3차원 레이싱 게임의 컴퓨팅 환경과 개발 방법

장현덕* · 김성훈**

A Game Computing Environment and a Guideline for Developing 3-Dimensional Racing Games

Hyun Duk Jang* · Seong Hoon Kim**

요 약

본 논문은 3차원 레이싱게임을 위한 최근 변화된 게임 개발환경과 그에 따라 달라진 3차원 컴퓨터게임 개발 방법을 모색한다. PC 환경 하에서 게임의 속도구현을 위한 3차원 게임 라이브러리 DirectX의 사용과 윈도우즈 환경 하에서의 게임개발 방법을 고찰함으로써 사용자 게임실행시의 다양한 하드웨어 환경을 지원하고, 유효한 게임실행 속도를 얻을 수 있는 3차원 컴퓨터게임의 개발방법론을 제시하였다.

ABSTRACT

The purpose of this paper is to overview the recently changed environments of developing a computer game software and propose an appropriate developemnt methodology to these environments. For the speed of playing 3-dimensional games in window-based environments of personal computer, we investigated DirectX as a game development library and suggested a guideline for developing 3-dimensional computer games which is compatible with various hardware platforms and keep up a fitting speed of game-playing.

키워드

game development environment, 3-dimensional racing game, DirectX 3D graphics library

Ⅰ. 서 론

최근 게임산업은 차세대 국가육성산업의 하나로 인식되어지고 있으며 이와 함께 게임 개발에 대한 관심은 최근 들어 과거 그 어느 때보다도 높아지고 있다. 또, 높아가고 있는 게임 개발에 대한 관심 만큼이나 게임개발환경은 급속도로 변화하고 있다[1].

하드웨어의 급속한 발달과 더불어 특정 하드웨어에서 구현 가능했던 3차원 게임이 PC에서 구현되면서 컴퓨터게임 시장의 주류는 3차원 게임으로 돌아선 것으로 보이며, 이러한 컴퓨터게임시장의 변화로 게임개발자들은 이제 3차원게임 개발자로서의 능력을 확보해야만 하게 되었다. 일반적으로 3차원 게임에서는 2차원 게임에서 사용되었던 여러가지 개발기법들을 재사용하는 것이 거의 불가능할 정도로 다른 방법론을 채택하고 있다. 예를 들어 그래픽 제작의 경우, 3차원

* (주)모퍼스
** 영동대학교 컴퓨터공학과

접수일자 : 2005. 9. 1

게임은 스프라이트(Sprite)[1] 모델링 과정에서부터 오프라인으로 제작된 비트맵으로 이루어져 있는 기존의 2차원 게임과는 달리 실시간에 랜더링(Rendering)되어야 할 다각형(Polygon)기반의 솔리드 모델(Solid Model)들로 이루어져야 하는 것이다.[2] 3차원 컴퓨터게임의 개발은 2차원 컴퓨터게임 및 업무용 소프트웨어의 개발과는 달리 운영체제 위에서 실시간으로 렌더링(Rendering)되어야 하는 그래픽 이미지를 처리하여야 함과 동시에 게임프로그램 내에서 스프라이트(Sprite)들의 빠른 속도를 고려해야 하는 특수한 상황의 기반 위에서 이루어진다. 또, 다양한 하드웨어 환경에 구애받지 않고 정확한 동작제어를 구현할 수 있어야 하고 범용 소프트웨어와는 달리 개발환경과 사용자 게임 실행환경의 변화속도가 매우 빠르다는 점을 고려한다면 3차원 컴퓨터게임의 개발방법론은 일반적인 개발방법과는 다소 다른 각도에서의 접근이 필요하다.

본 논문의 구성은 다음과 같다. 2장에서는 3차원 컴퓨터게임 개발환경을 살펴보고 3장에서는 3차원 레이싱 게임의 구현과 이에 대한 논의를 전개한다. 4장에서는 논문의 결론으로 연구결과와 후속연구를 위한 제언을 한다.

Ⅱ. 3차원 컴퓨터게임 개발환경

2.1 3차원 컴퓨터게임의 개발개요

Greg Costikyan은 "게임이란 예술의 한 형태로, 플레이어라고 불리는 참가자가 목표달성을 위해서 게임토큰[2]을 통해 자원관리를 위한 의사결정을 하는 것이다"라고 정의하고 있다.[3] 또, Chris Crawford은 "게임이란 인간존재의 기본적인 한 부분이다"라고 말한다.[4] 게임이란 인간의 생활에 직간접적인 영향을 미치는 예술형태의 하나로 자리잡아가고 있는 것이다. 게임을 개발한다는 것은 이제 '무엇을'에서 '어떻게 하느냐'가 더 중요한 시점이다.

게임의 개발은 다양한 측면에서의 종합적인 고찰이 선행되어야 한다. 게임이론가인 Geoff Howland는 게임에 있어서 필수요소를 그래픽, 사운드, 인터페이스, 게임플레이, 스토리등 5가지로 정의하고 있는데,[5] 이렇게 각각의 속성이 다른 매체들을 하나의 기반을 갖는 게임개발에 대한 종합적인 고찰이 필요하다는 것이다.

3차원 컴퓨터게임은 게임 플레이어들의 위치와 시선이 자주 변하면서 그에 따른 빠른 3차원 배경(Scene) 정보를 요구한다는 점이 2차원 컴퓨터게임과는 전적으로 다르다. 이를 실제로 구현하기 위해서는 컴퓨터과학과 수학 그리고 기하학등의 많은 지식을 필요로 한다. 실제로 3차원 컴퓨터 게임 만큼 수학과 기하학을 요구하는 게임형태는 없다. 또한 3차원 컴퓨터게임은 3차원 스프라이트들의 실시간 렌더링과 상호작용에 의한 충돌검사(Collision Detection)에 큰 계산량을 요구하므로 보다 뛰어난 하드웨어를 필요로 한다. 컴퓨터환경에서 3차원 컴퓨터게임이 가능하게 된 것은 하드웨어의 급속한 발전 덕택이었다.

현재의 3차원 컴퓨터게임에서는 오브젝트와 배경 모두를 폴리곤(Polygon)을 사용하는 완전한 3차원 컴퓨터게임의 모습을 보여주고 있다. 특히 하드웨어의 발달, 즉 그래픽 가속기(Graphic Accelator)의 등장으로 3차원 컴퓨터게임은 다양한 시도와 기법이 개발되고 있다. 액션과 비행 시뮬레이션에서 출발한 3차원 컴퓨터게임은 현재 스포츠, 어드벤처, 롤플레잉, 레이싱등 다양한 장르의 3차원 컴퓨터게임이 출시되고 있다.

2.2 3차원 컴퓨터게임의 그래픽스

3차원 처리를 위해서 그림 1의 데카르트 좌표계(*Cartesian coordinate system*)라는 매우 일반적인 좌표계가 사용된다.

게임의 이미지인 객체는 점과 선, 면과 각으로 구성되어 있으며, 이중 가장 기본적인 단위는 점이다. 하나의 점을 변형시키면 선이나 면, 객체의 변형은 손쉽게 구현된다. 예를 들어 3차원의 정육면체인 객체를 변형시키려면 변형 전에 해당 객체를 면으로 나누고 다시 선으로 나눈 뒤 궁극적으로는 점으로 나누어 표현하는 알고리즘을 작성해야 한다. 객체의 크기변환과 위치변화등 움직임에 대해서는 스케일링 행렬을 이용하여 구현하고, 회전은 점(x,y,z)의 축을 기본으로 하여 일정한 각도만큼 회전시키는 행렬을 사용한다.

1) 게임에 등장하는 제어가능한 등장인물의 개념으로 CRT화면상에 정의가능한 픽셀 패턴

2) 직접 플레이어가 조작할 수 있는 어떤 것. 카드게임에서의 카드, RPG에서의 인물, 스포츠 게임에서의 플레이어 자신을 뜻한다.

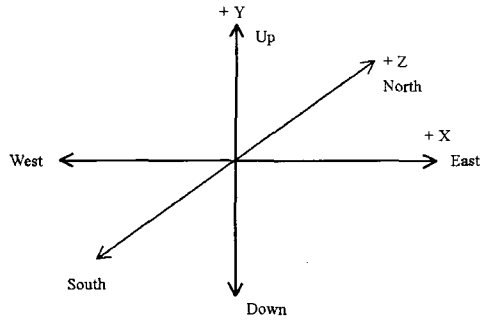

그림 1. 데카르트 좌표계
Fig. 1 Cartesian Coordinate System

객체를 입체구조를 갖는 것처럼 보이게 하기 위한 모델링은 많은 계산량이 필요하고 하드웨어나 소프트웨어 엔지니어들에게 매우 중요한 기술로 여겨지고 있다. 모델링에는 입체 모델링과 외곽선 모델링등이 있으며 데이터 변형기법, 투영기법을 사용하기도 한다.

텍스처링은 구현하고자 하는 객체의 평평한 부분이나 구부러진 표면에 비트맵 이미지를 매핑(Mapping)하는 과정을 의미하며, 이 작업을 거치면 이미지는 상당한 현실감을 가지게 된다. 또한 광선추적 기법은 3차원 객체의 완벽한 모델링을 이용하여 컴퓨터상에서 사진과 같이 실감나는 이미지를 생성하는 방법이다. 이 기법은 단일 알고리즘으로서 광선추적을 통해 감추어진 면의 제거, 투명처리, 반사처리, 굴절처리, 전체조명, 단일조명, 그림자 처리등의 효과를 얻을 수 있는 최신의 텍스처링 기법이다. 3차원 게임에서 현실감나는 이미지를 구현하는 기법중의 하나이다.

2.3 3차원 컴퓨터게임의 운영체제

게임개발에 있어서 윈도우즈와 도스 환경은 여러가지 차이점 및 장단점이 있다. 따라서 게임개발자에게 윈도우즈와 도스환경 선택의 권한이 주어질 수도 있다. 그러나 사용자들은 복잡한 인스톨 과정이나 하드웨어 세팅을 원하지 않는다. 또한 게임개발자들에게 최상의 개발방법 선택은 풍부한 라이브러리와 툴을 가진 윈도우즈 환경에서 보다 손쉽게 게임을 제작하는 것이다. 이미 운영체제는 윈도우즈에 의해 장악되었으며 현재의 모든 3차원 컴퓨터게임개발은 윈도우즈 환경하에서 이루어지고 있다.

윈도우즈 프로그램은 일반적인 도스 프로그램과 많은 차이를 보이고 있는데 그 중에서도 가장 큰 차이점은 메시지에 기반한 이벤트 구조이다. 일반적으로 도스 프로그램에서는 코드에 따라서 순차적으로 진행시키면 되지만 윈도우즈에서는 윈도우즈 또는 프로그램에서 전달해주는 메시지에 따라 처리해주어야 한다. 사용자에게는 다소 불편하겠지만 컴퓨터를 게임기와 유사하게 만들수 있다는 측면에서 게임개발자에게는 좋은 여건이 될 수도 있다. 즉, 모든 프로세스 시간 및 하드웨어나 시스템 자원을 독점하거나 인터럽트를 가로채 사용자의 키보드 입력을 바로 처리할 수 있고, 원하는 그래픽 모드로 세팅한 후 바로 그래픽 카드를 제어하여 효율을 높일 수도 있는 것이다. 반면에 멀티태스킹 운영체제인 윈도우즈에서는 하나의 응용 프로그램이 하드웨어나 시스템 자원을 독점하거나 제어하는 것을 허락하지 않는다. 윈도우즈에서는 그래픽이 필요할 때는 Device Context를 얻어 해당 윈도우즈에 표현하며, 사운드 카드나 시리얼 포트등을 제어할 경우도 직접 액세스하지 않고 윈도우즈의 해당 라이브러리 함수를 호출한다.

한편, 윈도우즈는 이미 많은 부분이 라이브러리화 되어 있고 프로그래머는 필요한 라이브러리의 인터페이스에 대해서만 알고 있으면 쉽게 별다른 코딩없이 사용할 수 있다. 스프라이트나 맵 에디터등 게임에서 필요한 여러 툴들은 윈도우즈에서 MFC, 비주얼 베이직,델파이등을 이용해 도스에서보다 훨씬 빠른 시간에 제작할 수 있다. 윈도우즈(Window95/ Window98/ Window NT)는 32비트 보호모드를 기반으로 돌아가기 때문에 속도뿐 아니라 메모리 사용에서 도스에 비해 많은 잇점을 가지고 있다. 프로그래머는 도스에서의 복잡한 구조와 64KB의 메모리 한계를 벗어날 수 있다. 따라서 메모리에 그래픽, 사운드등 많은 데이터를 올려놓고 사용하는 게임 프로그래밍에서는 윈도우즈 환경이 도스환경에 비해 메모리 사용에 대해서는 장점을 갖고 있다.

2.4 3차원 컴퓨터게임 개발툴과 라이브러리

3차원 컴퓨터게임그래픽의 제작에는 다양한 3차원 그래픽 툴들이 사용되고 있다. 3차원 그래픽은 모델링(Modeling), 맵핑(Mapping), 렌더링(Rendering)등 많은 수학적 계산량을 요구하는 작업이 선행되어야 하므로 과거 3차원 그래픽 툴들은 웍크스테이션급의 고사양

컴퓨터에서 주로 사용되었다. 컴퓨터사양의 발달로 이러한 그래픽 툴들이 윈도우즈NT, 윈도우즈 등으로 포팅되면서 3차원 그래픽 제작의 비약적인 발전을 가져왔다. 이러한 3차원 그래픽 툴들에는 우리나라에서 가장 많이 쓰이고 있는 AutoDesk의 도스버전의 3D Studio, 윈도우즈버전의 3D MAX가 있고, 실리콘 그래픽스의 SOFTIMAGE, 방송용 애니메이션과 CF에 많이 쓰이는 LightWave 3D, 영화 고질라와 타이타닉, 인디펜스 데이에 사용된 것으로 유명한 HOUDINI가 있으며 최근 3차원 그래픽 툴로서 최고의 명성을 얻고 있는 실리콘 그래픽스의 MAYA와 NURBS[3] 모델러인 RHINO 3D, 그리고 Bryce 3D, POSER, TRUE SPACE 등이 있다.[6][7]

윈도우즈 환경에서는 많은 라이브러리(library)가 제공되어 이의 사용으로 보다 편리한 프로그래밍을 할 수 있도록 하고 있다. 3차원 컴퓨터게임 개발에서도 많은 라이브러리가 사용되고 있는데 이는 주로 3차원 그래픽과 관련된 것들이다. 3차원 그래픽라이브러리는 OpenGL, Glide, Direct3D등이 있고, 이들은 3차원 컴퓨터게임 개발에 필수적으로 쓰이고 있다. OpenGL은 UNIX 웍크스테이션에서 출발한 것으로 안정적이고 정교한 그래픽을 생성하지만 OpenGL을 지원하는 고가의 하드웨어가 동반될 때만이 최적의 효과를 낼 수 있다. Glide는 3Dfx사의 그래픽 가속칩인 VOODOO를 지원하기 위해 만든 것으로 VOODOO 칩이 탑재된 가속보드를 필요로 한다. Direct3D는 마이크로소프트가 제공하는 게임개발 툴킷인 DirectX의 하나로 범용성이 가장 뛰어나 3차원 컴퓨터게임 개발 시에 가장 적합한 그래픽 라이브러리로 여긴다.

Ⅲ. 3차원 레이싱 게임의 개발방법

3.1 3차원 레이싱 게임기획 및 시나리오

3차원 레이싱 게임을 개발하기 위해서는 게임에 등장하는 모든 스프라이트의 모델링 방법, 렌더링, 충돌검사등 게임프로그래밍 방법과 개발툴을 결정하는 것은 물론, 범용성과 일반 PC에서도 플레이가 가능한 3차원 PC게임이 될 수 있도록 사용자 환경을 고려하여야 한다. 무엇보다도 3차원 레이싱 게임의 속성상 속도를 최우선하여 디자인한다. 그래픽가속기가 없는 PC에서도 게임이 실행될 수 있도록 게임엔진의 실시간 렌더러를 자체 개발시에는 속도 구현을 위해 Z-buffer[4]보다는 Painter Sort[5]를 선택하는 것이 바람직하다.

게임의 기획은 게임의 성패를 결정짓는 중요한 요소기술로 사용자에게 게임의 차별화를 느낄 수 있게 하는 부분이다. 게임의 기획은 영화의 기획과 비슷하며 장르를 결정하고, 시나리오 작성과 그래픽 처리 방식등을 고려해 게임의 개발시스템을 결정한다. 특별한 시나리오를 필요로 하지 않는 레이싱 게임 같은 경우에는 사용자에게 속도감, 조작감을 우선적으로 부여할 수 있도록 구성한다. 레이싱게임에서의 시나리오는 게임의 도입과 결말의 모티브를 제공하고 게임 스테이지의 구성을 결정한다.

3.2 3차원 레이싱 게임의 그래픽스 구현

3차원 그래픽에서는 물체의 외형을 구성하는 모델링 데이터(3D Data), 질감과 색상을 표현하는 텍스춰어(Texture), 빛의 반사와 사실감을 더해주는 쉐이딩(Shading), 그리고 각 물체의 움직임을 나타내주는 모션데이터(Mortion Data)가 필요하다. 폴리곤만을 이용한 3차원 PC게임 그래픽은 실시간으로 렌더링을 해야만 하므로 PC의 성능과 밀접한 관계가 있으며, 게임실행 환경이 범용 PC라는 한계가 있으므로 게임실행 속도의 증진을 위해 일반적인 3차원 그래픽제작과는 달리 캐릭터당 사용가능한 폴리곤의 수를 한정시키는 등의 기법을 사용하고 있다.

최근에는 NURBS방식의 모델링이 등장하여 보다 사실적이고 부드러운 곡면을 가진 캐릭터 제작도 이루어지고 있다. 폴리곤 수의 제약이 없는 일반적인 3차원 그래픽에서는 그림 2와 같이 자동차 오브젝트를

3) None Uniform Rational B-Spline, 뼈대가 되는 선분을 기본으로 변들을 입혀가는 모델링 방식으로 곡면처리와 모서리의 라운딩 처리가 우수하다. 기존에는 폴리곤방식이 사용되었다.

4) 다각형을 그릴 때 각 픽셀의 Z값이 계산되어 그 점에 마지막으로 그려진 픽셀의 Z값과 비교해 이전의 값보다 POV에 더 가깝다면 픽셀이 기록되고 Z버퍼에 Z값을 저장한다. 정확한 계산이 가능하나 모든 픽셀에 대한 계산이 필요하므로 속도가 다소 느리다.

5) 가까운 곳에 위치한 다각형이 먼곳에 위치한 다각형을 가리는 방식으로 그리고 다각형의 평균 Z값을 이용해 정렬해 주는 방식

표현할 때 보통 수천 개의 폴리곤을 사용하여 이미지를 만들어 낸다.

그러나, 컴퓨터게임은 보통 프레임당 6~7천개의 폴리곤의 구현이 가능한 환경이므로 이 환경하에서 캐릭터와 배경을 그려내어야 한다. 따라서, 배경에 사용가능한 폴리곤 숫자의 1/2를 할당하면 캐릭터(자동차)를 3,000~3,500개의 폴리곤을 이용하여 그려야 한다. 또, 화면에 자동차가 동시에 5~6개까지 등장해야 하므로 결국 1대의 자동차를 그려내는 데에는 500~600개의 폴리곤을 사용하여야 한다. 이는 그래픽 디자이너들의 오랜 경험에서 얻어지는 제작 기법에 의해 해결할 수 있다.

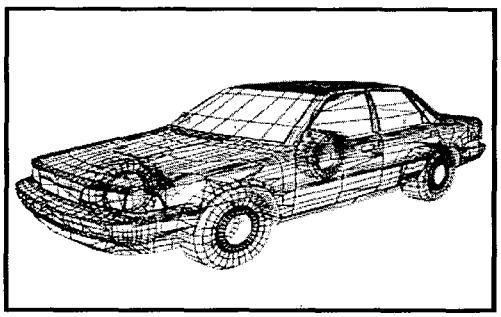

그림 2. 일반적인 컴퓨터 그래픽의 자동차 모델링
Fig. 2 Car Modeling for General Computer Graphics

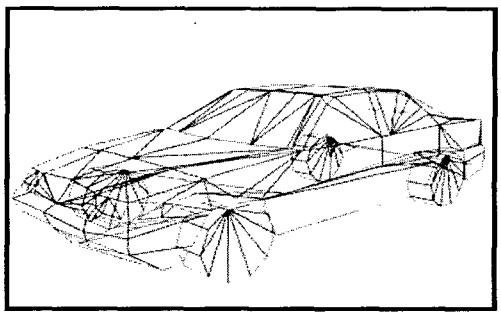

그림 3. 레이싱 게임 그래픽의 자동차 모델링
Fig. 3 Car Modeling for Racing Game Graphics

폴리곤을 이용한 3차원 게임그래픽에서는 모델링보다는 오히려 텍스처매핑기술이 더 중요할 때가 많다. 게임그래픽에서의 제한된 폴리곤으로 모델링한 오브젝트에 어떻게 텍스처매핑을 하느냐에 따라서 사실감의 차이는 크다. 국내 게임의 3차원 그래픽이 외국의 그것보다 다소 사실감이 떨어지는 이유도 바로 여기에서 기인한다. 그림 4와 그림 5는 그림 3의 모델링을 이용하여 최종의 캐릭터를 만들어 낸 것이다.

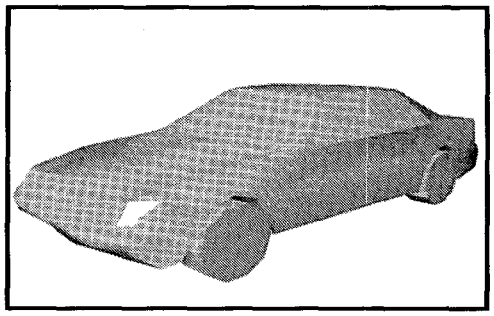

그림 4. 재질을 포함하지 않은 렌더링 이미지
Fig. 4 Rendering Image without Solid Textures

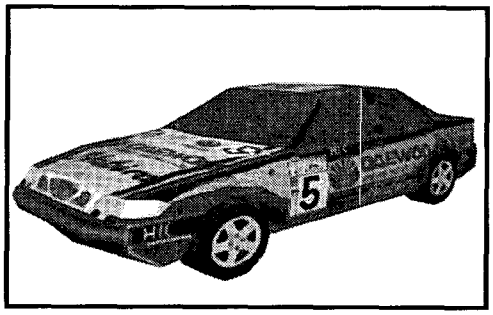

그림 5. 재질을 포함한 렌더링 이미지
Fig. 5 Rendering Image with Solid Textures

3.3 DirectX를 기반으로 한 개발

마이크로소프트의 DirectX API는 차세대 컴퓨터게임개발과 멀티미디어 어플리케이션 제작에 전략(Strategies), 기술(Technologies), 도구(Tools)를 제공한다.[7] DirectX는 윈도우즈 게임제작에 있어서의 화면과 사운드 출력, 사용자 입력처리, 3차원 그래픽, 네트워크 멀티유저 게임등을 지원하는 마이크로소프트가 제작한 라이브러리를 팩키지화 게임개발 툴킷이다.

DirectX는 DirectDraw, Direct3D, DirectSound, DirectPlay, DirectInput등으로 구성되어 있다. GUI라는 막강한 편리함을 제공하며 컴퓨터 운영체제 시장을 장악한 마이크로소프트의 윈도우즈 95, 윈도우즈 98로 인해 컴퓨터게임 개발자들은 DOS 운영체제와는 상관

없이 하드웨어를 직접 제어하던 과거의 컴퓨터게임 개발환경에서 윈도우즈 운영체제 위에서의 컴퓨터게임 개발환경으로의 전환이 불가피하게 되었다. 하지만 하드웨어를 직접 제어하지 못하는 윈도우즈 운영체제 위에서의 컴퓨터게임 개발환경은 개발자들이 좋은 게임을 만들어 내기에 적합하지 못한 것이었고 개발자들은 여전히 도스환경하에서의 게임개발을 선호하였다. 결국, 마이크로소프트는 게임개발자들이 쉽게 게임을 개발할 수 있는 툴을 제공하기 시작하였다. 최초의 게임 SDK[6]는 윈도우즈3.1 시절에 나온 WinG였다. 하지만 SDK로서의 기본자질을 갖추지 못한 WinG는 게임개발자들로부터 그다지 환영을 받지 못했다. 그러자 마이크로소프트는 윈도우즈 95의 등장과 함께 WinG를 업그레이드한 GDK[7]를 내놓았고 이를 버전업하여 새롭게 탈바꿈시킨 DirectX를 발표했다. DirectX는 계속된 버전업을 추구해 3.0버전에 이르러서는 게임개발자들의 열렬한 환영을 끌어냈다.[3] 이제 윈도우즈 환경하에서의 게임 개발자들은 현재 버전 7.0까지 출시된 마이크로소프트의 DirectX사용을 보다 적극적으로 사용하고 있다. 해외개발사들은 이미 오래전부터 DirectX를 이용한 윈도우즈용 게임을 제작해 왔지만 국내에서는 97년에서야 비로소 DirectX를 이용한 윈도우즈용 게임이 등장하고 있다.

DirectX의 핵심 구성요소인 DirectDraw의 구조는 그림 6과 같다. DirectDraw의 API는 기존의 윈도우즈 디바이스에 대한 인터페이스 GDI(Graphic Device Interface)와는 달리 COM(Component Object Model) 인터페이스를 이용하여 하드웨어 추상계층(HAL; Hardware Abstraction Layer) 혹은 하드웨어 에뮬레이션 계층(HEL; Hardware Emulation Layer)과 직접 대화함으로써 스피드 향상과 확장성을 동시에 가져왔다. HAL은 하드웨어에서 비디오 카드의 특정한 기능들을 직접적으로 지원하는 것을 말하며 HEL은 비디오 카드에서 특정한 기능들을 제공하지 못할경우 소프트웨어적으로 에뮬레이션하는 것을 말한다. 당연히 HAL쪽이 속도가 빠르다.

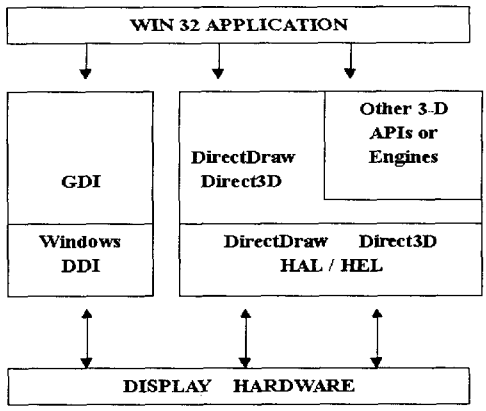

그림 6. DirectDraw의 구조
Fig. 6 Architecture of DirectDraw

DirectDraw가 제공하는 중요한 기능으로는 DirectDraw 표면(Surface)객체 생성에 의한 그들간의 빠른 Bitblt(Bit Block Transfer)기능, 컴퓨터 게임을 만들 때 가장 중요한 기능중의 하나인 프레임 버퍼와 오프 스크린 버퍼 사이의 이중 버퍼링 시스템(Double Buffering System), 팔레트 객체 만들기, 컬러 키잉(Color Keying)등이 있다.

Direct3D는 컴퓨터에서 리얼타임 3D 어플리케이션을 만들기 위해 제공된 3차원 그래픽 솔루션이다. 컴퓨터기반하에서의 빠른 속도의 3차원 그래픽을 위해 OpenGL과 같은 범용 3차원 그래픽스 라이브러리까지는 제공하고 있지는 않으나 라이팅(Lighting), 쉐이딩(Shading), 텍스처 매핑(Texture Mapping) 등 3차원 컴퓨터게임개발에 필요한 수준의 API는 모두 제공하고 있다. 또, Direct3D는 3D 비디오 디스플레이어 하드웨어에 대한 접근을 장치 독립적인 방법으로 장치 종속적이게 접근하는 방법을 취하고 있으므로 차후 새로운 디바이스가 등장하더라도 이식이 쉽도록 되어 있다.

이러한 Direct3D는 자체적으로 그래픽스 데이터를 유지하면서 제공되는 보다 높은 수준의 API인 Retained Mode와 Execute Buffer라는 그래픽스 데이터의 스트림을 직접적으로 건드릴 수 있는 저 수준의 API인 Immediate Mode로 나누어져 있다.

윈도우즈에서 사운드를 출력하기 위해 사용해왔던 기존의 API들은 일반적인 방법으로는 한번에 하나의 응용프로그램만을 지원할 수 밖에 없고 한번에 하나의 오디오 스트림만을 가질 수 있다. 또한 사운드가

6) Software Development Kit , 소프트웨어 개발툴 킷
7) Game Development Kit., DirectX 1.0의 옛이름

들리기 전에 100~150ms의 지연시간이 있다. DirectSound는 이러한 문제점들을 해결하고 게임에 보다 적합한 사운드출력위해 제공되는 API이다.

DirectSound는 믹싱이 가능하며 지연시간도 15~20ms밖에 되지 않는다. 그리고 응용프로그램간의 포커스가 바뀔 때마다 자동적으로 사운드 포커스가 바뀌도록 되어 있다. 만약 사운드카드에서 사운드 출력에 관한 특수한 기능을 지원하면 DirectSound는 HAL을 이용하여 하드웨어적인 가속 기능을 이용하도록 설계되어 있다.

DirectInput은 입력장치들에 대한 정보와 이들 입력장치들을 관리하기 위한 여러가지 API들을 제공한다. 다른 DirectX 컴포넌트처럼 DirectInput은 COM인터페이스에 기반을 두고 있다.

아날로그 조이스틱의 경우 4개의 축과 4개의 버튼을 가진 조이스틱 두개를 동시에 사용할 수 있으며, 2축과 4개의 버튼을 가진 조이스틱의 경우에는 동시에 4개를 사용하는 것도 가능하다. 아날로그 조이스틱과는 달리 디지털 조이스틱의 경우에는 6개의 축과 32개의 버튼을 가진 조이스틱 16개를 동시에 사용하는 것도 가능하다. 또한 일반적인 조이스틱 외에도 게임의 특성에 따른 진동전달 조이스틱이나 레이싱게임을 위해 스티어링 기능을 가진 특수한 기능의 Force Feedback 조이스틱도 지원한다.

에이지오브엠파이어(Age of Empire)나 스타크래프트(StarCraft)등 요즘의 게임은 모뎀이나 IPX를 사용하는 네트워크 환경을 지원하는 게임이 주류를 이루고 있는데 DirectPlay를 사용하면 일반적인 모뎀이나 시리얼 등의 네트워크 커넥션, 온라인 서비스를 이용하는 멀티플레이 게임을 더욱 손쉽게 제작할 수 있다.

DirectPlay의 기본 통신모드는 피어투피어(Peer-To-Peer)로 한 컴퓨터가 변화를 일으키면 세션(Session)에 이미 복사된 플레이어의 목록,이름, 원격데이타등의 세션의 완전한 상태(Sessions complete state)를 다른 컴퓨터의 세션으로 즉시 전파한다. 또, 클라이언트/서버 통신모드를 지원하는 데 이는 서버만이 세션의 완전한 상태를 가지고 클라이언트는 이를 이용한다.

3.4 3차원 컴퓨터 게임 프로그래밍

게임 프로그래밍에서 게임엔진을 개발하는 부분은 매우 중요한 위치를 차지한다. 게임결과물에 직접적인 영향을 미치기 때문이다. 3차원 그래픽의 가상공간에서 구현될 게임엔진의 대부분은 3차원 그래픽 제어와 깊은 관련이 있다. 3차원 컴퓨터게임 프로그래밍에서 구현되어야 할 핵심 게임엔진 3가지는 실시간 렌더러(Renderer), 3차원 충돌감지(Collision Check), 3차원 역학이다.

실시간 렌더러(Renderer)는 3차원 이미지를 화면에 실시간으로 실제 그려주는 엔진으로 얼마나 빠른 속도로, 중복계산없이 그려줄 수 있는가가 엔진성능을 평가하는 척도가 된다. 기 작업된 게임그래픽 3D 파일포맷의 변환, 투영(Projection), 클리핑(Clipping), 주사(Rasterization), 버퍼링(Buffering), 텍스처 매핑(Texture Mapping), 쉐이딩(Shading)등을 수행한다. 최근 그래픽 가속기의 등장으로 이를 지원하는 3차원 그래픽라이브러리의 이용은 3차원 컴퓨터게임프로그래밍의 기본으로 부상하였다.

3차원 충돌감지는 2D 게임과 달리 스프라이트들의 3차원 공간안에서의 충돌을 감지해내는 엔진이다. 설계된 게임규칙에 의해 사용자의 입력에 따라 인터랙티브하게 움직이는 스프라이트들의 충돌을 감지한다. 게임의 흥미도와 몰입을 좌우하는 게임플레이의 핵심엔진으로 구(sphere) 알고리즘[8]과 교차 방식 충돌 감지(intersection based collision detection)[9]가 있다.

3차원 역학은 실세계와는 다른 게임만의 역학(dynamics)을 프로그래밍하는 것이다. 예를 들면, 레이싱게임의 경우 실세계에서는 자동차가 트랙의 경사를 넘을 때 눈으로 확인이 어려울 정도의 미세한 비행이 발생하나 게임에서는 게임플레이의 흥미도 배가측면의 관점으로 다소 과장된 비행발생을 야기시킨다. 이러한 역학은 게임만이 가질 수 있는 고유의 것으로 이에 대한 특별한 프로그래밍의 처리가 필요하다. 또, 게임내 가상의 공간에서는 독자적인 중력장을 형성하는 등 게임의 3차원 역학은 실제와는 사뭇 다르다.

Ⅳ. 결 론

3차원 레이싱 게임의 개발은 하드웨어의 직접적인

8) 오브젝트를 구(Sphere)로 정의해 충돌을 감지하는 방식
9) 두 오브젝트를 다각형 대 다각형으로 표현하여 충돌을 감지하는 방식

액세스가 불가능한 운영체제의 제약, 실시간 게임그래픽의 제어, 게임실행시의 빠른 속도구현, 다양한 사용자 컴퓨터게임 실행환경, 등 매우 고려해야 할 것이 많으며 이에 따라 새로운 개발방법이 고려되어야 한다.

윈도우즈 환경 하에서 최적의 게임구현은 '어떻게 해야 하는가'라는 많은 게임개발자들의 질문에 개발툴킷으로써 DirectX가 그 대안으로 받아들여지고 있다.[8] 3차원 컴퓨터게임 개발에서 빠른 게임실행 속도의 구현과 다양한 하드웨어 제어, 개발기간 단축을 DirectX를 사용함으로써 해결할 수 있다. 게임개발시 하드웨어 제어가 게임프로그래밍의 대부분을 차지하고 있음을 고려한다면, 하드웨어 제어를 DirectX에 의존하는 윈도우즈 환경하의 3차원 컴퓨터게임개발은 도스환경하의 게임개발에 비해 대략 10분의 1정도의 시간이 소요되는 것으로 파악돼 현저한 개발기간의 단축을 가져온다.

과거 많은 게임 프로그래머들은 게임엔진을 처음부터 끝까지 자신이 만들기를 원했고, 각종 하드웨어를 제어하기 위해 많은 노력을 기울였으며, 또 그것이 게임 프로그래머로서의 능력을 판가름하는 기준이 되기도 했다. 그러나, 이러한 방식의 개발은 프로그래머에게 많은 개발시간의 투자를 요구하는 등 프로그래머의 부담을 가중시키는 것으로 그리 바람직한 것은 아니다. 뿐만 아니라 윈도우즈 운영체제는 프로그래머들의 이러한 시도를 사실상 불가능하게 만드는 환경을 제공한다.

3차원 컴퓨터게임 개발 시 우선 고려해야 할 것은 DirectX 같은 유용한 게임라이브러리를 확보하고 이를 게임개발 과정에서 어떻게 사용할 것인가 하는 정책을 결정하는 일이다. 이를 통해 유행에 민감한 패션상품의 속성을 갖고 있는 게임 개발기간의 단축을 기대할 수 있으며, 안정적인 게임환경을 확보할 수 있다.

앞으로 남은 과제는 본 연구가 게임개발의 수준을 가늠하는 2가지 요소인 게임설계(Game Design)와 게임실행(Game Play)에서 게임설계에 국한된 것이므로, 게임 실행측면에서의 체계적인 3차원 컴퓨터게임 개발론을 정립하는 것이다. 또, DirectX같은 유용한 라이브러리들을 채용한 3차원 컴퓨터게임 저작도구를 고안하여 3차원 컴퓨터게임 개발환경을 개선하는 일이 될 것이다.

참고문헌

[1] 한국컴퓨터게임학회, 한국 컴퓨터게임산업의 현황과 발전전망, 사단법인 한국컴퓨터게임학회 창립기념 심포지움, 1998.9

[2] 김영준, 3차원 컴퓨터게임개발에 관한 연구, 서울대학교 대학원 석사학위 논문, 1996

[3] Greg Costikyan I Have No Word & I Must Design, http://lunaris.hanmesoft.co.kr/~sonnet/rpg/noword.html

[4] Chris Crawford The Art of Computer Game Design, http://www.vancouver.wsu.edu/fac/peabody/game-book/

[5] Geoff Howland, Game Design: The Essence of Computer Games, http://www.devagmes.com/XPlus/Articles/essgames.html

[6] Microsoft DirectX 6.0 Software Development Kit Document, 1998

[7] OpenGL- The Integration of Windowing and 3D Graphics, http://hertz.eng.ohio-state.edu/~hts/opengl/article..html

[8] David Joffe, Game Programming with DirectX, http://www.geocities.com/SoHo/Lofts/2018/