3차원 게임 제작을 위한 렌더링 엔진의 설계와 구현

박태준°, 표순형, 추창우, 최병태, 오원근
한국전자통신연구원 가상현실(VR)연구개발센터

Design and Implementation of a Rendering Engine for Producing 3D Computer Games

Tae-Joon Park, Soon Hyoung Pyo, Chang Woo Chu, Byoung Tae Choi, Weon Geon Oh
Virtual Reality Research and Development Center, ETRI
E-mail: ttjjpark@etri.re.kr, shpyo@etri.re.kr, cwchu@etri.re.kr, btchoi@etri.re.kr, owg@etri.re.kr

요 약

하드웨어의 발달과 컴퓨터 그래픽스 및 가상현실 기술의 발전으로 일반 PC 상에서도 고품질의 영상을 실시간에 생성하는 것이 가능해졌다. 이러한 발전은 고해상도 가상공간 구축을 가능하게 하여 누구나 자신의 PC를 통해 가상 쇼핑몰이나 가상 박물관 등의 서비스를 이용할 수 있게 되었다. 최근에는 이러한 기술을 이용한 PC 환경에서의 3차원 게임이 속속 발표되고 있다. 본 고에서는 이러한 3차원 PC 게임 제작을 위해 필수적인 기술인 렌더링 기술 요구사항을 분석하고, 이를 렌더링 엔진의 형태로 설계하고 구현하는 방법을 제안한다.

1. 서론

컴퓨터 그래픽스 및 가상현실 기술의 발전과 하드웨어의 발전으로 과거에는 고가의 전용 장비에서만 가능했던 실시간 고품질 영상 생성이 일반 PC 상에서도 가능하게 되었다. 이러한 기술의 발전으로 인해 3차원 가상 쇼핑몰이나 3차원 가상 박물관 등의 서비스가 속속 등장하고 있으며 인터넷의 보급에 따라 누구나 네트워크를 통해 이러한 서비스에 접근할 수 있게 되었다.

최근에는 고성능 실시간 3차원 PC 게임이 속속 출시되고 있다. 사용자의 관심을 끌기 위해 환상적이고 사실적인 영상을 빠르게 생성해야 하는 게임의 특성 상, 그동안 3차원 게임은 PlayStation[1]을 비롯한 하드웨어 기반 전용 그래픽 처리기를 탑재한 게임 전용 기기를 위해서만 개발되어 왔다. 하지만 PC 환경에서도 CPU보다도 빠른 그래픽 처리기가 점차로 일반화됨에 따라 DOOM[2]이나 Quake 3[2] 등 환상적인 3차원 PC 게임을 접할 수 있게 되었다.

3차원 PC 게임의 제작을 위해서는 빠르고 사실적인 렌더링 기법의 구현이 필수적이다. 게임에서 렌더링이란 주어진 게임 환경의 정보를 영상으로 바꾸는 과정의 총칭으로 제작된 게임의 품질을 크게 좌우하는 부분이다. 대부분의 게임 개발사는 렌더링 과정에서 속도와 품질의 향상을 위한 나름대로의 knowhow를 축적하고 있으며 빠른 렌더링을 위해 각종 하드웨어 가속 장치와 OpenGL[3]이나 MicroSoft 사의 DirectX[4] 등의 API를 활용한다.

각 게임 개발사가 보유하고 있는 렌더링을 위한 각종 knowhow와 기술, 그리고 하드웨어를 제어하기 위한 각종 API의 사용법 등은 게임 하나만을 위해서만 사용되고 폐기되는 것이 아니라, 같은 게임 개발사의 다른 게임에도 계속적으로 보완 활용된다. 이러한 기술 재사용의 편의성을 위해, 각 게임 개발사는 자사의 렌더링 knowhow 및 기술을 언제라도 수정 및 보완하고 재사용할 수 있는 소프트웨어 엔진의 형태로 관리하고 있으며, 최근에는 Quake 3 엔진[2]이나 언리얼 엔진[5]과 같이 게임 자체보다는 그 엔진을 판매하는 사례도 나타나고 있다.

본 논문의 구성은 다음과 같다. 우선 2장에서는 보편적인 게임 제작을 위해 렌더링 엔진이 반드시 만족해야 할 요구 사항을 정리하며, 3장에서는 이러한 요구조건을 만족하기 위한 렌더링 엔진의 형태와 실제 렌더링 과정을 제안한다. 계속해서 4장에서 렌더링 엔진 구현 결과를 보였으며, 마지막으로 5장에서는 전체 논문의 결론을 내린다.

2. 렌더링 엔진의 요구 사항

3차원 게임 제작을 위해 렌더링 엔진이 반드시 만족하여야 할 사항들은 대단히 많다. 본 논문에서는 이러한 요구사항을 하드웨어 제어 부분과 소프트웨어 제어 부분, 렌더링 엔진이 처리할 수 있는 대상 객체의 종류, 개발 및 동작 환경, 렌더링 엔진이 사용되는 게임의 장르에 의한 요구 사항, 그리고 기능 측면에서 정리해 본다.

2.1 하드웨어 제어 부분과 소프트웨어 제어 부분에서의 요구 사항

최근 그래픽스 하드웨어의 발달로 일반 PC 환경에서도 대단히 빠른 속도로 실감 영상을 생성하는 것이 가능해졌지만 여전히 렌더링 속도는 하드웨어가 시간당 처리할 수 있는 다각형의 개수, 그래픽 하드웨어 내부의 pipe lining의 효율성, 그리고 환경의 복잡도에 의해 크게 영향을 받는다. 예를 들어 주어진 그래픽 하드웨어가 초당 백만 다각형을 처리할 수 있다고 해도, 주어지는 환경에 다각형이 천만개가 있다고 하면, 그림 한 장 생성하는데 10초가 걸리게 되어 실시간 렌더링이 불가능하게 된다.

이러한 문제를 극복하기 위한 소프트웨어적인 접근 방법으로는 LOD 기법[6,7,8]과 Culling 기법[6,7,8]이 있다. LOD 기법은 같은 대상 객체에 대해 하나의 기하모델만 작성하는 것이 아니라 다각형 수가 다른 여러 개의 모델을 작성하고, 화면에서 그 물체가 차지하는 크기에 따라 적절한 모델을 선택하여 렌더링하는 기법이다. 어떤 물체가 렌더링된 화면에서는 약 10 픽셀정도밖에 차지하지 않는데 이 물체가 매우 많은 다각형으로 이루어져 있다면 실제로 그림에서 얻게 되는 결과에 비해 비효율적으로 많은 처리시간을 빼앗기게 된다.

Culling 기법은 화면에 보이지 않다고 확신할 수 있는 다각형을 미리 소프트웨어적으로 선택하여 전체 렌더링 과정에서 미리 제외하는 기법이다. 이 기법에는 주어진 카메라 시야에 들어오지 않는 물체를 모두 제외하는 view frustum culling 기법, 물체를 구성하는 다각형 중 카메라 방향으로 향하지 않은 다각형을 미리 제거하는 back face culling 기법, 그리고 다른 물체의 의해 가려져서 보이지 않게 되는 다각형을 미리 제거하는 visibility culling 기법이 있다. 이러한 소프트웨어적인 기법을 이용하여 주어진 다각형 전체를 그래픽스 하드웨어에 전달하지 않고 그 중 일부만을 전달하여 렌더링 속도를 빠르게 할 수 있다.

렌더링 엔진의 요구 사항 중 하드웨어 적인 측면은 다음과 같이 정리할 수 있다. 대부분의 그래픽 하드웨어의 동작 구조가 렌더링의 각 단계를 병렬로 수행하면서 전체적으로 pipelining이 되도록 구성되어 있으므로, 효율적인 동작을 위해서는 이러한 pipelining이 멈추지 않고 계속적으로 진행하도록 최대한 하드웨어 내부의 상태변화를 막아야 한다. 이를 위해 각 객체를 렌더링할 때, 렌더링되는 다각형들을 저장한 메모리 블록을 하나로 유지한다거나, 같은 텍스쳐어를 사용하는 객체들은 하나로 묶어서 렌더링한다거나, 각종 render state[4]의 변화를 최소화한다거나 하는 등의 렌더링 엔진 최적화가 요구된다.

또한 그래픽스 하드웨어가 별도의 메모리를 가지고 동작하는 만큼, 렌더링되는 환경의 크기가 이 메모리의 용량에 다 들어갈 수 있도록 해야 한다. 만일 환경의 크기가 이 용량을 초과할 경우 그래픽 하드웨어의 메모리와 시스템 메모리 사이의 data swapping이 발생하여 전체 시스템의 성능이 저하된다. 가장 문제가 되는 부분이 텍스츄어인데, 텍스츄어 압축[4]이나 mipmap 기법[4] 등을 이용하여 가능한 이들이 차지하는 영역의 크기를 줄여야 한다.

2.2 대상 객체 종류에 대한 요구 사항

구현되는 렌더링 엔진은 게임 제작에서의 활용을 목적으로 하는 만큼, 게임에 등장하는 여러가지 대상 물체를 화면에 출력할 수 있어야 한다. 게임에 등장하는 대상 물체로는 지형, 나무나 집 등의 정적인 물체, 사람이나 동물, 괴물 등의 동적인 캐릭터, 각종 특수효과 등이다.

동적인 캐릭터가 등장하는 게임의 구성을 위해서는 렌더링 엔진 뿐만 아니라 이들 캐릭터의 동작을 생성하는 애니메이션 엔진이 함께 활용되어야 한다. 애니메이션 엔진은 키프레임 기반의 동작 지정과 같은 기본적인 방법에서 최신의 동작 캡춰 기법과 같은 고급 기법을 사용한 동작 생성을 담당해야 하며 렌더링 엔진은 화면에 출력되는 캐릭터에 이들 애니메이션 엔진의 동작 생성 결과를 반영할 수 있어야 한다. 렌더링 엔진에서는 동작 캐릭터를 skelecton model이나 skinned mesh [4]형태로 구성할 수 있어야 하며 이들을 효율적으로 제어할 수 있어야 한다.

2.3 개발 및 동작 환경에 대한 요구 사항

렌더링 엔진의 원활한 동작을 위해서는 GPU가 탑재된 전용의 그래픽 가속기와 이를 제어할 수 있는 API 활용이 필수적이다. 많은 그래픽 하드웨어 vendor들이 나름대로의 가속 하드웨어를 출시하고 있으나 그래픽 API는 Silicon Graphics 사를 주축으로 개발되어 온 OpenGL과 MicroSoft 사의 Direct3D로 크게 양분된다. 둘 중에 어느 하나가 나머지 하나보다 뛰어나다고 확정지어 말할 수는 없으며, 시간과 인력이 충분하다면 같은 렌더링 엔진을 Direct3D 버전과 OpenGL 버전으로 별도로 개발하는 것이 바람직하다고 본다.

2.4 게임 장르에 의한 요구 사항

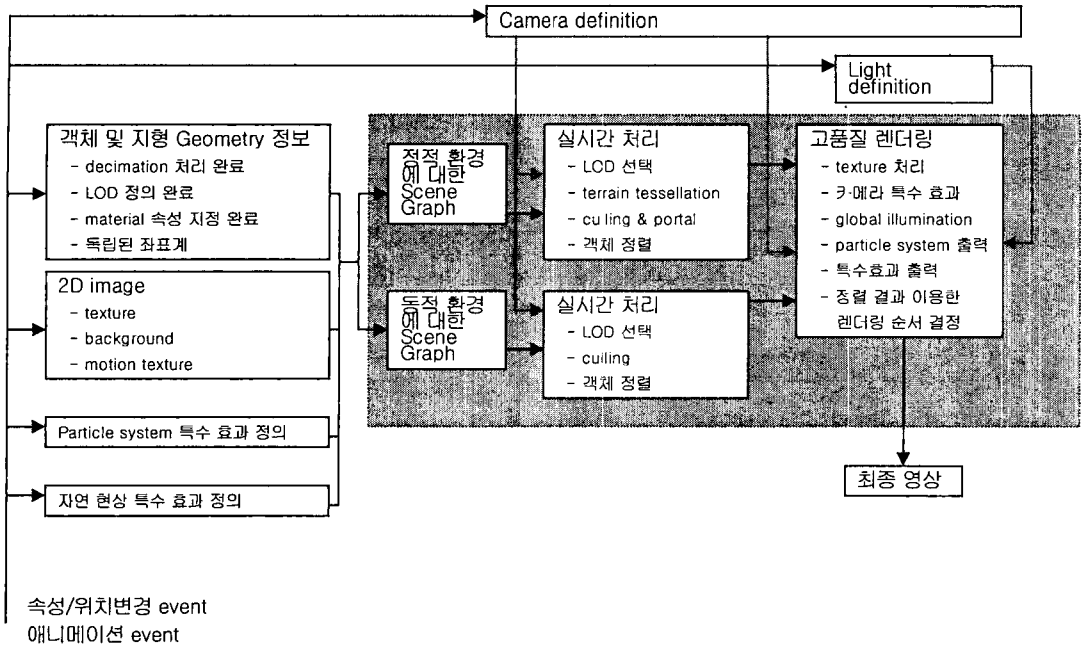
게임의 장르를 구분하는 기준은 많이 있지만 렌더링 엔진 구현의 관점에서는 출력해야 하는 환경의 속성에 따라 outdoor 게임과 indoor 게임으로 크게 구분할 수 있다. Outdoor 게임은 게임 환경의 크기가 방대하며 넓은 지역을 표현하는 지형에 대한 처리가 중요하다. 특히 지형을 다각형으로 표현하는데 있어 그 다각형의 개수를 조정한다거나 지형과 다른 객체간의 충돌을 어떻게 판단하느냐 등이 큰 문제이다.

Indoor 게임의 경우 그 특성상 주어진 환경이 복도나 방 등 실내인 경우가 많으며 따라서 벽 등의 물체가 그 뒤의 물체를 가리고 있는 경우, 또는 옆방이 열려있는 문을 통해서만 들여다보이는 경우가 많다. 그러므로, indoor 게임의 경우 화면에 표시되지 않는 객체를 빠르게 판단하여 렌더링에서 제외하는 기능이 반드시 요구된다.

렌더링 엔진 입장에서 본 또 다른 게임 장르 분류 기준은 2.5D 게임과 3D 게임이다. 2가지 부류 모두 게임 환경 자체는 3D이나 2.5D 게임의 경우 게임 제작의 편의를 위해 캐릭터와 게임 환경 사이에서의 상호 작용이 지형을 기반으로 하여 2D 공간에서만 발생한다는 제한점을 가지고 있다. 이러한 상호 작용을 어느 정도까지 허용하느냐에 따라 렌더링 엔진 입장에서는 내부 자료구조의 구성과 각 객체에 대한 접근 방법 등이 달라지게 된다.

2.5 렌더링 기능에 대한 요구 사항

이상과 같은 요구 사항 이외에 렌더링 엔진 기능에 대한 요구 사항은 다음과 같다 [9].



[그림 1] 렌더링 엔진 설계 개념도

- 3차원 객체를 기존의 상용 프로그램을 이용하여 모델하고 export하는 기능
- 3차원 객체를 텍스츄어와 광원 효과와 함께 렌더링하는 기능
- 광원과 물체와의 상호작용을 고려한 고품질 영상 생성 및 그림자 제어 기능
- 투명, 반투명 물체에 대한 처리와 빛의 굴절 및 반사 효과 처리 기능. 반투명 물체를 화면에 올바르게 출력하기 위해서는 이들을 카메라에서 먼 것부터 가까운 순으로 정렬하여 출력하는 기능이 필요하다.
- 인공 지능 카메라 제어 기능
- 자연 현상, 카메라 효과, 각종 particle system 출력 등의 특수 효과 생성 기능

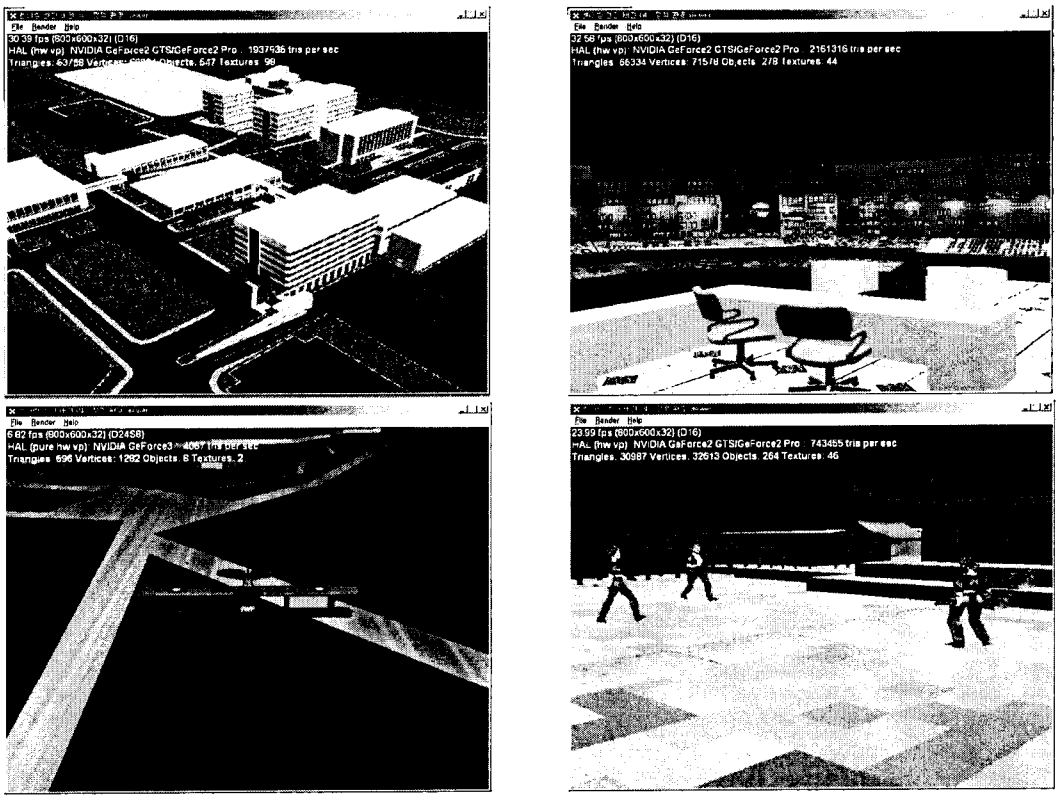
3. 렌더링 엔진 설계 개념도

이상의 요구 사항에 따라 설계된 렌더링 엔진의 기본적인 설계 개념도는 [그림 1]과 같다.

3.1 입력되는 자료

렌더링 엔진에 입력되는 자료는 다음과 같다

- 내부 렌더링 객체
  - 표면 속성 정보를 포함한 객체 및 지형의 기하 정보
  - 텍스츄어로 사용되는 이미지
  - Particle system, 자연현상 특수효과를 지정한 자료 파일
  - 광원 및 카메라 초기변수 정보
- 외부 제어 이벤트
  - 각 객체의 속성 및 위치 변경 이벤트



[그림 2] 렌더링 엔진 동작 결과

- 애니메이션 제어 이벤트
- 특수효과, particle 제어 이벤트
- 광원 및 카메라 속성 변경 이벤트

3.2 환경 SceneGraph 및 실시간 처리를 위한 처리

설계된 렌더링 엔진에서는 객체 관리의 편의를 위해 여러가지 렌더링 객체를 동일한 구조의 scene graph로 구성하여 내부 관리한다. Scene graph는 동적인 물체와 정적인 물체에 대해 별도로 구성되며, 실시간 처리를 통해 실제 렌더링 될 다각형과 전환 행렬 (transform matrix), 텍스츄어 정보로 변경된다.

실시간 처리 모듈에서는 주어진 scene graph를 탐색하여 각 객체에 LOD 기법을 적용하여 물체를 구성하는 다각형 수를 제어하고, 지형 정보로부터도 역시 적정한 수의 다각형을 생성한다. 이 때 앞서 설명한 culling 기법을 적용하여 렌더링될 필요가 없는 부분은 미리 제외한다. 생성된 다각형들은 카메라로부터의 거리에 따라 정렬되어 그래픽스 하드웨어로 전달된다.

속도의 향상을 위해 정적환경에 대한 실시간 처리는 카메라 변수가 변경된 경우에만 수행하며, 동적환경에 대한 실시간 처리는 대상 객체의 자세가 변경될 때마다 수행한다.

하드웨어 렌더링 단계에서는 카메라 변수와 광원 변수, 그리고 각종 텍스츄어와 선택된 다각형을 대상으로 영상을 생성한다. 각종 특수효과와 particle system의 출력 또한 이 부분에서 담당한다. 반투명 물체의 올바른 출력을 위해 그래픽 하드웨어로 전달되는 다각형들은 이전 실시간 처리 단계에서 카메라로부터의 거리에 따라 정렬된 순서에 따라 처리된다. 이 때, 가능한 같은 텍스츄어를 사용하는 다각형들이 함께 모여서 렌더링되도록 하여 그래픽스 하드웨어 내부에서 발생하는 텍스츄어 상태변화를 최소화하며,

텍스쳐어 압축 및 mip map 생성 기법을 적용하여 텍스쳐어가 그래픽 메모리에서 차지하는 용량을 줄인다.

4. 구현 결과

설계된 렌더링 엔진을 Nvidia GeForce 2 GTX 그래픽 보드가 장착된 Pentium 3 PC에서 MS Visual C++과 DirectX를 이용하여 구현하였다. 또한 사용되는 렌더링 객체는 3D Studio MAX를 이용하여 작성하여 별도로 구현한 exporter에 의해 독자적인 파일로 저장되었으며 렌더링 엔진에 의해 읽혀져서 화면에 출력된다.

텍스쳐어는 TGA, BMP, JPG 등 DirectX에서 지원하는 이미지 형식으로 만들어질 수 있으며 투명/반투명 속성을 표현하는 alpha channel이 지원된다.

동적인 물체를 처리하기 위해 렌더링 엔진에서는 skelecton mesh와 skinned mesh를 모두 지원한다. Skinned mesh를 이용하는 경우 3D Studio MAX 4.0에 별도로 함께 제공되는 Character Studio를 이용한 동작 제어 결과를 그대로 렌더링 엔진에서 활용할 수 있다.

또한 각종 그림자 효과와 광원 효과를 생성하기 위한 multi texturing 기법이 구현되어 있으며, 동작물체에 대한 그림자 생성도 가능하다. [그림 2]에 구현된 렌더링 엔진의 실행 결과 스크린 샷을 담았다.

구현 결과 초당 30에서 50장 정도의 영상생성이 가능하였으며 초당 백만 개 이상의 다각형 처리능력을 보였다.

5. 결론

본 논문에서는 한국전자통신연구원 가상현실연구개발센터에서 과제로서 진행하고 있는 3D 온라인 게임엔진 개발 과제의 일부인 렌더링 엔진의 설계 및 구현에 대해 살펴보았다. 3D 게임을 제작하기 위한 렌더링 엔진의 요구 사항을 여러 관점에서 살펴보았으며 이를 충족하기 위한 렌더링 엔진의 구조를 설계하였으며 MS Direct3D와 Visual C++를 이용하여 GeForce 2 GTX 그래픽 가속기가 장착된 Pentium 3 PC에서 이를 개발하였다. 개발 결과 초당 30에서 50장 정도의 영상을 생성할 수 있었으며 역시 초당 백만 개 이상의 삼각형을 처리할 수 있었다.

향후에는 이 렌더링 엔진을 OpenGL 등 다른 그래픽스 API를 이용하는 버전으로 전환할 계획을 가지고 있으며 Web3D 등의 응용 분야에 활용하는 방안을 계획 중이다. 또한 애니메이션 엔진 및 게임 진행 엔진 등과의 통합을 통해 상용 게임의 제작에 활용될 수 있도록 하는 방안을 추진중이다.

[참고문헌]

[1] http://www.sonyweb.com/

[2] http://www.idsoftware.com

[3] M. Woo, J. Neider, T. Davis, and D Shreiner, "OpenGL Programming Guide 3rd Ed.", Addison-Wesley, 1999.

[4] http://www.microsoft.com DirectX 8.0 Online Mannual

[5] http://www.infograms.com

[6] T. Moeller and E. Haines, "Real-Time Rendering", A K Peters, 1999.

[7] A. Watt, "3D Computer Graphics 3rd Ed.", Addison-Wesley, 2000.

[8] A. Watt and F. Policarpo, "3D Games: Real-Time Rendering and Software Technology", Addision-Wesley, 2001.

[9] 유채곤, "게임 제작용 3D 그래픽 클래스 설계", 사단법인 한국게임학회 2001년 하계 학술발표대회 논문집, pp. 107-110, 2001.