. 도움말 선택 시 키보드조작법에 이미지 출력, 2번으로 되돌아감 5. 게임시작 선택 시 에필로그로 이동 6. 게임종료 선택 시 아케이드게임 종료
종료 조건	7. 메뉴에서 도움말을 제외한 항목 중 한 가지가 선택 되었을 경우 종료
사례 이름	키보드 입력
참여 액터	아케이드게임
시작 조건	1. 아케이드게임이 시작되었을 경우
사건 흐름	2. 키보드 입력에 대한 이벤트를 실행 3. 상하좌우 방향 키 : 상하좌우 이동 4. 스페이스 키 : 미사일 발사 5. 엔터 키 : 이벤트 선택

종료 조건	6. 이스케이프 키 : 메뉴 화면으로 이동 7. 아케이드게임이 종료 되었을 경우
사례 이름	에필로그
참여 액터	아케이드게임
시작 조건	1. 메뉴에서 게임시작이 선택 되었을 경우
사건 흐름	2. 주인공이 등장, 카메라 방향으로 이동 3. 주인공이 게임에 대한 줄거리를 설명 후 오른쪽 방향으로 이동 4. 악당들의 등장하고 대사
종료 조건	5. 악당들이 대사를 마치면 종료
사례 이름	게임진행
참여 액터	아케이드게임
시작 조건	1. 에필로그가 종료 되었을 경우
사건 흐름	2. 첫번째 맵이 생성되고, 게임 스타트 표시 3. 키보드 입력을 통해 게임을 진행 4. 악당들은 인공지능을 가지고 예측할 수 없는 방향과 이동을 가짐 5. 3분 이내에 주인공은 악당들의 방해를 피하고 열쇠를 획득하여 출구로 탈출해야 함 6. 미사일을 사용하여 악당들의 방해를 저지할 수 있으나 미사일 사용가능 횟수는 20발로 제한 되어 있음 7. 악당들과 충돌 하였을 경우 주어진 생명 감소 8. 4단계까지 반복
종료 조건	9. 주인공이 네번째 단계에서 보스를 물리치고, 열쇠를 획득하여 출구로 탈출하였을 경우 10. 정해진 시간이 지났을 경우 게임오버 표시 후 종료 11. 주어진 생명을 모두 잃었을 경우 게임오버 표시 후 종료 12. 이스케이프 키를 입력 했을 경우

2.2 기능 리스트

본 절은 게임에 대한 베타 및 최종 배포 판에 포함될 기능을 설명한다. 각각의 기능은 시스템이 제공하여야 할 구현의 단위가 된다. 구현하는 기능이 배포판에 포함될 수도 있고, 포함되지 않을 수도 있는데 포함될 기능은 해당 배포판을 포함하고, 그렇지 않은 기능은 “stretch”로 표시한다.

표 2의 기능 리스트는 정해진 기간 내에 제작할 수 있는지의 여부에 대한 설계를 나타낸다.

표 2. 기능 리스트
Table 2. Feature List

Feature	Target
3D 오브젝트 모델링 아군탱크	Final
2D User Interface 디자인	Final
2차원 배열에 의한 맵 생성	Final
인공지능을 이용한 악당 구현	Final
오브젝트 충돌 감지	Final
카메라 시점 변경	Final
이벤트 효과	Final

III. 게임 구현

3.1 2차원 배열을 이용한 맵 생성

맵은 2차원 배열[2]로 생성하며 초기화를 위한 임시 2차원 배열을 포함한다. 시나리오의 구성에 따라 4단계의 맵을 구성하기 위해서는 2차원 배열이 5개가 필요하고 각각의 크기는 11*12로 구성한다.

맵의 구성에 따라 벽돌로 표시되는 구역, 나무상자의 위치, 악당의 등장 위치, 시각장애물의 구역 등 미리 약속된 배열의 위치에 정수 값을 대입한다. 생성된 2차원 배열은 만들어진 Draw_StageMap()함수에서 호출하고, 각 인자 값으로 받는 값에 따라서 단계적으로 맵을 출력한다.

2차원 배열을 이용함으로써 단계적 맵 출력과, 앞으로 구현되어질 모든 물리적인 충돌 감지에 대해 쉽게 적용 가능 하다. 그림 2는 단계에 따른 맵 구성도이며 이와 같은 설계를 통하여 맵 구성의 단계를 늘릴 수 있는 장점이 있다.

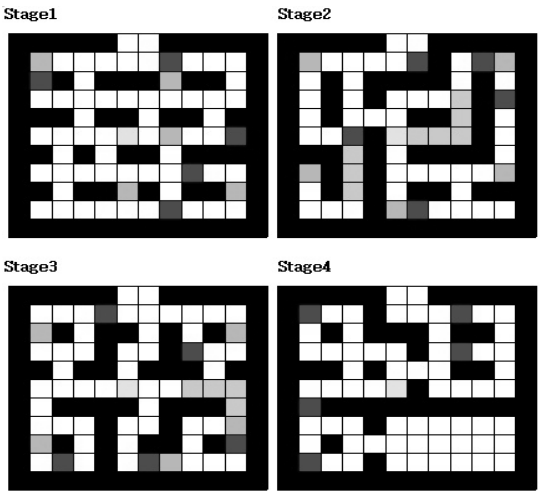

그림 2. 단계에 따른 맵 구성
Fig 2. Stage of Map

3.2 물체의 충돌 감지

3차원 공간에서 물체간의 충돌을 감지[6]하는 방법은 여러 가지가 있지만, 본 논문에서는 좌표범위에 대한 충돌 감지를 구현 한다. 물체의 월드좌표를 중심으로 사각형의 넓이 값과 높이 값을 측정하여 충돌을 감지하는 방식이다. 좌표범위를 이용하여 경계를 지정할 경우 3차원의 y축 감지에 있어서는 문제가 생기지만, 게임의 2차원 진행 방식 특성상 좌표범위 충돌 감지는 전혀 문제 될 것이 없고, 2차원 배열을 이용한 맵 생성 함수와 연동이 쉬운 편이다.

2차원 맵 배열에서 각 배열에 들어있는 값이 캐릭터가 지나갈 수 없는 장애물의 값일 경우 그 배열 대한 월드좌표를 중심으로 범위를 구하고, 현재 캐릭터의 월드좌표 범위와 충돌을 감지한다. 제안 방법에서는 모든 물체에 대해 범위 값을 계산하여 충돌을 감지한다. 그림 3은 월드좌표(x, z) 값을 이용하여 범위를 구하는 원리를 보여준다. 물체에 지정된 값을 더하여 범위를 구하게 되는데, 이는 Extent()함수에서 월드좌표 값을 인자로 받아 계산하고 범위를 구하여 전역변수에 저장시키는 구조로 되어 있다.

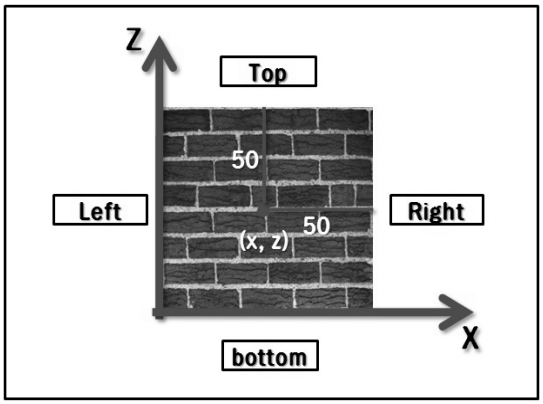

그림 3. 오브젝트의 범위 계산
Fig 3. Calculation extent of Object

3.3 인공지능을 이용한 적 구현

게임에서 가장 중요한 부분은 적을 구현하는 것이다. 적은 직접 조종되어지지 않고 일정 상황이 발생하였을 경우 자동으로 작동되어야 한다. 이를 위해 인공지능[7]을 이용하여 적이 자동으로 움직이고 충돌을 감지하는 알고리즘을 구현해야 한다. 적의 움직임에 대한 알고리즘에는 A*알고리즘을 포함한 다양한 방법들이 있으나, 본 논문에서는 복잡한 알고리즘이 필요하지 않으며, 적은 주어진 위치에서 주인공의 방해를 일으키는 요소로서의 역할만 수행하면 된다.

본 논문에서 구현된 인공지능은 적의 이동, 적의 충돌 감지, 적의 미사일 발사가 있다.

적의 이동은 Enemy_move()함수에서 각 적에 대한 Index값을 인자로 받아 이동 상태를 결정해준다. 적의 이동 상태는 상, 하, 좌, 우의 4가지 방향을 가지고 이동하는 상태가 있으며, Define으로 미리 선언 되어 있다. 오브젝트간의 충돌 감지[8, 9]가 일어났을 경우 Enemy_move()함수가 호출되어 상태를 변경해준다.

그림 4는 적이 이동 중에 이동 불가능한 물체의 경계를 감지하였을 경우 방향을 다시 재탐색 하는 장면을 보여주고 있다. 방향을 재탐색 하는 경우는 무작위로 방향을 탐색하여 예측할 수 없는 이동을 함으로써

사용자는 적의 이동에 긴장하게 된다. Extent_enemy() 함수에서는 적의 모든 충돌 감지를 수행하며 미사일에 맞았을 경우 및 주인공과의 충돌을 감지한다. 미사일이 또는 주인공과 충돌 하였을 경우 적은 DirectX 라이브러리가 제공하는SetRenderState()함수에서 알파값을 적용하여 투명한 상태로 적용하고 y축으로 회전하며 수직 상승하게 된다. 이 경우는 주인공과 충돌을 감지하지 않으므로 주인공은 계속해서 이동이 가능하다.

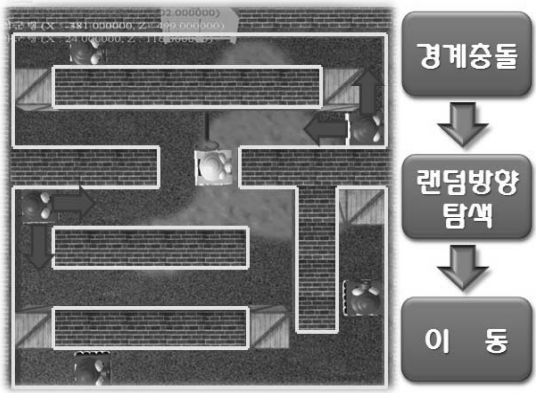

그림 4. 적 이동 원리
Fig 4. Enemy move principle

적의 미사일 발사는 마지막 4단계에 등장하는 적들 중 보스에게만 구현 되어 있다. 보스의 미사일 발사 방식은 총 2가지가 있고, 미사일이 가운데서 1발이 발사하는 것과 양쪽에서 2발이 발사하는 무작위 순서로 되어 있다. 이는 사용자가 적의 미사일을 예측하기 어렵게 하여 난이도를 높이기 위함이다. 보스의 미사일 방식은 Move_Kmissile()함수에서 호출하고 보스가 생명을 모두 잃었을 경우에 종료하게 된다.

3.4 카메라 시점 변경

본 절에서는 카메라의 시점 변경에 대해 설명한다. 본 논문에서의 카메라는 두 가지의 시점을 가지고 가상의 공간을 바라본다. 첫 번째 시점은 게임 진행화면을 바라보는 위쪽 시점(Top View)으로 마치 하늘에서 지상을 바라보는 느낌의 시점으로 대부분의 아케이드 게임의 카메라 시점으로 사용된다. 본 구현에서는 카메라의 월드 좌표를 정의 하여 일정 높이에 위치시키고, 주인공의 움직임에 따라 카메라가 상하좌우로 이동을 하면서 바라본다.

두 번째 시점은 1인칭 시점으로 카메라가 정면에서 주인공을 바라보는 시점이다. 대부분의 일인칭 슈팅 게임에서 사용하는 시점이지만, 본 구현에서는 처음 게임의 스토리 진행방식을 설명하기 위한 에필로그와, 엔딩 부분에 구현 되어있다. 두 번째 카메라의 월드좌표를 정의하여 에필로그의 흐름에 따라 주인공을 바라보고 움직한다. 본 논문에서는 카메라의 시점 변경 시에는 getPosition()함수에서 카메라의 위치 값을 불러오고 setPosition()함수에서 값을 위치시킨다.

3.5 이벤트 효과

본 논문에서는 파티클 및 조명 값을 이용하여 이벤트 효과를 만든다. 미사일이 날아가고 탈출구의 위치를 표시하는 효과에는 파티클을 적용하고, 주인공이 적과 충돌 시 발생하는 효과는 조명 값을 변경하여 반짝이는 효과를 적용 한다. DirectX 라이브러리에서 제공하는 CParticleSystem()함수를 이용하여 인자 값을 변경하고 색깔 정보를 수정 하여 미사일이 생성되어 날아가는 장면에 적용 한다.

미사일의 회전과 동시에 파티클도 회전하며 좀 더 사실적인 장면을 표현 하게 된다. 주인공이 탈출해야만 단계가 종료가 되는 위치에도 이와 같은 함수를 이용하여 물체가 생성되는 동시에 파티클을 적용하고 탈출위치를 표시한다.

다음은 조명 값[10, 11]을 변경한 충돌 효과이다. DirectX 라이브러리에서 제공하는 InitPointLight()함수에서 R, G, B 3가지 색깔 값을 인자로 받아 조명의

색깔을 변경 한다. 충돌 시 함수의 인자 값으로 Red값을 증가 하고 일정 시간이 지나면 다시 Red값을 감소하는 과정을 반복하여 마치 카메라에 충격을 주는듯한 효과를 가져 온다.

지금까지 설명된 모든 효과는 Display()함수에서 호출하여 시작하고 종료하는 구조로 되어 있다. 그림 5은 파티클 효과를 나타내며, 그림 6은 최종 게임 실행 화면을 보여준다.

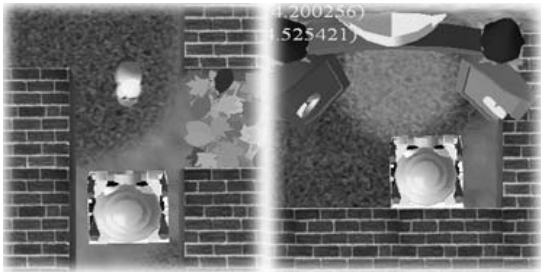

그림 5. 파티클 효과
Fig 5. Particle Effect

그림 6. 게임 실행 화면
Fig 6. Game Play Scene

IV. 개발환경

인텔 듀얼코어 프로세서 환경의 마이크로소프트 윈도우7 운영체제에서 제작하고, 마이크로소프트 Visual Studio Team System2008 SP1에서 DirectX SDK9.0c 라이브러리 기반의 언어를 사용한다. 3차원 환경 구현을 위한 그래픽 디자인 작업은 Autode나 Max2009, XSI Softimages6.5, Adobe Photoshop CS3, Adobe Illustrator CS3를 이용하여 제작한다.

V. 결론

본 논문에서는 '배틀시티'라는 고전게임을 주제로 하여 기존 게임에서 보여 주었던 2차원 요소를 3차원으로 확장하여 고기능의 아케이드 게임을 구현 하였다. 특히 게임 엔진을 사용하지 않고 DirectX 라이브러리를 이용하여 비용을 절감하고 유연한 설계에 대한 높은 확장성을 제공하였고, 고전 게임의 시나라오를 이용하여 제작함으로써 사용자들에게 향수를 일으킬 수 있으며, 이를 통해 다양한 아케이드 게임을 3차원화 할 수 있는 연구 방향을 제시하였다.

참고문헌

[1] Frank D. Luna, " *Introduction to 3D Game Programming with DirectX 10* ", 2008
[2] Todd Barron저, 최현호 역, "*DirectX9를 이용한 전략 게임 프로그래밍*", 정보문화사, 2008
[3] 김용준, "*3D 게임 프로그래밍*", 한빛미디어, 2008
[4] 이노우에 타츠키, "*다이어그램으로 쉽게 배우는 UML*", 한빛미디어, 2008
[5] Martin Fowler and Kendall Scott, "*UML Distilled: A Brief Guide to the Standard Object Modeling Language (2nd Edition)* ", Booch Jacobson Tumbaugh
[6] Kelly Dempski, "*DirectX 실시간 렌더링 실전 테크닉*", 정보문화사, 2003
[7] Dante Treglia, "*GAME Programming Gems 3*", Charles River Media
[8] Mark Deloura, " *Game Programming Gems*", Charles River Media

[9] Eric Lengyel, "*Mathematics for 3D Game Programming & Computer Graphics*", 2008
[10] Mark Deloura, " *Game Programming Gems* 2", Charles River Media
[11] Kim Pallister, " *Game Programming Gems* 5", Charles River Media

저자소개

김수균(SooKyun Kim)

2006년 고려대학교 컴퓨터학과(이학박사)
2006.3~2008.2 삼성전자 통신연구소 책임 연구원
2008년~현재 배재대학교 게임공학과 조교수
※ 관심분야: 기하모델링, 게임그래픽, 실감미디어

이찬섭(Chan-Seob Lee)

2000년 한남대학교 컴퓨터공학과 (공학석사)
2003년 한남대학교 컴퓨터공학과 (공학박사)
2003년~현재 혜천대학 물류유통정보과 조교수
※ 관심분야 : 웹 DB, 유비쿼터스, 캐싱

안성옥(Sung-Og An)

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