2002년 한국정보처리학회 춘계학술발표논문집 제9권 제1호

실시간 3차원 공개 게임엔진 비교 분석

허원[0], 황요한, 김동균, 신동규, 신동일
세종대학교 컴퓨터공학과
e-mail : { heowon[0],xfilen,kd999,shindk,dshin}@gce.sejong.ac.kr

Comparative Analysis of Realtime 3D Open Source Game Engine

Won Heo, Dongil Shin, Dongkyoo Shin
Dept. of Computer Engineering, Sejong University

요 약

본 논문에서는 실시간 3 차원 공개 게임 엔진에 대해서 간단한 소개와 함께 게임 엔진으로서 중요한 요소를 기준으로 비교분석 하였다. 개인의 개발연구용과 학교 및 연구소에서 비영리적으로 이용이 가능한 실시간 3 차원 게임 엔진에 대해서 간단하게 소개하며 각각의 특성을 비교 분석함으로써 더 많은 개발자들의 게임 엔진 연구에 효과적인 역할을 하였으면 한다.

1. 서 론

현재는 2 차원에서 3 차원으로의 게임 컨텐츠가 전환되는 시점에 놓여 있다. 이미 패키지 게임 산업은 화면의 구성 및 게임 컨텐츠의 구성을 3 차원으로 가져간지 오래되었으며 온라인 게임과 네트워크 게임들도 차차 그 입지를 3 차원으로 옮겨가고 있는 실정이다. 지금의 3 차원에서 게임 컨텐츠의 표현방식은 2 차원의 방식과는 달리 좀 더 복잡하고 정교하며 현실에 가까운 그래픽과 사운드를 접목하여 인터렉티브한 게임 환경을 제공하고 있다. 이러한 게임 환경을 제공하는 기술적인 중심에는 3 차원 게임 엔진이라는 핵심 기술이 존재하며 수많은 게임 개발 회사와 개발자 그룹에서는 좀 더 향상된 3 차원 게임 엔진의 연구와 개발에 대한 투자를 아끼지 않고 있다. 그 성과물로 다양한 종류의 3 차원 게임 엔진이 등장을 하였으나 엔진을 이용하여 3 차원 게임 컨텐츠를 개발하기 위해서는 높은 비용을 지불해야만 한다.

이에 본 논문에서는 개인의 개발연구용과 학교 및 연구소에서 비영리적으로 이용이 가능한 실시간 3 차원 게임 엔진에 대해서 간단하게 소개하며 각각의 특성을 비교 분석하였다. 2 장에서는 3 차원 게임 렌더링 엔진에 대한 내용을 기술하고 3 장에서는 공개 엔진들의 비교분석을 논하고 4 장에서는 결론 및 향후 전망에 대해 기술한다.

2. 관련연구

3 차원 게임 렌더링 엔진의 요소에는 다음과 같다.

2.1 렌더링 요소

2.1.1 광원효과(Lighting & Shading)

Lighting 은 3D 렌더러에서 객체나 지형에 빛의 효과를 반영하는 과정이다. 라이트 효과는 주변광(ambient), 난반사광(diffuse), 전반사광(specular), 방사광(emissive)이라는 네 종류로 분류된다. 정적 라이팅(Static Lighting)은 고정 광원이 고정 객체에 주는 최종 효과를 계산하는 것이다. 이 계산들은 오프라인에서 수행될 수 있으며 버텍스들은 렌더러가 사용할 최종적인 색을 간단히 저장 할 수 있다. 동적 라이팅(Dynamic Lighting)은 실행시간에 광원이 꼭지점에 주는 최종 효과를 계산하는 것이다. 버텍스 법선(normals)은 방향과 점 또는 스포트 라이트를 비롯한 계산에서 필요하기 때문에 꼭 저장이 되어야 한다. 실시간 3D 에서 자연과 같은 빛의 처리는 무리이기 때문에 단순화된 모의 광원의 효과를 가정해 계산한다.

Shading 은 빛으로 인한 명암을 폴리곤에 반영하는 과정이다. 플랫 세이딩(Flat shading)은 렌더링된 삼각형의 모든 픽셀들에게 동일한 색을 적용시키는 방법이다. 그라운드 세이딩(Ground shading)은 삼각형의 버텍스에 라이팅 방정식을 적용하여 최종 버

텍스의 색상을 구하고 그 색상을 보간하여 나머지 픽셀의 색상을 적용시는 방법이다. 퐁 세이딩(Phong shading)은 각 픽셀에서의 버텍스 법선(normals)을 보간하고 픽셀마다 라이팅 방정식을 적용하는 방법이다. 이 방법은 비용이 많이 들기 때문에 대게 실시간 시스템에서는 사용되지 않는다. 실시간 3D 에서는 모든 점에 대해 lighting을 적용하는 것은 어렵기 때문에 면단위로 적용한 세이딩 테크닉이 주요하게 사용된다[1][2][3].

2.1.2 텍스처링 (Texturing)

텍스처는 폴리곤 위에 입혀지는 2 차원 이미지라고 말한다. 텍스처 자체의 이미지는 단순한 2 차원적 평면이지만, 이것이 3 차원 Mesh 에 알맞게 포장되면 빛의 효과로는 표현하기 어려운 세부적인 사실감이 표현 가능하다. 멀티 텍스처링은 동일한 Mesh 에 대해 여러 개의 텍스처를 동시에 적용할 수 있는 기능이다. 각각의 텍스처는 스테이지별로 분리되어 순서대로 사용되며, 이때 스테이지간의 다양한 blending 효과를 지원함으로써 라이트 매핑, 환경매핑 그리고 그림자 같은 상위 레벨의 효과를 빠르게 처리하는 것이 가능하다. 텍스처링된 삼각형이 화면에서 멀리 있게 보일 경우 선이 " 들쭉날쭉" 하게 표현되는 Aliasing 문제가 발생하는데, 이 Aliasing을 줄이기 위한 방법으로 약간의 번짐효과를 적용하는 밉매핑이 사용된다[1][2][3].

2.1.3 라이트 매핑

기존의 라이트 방식은 작은 정점으로 이루어진 Mesh 에 적용할 때 정확성이 떨어지고 만족스러운 결과물을 얻을 수 없는 단점이 있다. 라이트 매핑의 기본 원리는 기본 텍스처 위에 밝기를 나타내는 또 하나의 텍스처(라이트 맵)를 준비해 이를 멀티 텍스처링하는 것과 같다. 이때 사용되는 라이트 맵은 대개 기본 텍스처보다는 아주 작지만 필터링이 이루어져 부드럽게 보인다[1][2][3].

2.1.4 환경매핑

환경매핑은 주위의 환경을 텍스처로 투여해 준비하고, 이것을 실시간으로 시점과 반사되는 각도로 보이도록 텍스처 좌표를 설정해 주위의 물체가 실제 표면에서 반사되는 것처럼 보이는 효과를 내는 테크닉이다. 이 효과는 미러 효과 또는 실제 라이트 소스가 장면에 존재하지 않을 경우 다수의 라이트 소스 효과를 제공할 때 유용하다[6].

2.2 Culling System

culling 은 시점에서 보이지 않는 객체의 부분들을 제거하는 작업으로서 전체 객체를 제거할 수도 있다. 이 작업을 함으로써 화면에 렌더링이 될 객체들중에 시점에서 보이지 않는 부분의 데이터들을 줄여나가 렌더링의 성능을 향상시킬 수 있다.

2.2.1 지형처리 방법

3D 게임의 특징은 가상공간으로의 시점 및 카메라의 이동 제약이 적다는 것이다. 커다란 가상공간에 대한 효율적인 공간처리 방법이 필요하다. BSP(Binary Space Partitioning)는 이분적인 공간분할 방법으로써 실시간으로 동시에 처리하기 불가능한 지형 데이터를 이진트리 형태로 분석해 노드 정보를 구성하고 이를 참조해 대상 물체를 탐색해간다. Octree 는 8 개의 자식을 가진 트리에서 하나의 노드를 나타내는 데이터 구조로서 대개 공간분할에 사용된다. Octree 는 BSP 트리의 특수한 경우이다. 포털은 실내 환경에서 주로 적용되는 것으로, 벽으로 나눠진 방과 같은 구조에서 연결되는 문이나 창과 같은 부분으로 들여다 보는 방법을 말한다. 관찰자는 통로의 벽이 다른 객체들을 막기 때문에 통로를 통해 보이는 것만을 볼 수 있다. PVS(Potentially Visible Set)는 보여질 가능성이 있는 리스트를 기억해 두는 방식으로 BSP 같은 공간분할방식과 포털(Portal)이라는 테크닉을 같이 사용하여 구현한다[4][5].

2.2.2 LOD

LOD(Level Of Detail)란 게임 사용자의 시점에서 먼 거리에 있는 물체는 적은 수의 폴리곤으로 표현하고 가까이에 있는 물체는 폴리곤의 수를 증가시켜 표현하는 기술이다. 이전에는 몇 단계의 모델을 거리별로 직접 제작해 사용했다. 하지만 요즘 새로 개발되는 게임은 실시간으로 폴리곤의 개수를 조절할 수 있는 기능을 가지고 있다[4][5].

2.3 Dynamic Effect

2.3.1 Particle System

조그마한 객체들의 모음으로, 삼각형 메쉬의 꼭지점에 색상이 지정되는 방식과 같이 점이나 사각형에 색상이 지정된다. 연기나 안개, 분수와 같은 다양한 특수효과들의 시뮬레이션에 사용될 수 있다. 파티클들의 운동은 물리학의 영향을 받는다.

2.3.2 그림자

그림자는 최근 들어 활발하게 적용되기 시작했는데, 이는 자연스러운 그림자 처리를 위한 수학 연산과 이에 대한 CPU 사이클링에 대한 비용이 많이 걸리기 때문이다. 적절한 그림자 효과는 장면의 볼륨감 및 원근감의 표현을 보다 사실적으로 나타내어주어 게임 환경을 보다 현실감있게 구성할 수 있는 장점이 있다[1][2][3].

2.3.3 Bones-Animation

객체안에 Bone 을 추가함으로 써 매 프레임마다 삼각형의 모든 정보를 갖는 것이 아니라, 단지 Bone 의 위치값과 회전값만을 가지고 객체에 움직임을 줄 수

있다. 이 방법을 사용하면 퀘이크[7]에 비해 캐릭터의 깨짐이 적게 나타나는데 조금 느려지는 단점이 있다.

2.4 Shader

2.4.1 Vertex Shader

DirectX 8.0 부터 새로 지원되는 기능으로, 하드웨어의 지원을 받아 3차원의 버텍스에 대한 속성값들을 스크립트로 제어하는 방식으로 처리함으로써 렌더링의 품질을 향상시킨다. 이 방법을 사용하면 버텍스 에니메이션과 카툰 렌더링, Motion Blur, Morping 등을 만들 수 있다.

2.4.2 Pixel Shader

Geforce 3[8] 부터 지원되는 기능으로, 렌더링 된 후의 버텍스인 2 차원의 픽셀을 제어하는 방법으로, 이 방법을 사용하면 전문적인 3차원 렌더링 프로그램으로 렌더링한 듯한 화면을 빠른 시간 내에 구할 수 있다.

3. 실시간 3차원 공개 게임 엔진 비교분석

3.1 공개 게임 엔진에 대한 소개

본 논문에서는 실시간 3 차원 공개 게임 엔진으로 퀘이크 2, Genesis 3D, Crystal Space, Fly 3D, Nebula Device를 소개 및 비교분석을 하겠다.

3.1.1 퀘이크 2

ID 소프트에서 만든 퀘이크 2[7] 엔진의 특징은 높은 차원의 렌더러에 있다. 공간분할은 BSP 를 사용하였고, 애니메이션에 있어서는 버텍스 애니메이션 기법을 사용하였다. 최초의 라디오시티 라이트 맵을 사용한 엔진답게 라이트 맵의 활용이 매우 뛰어나다. 실시간 빛 효과를 내기 위해 컬러 라이트 맵을 사용해 빛의 움직임을 표현한것과, 환경 매핑을 실제 게임에 사용하는 등 뛰어난 화면을 위한 많은 기술적 시도가 있었다. 놀라운 사실은 Geforce 2[8]부터 지원하기 시작했던 버텍스 세이더가 퀘이크 엔진에서는 이미 지원되고 있다는 것이다. 그러나 다른 상용엔진에 비해 사용하기가 어렵다는 단점이 있다. 이를 증명하듯 거의 비슷한 평가를 받는 언리얼 엔진에 비해 퀘이크 엔진을 사용한 게임은 소수에 불과하다.

3.1.2 Genesis 3D

Eclipse Entertainment 에서 개발된 Genesis3D[9]는 공개엔진이라 하기 무색할 정도로 일반 상용 게임 엔진과 같은 비슷한 기능들을 제공한다. 라이트맵을 이용한 원근 텍스처 매핑, 움직이는 물체에 대한 텍스처 처리, 반투명한 텍스처 등 사실적인 표현을 위한 텍스처 기법들이 사용되었다. 동적인 그림자 효과와 빛의 회절, 다양한 색의 빛을 표현할 수 있다. 실시간으로 물에 대한 표면의 뒤틀림을 표현해 줄 수 있다. 그리고 월드 객체에 대한 강체(rigid body)물리학 시뮬레이션을 제공한다. LOD 기법이 사용되고, 지형처리방법으로는 포털이 사용된다.

3.1.3 Crystal Space

Crystal Space[10]는 진정한 의미의 6DOF(degree's of freedom)[12]를 구현하여 사용자가 머리를 앞으로, 뒤로, 왼쪽으로, 오른쪽으로, 위로, 아래로 움직일 수 있다. 다른 모듈을 쉽게 추가할 수 있게 플러그인 시스템을 사용하고 있다. 텍스처는 크기가 두배가 되어도 상관이 없고 꼭 사각형일 필요가 없다. 또한 JIF, TGA, BMP, JPG,PNG 등 여러 종류의 그림파일들을 텍스처에 적용시킬 수 있다. 회전시키거나 크기와 좌우를 바꿔서 폴리곤에 텍스처를 입힐 수 있으며 수면이나 유리창과 같이 투과되거나 반 투과되는 텍스처를 만들 수 있다. 일반적인 라이트매핑된 텍스처들에 대해서 지형을 표현하는데 사용했던 삼각형들을 사용할 수 있다. 텍스처 케시에 대한 메모리 소모를 감소시키고, 폴리곤이 멀리 있을 때 텍스처들이 더 좋게 보기게 하기 위해 밉매핑이 사용되었다. 물체 반사 효과를 지원하고 빛과 그림자가 미리 계산 된 후 표현된다. 거울들과 알파 블렌딩을 이용하여 사실적으로 빛나거나 반사되는 표면들을 표현할 수 있다. bezier 와 같은 곡면을 만들 수 있다. 지형처리 방법으로는 포털과 Octree, BSP 트리와 c-버퍼(coverage buffer)를 함께 사용한다.

3.1.4 Fly3d

Fly3d[11]는 한 유니트에 4 개까지의 텍스처를 매핑 시킬 수 있고, 세부적인 텍스처를 표현할 수 있다. 정적인 지형을 표현하는데 라이트맵이 사용되었고, 동적인 빛과 라이트 맵을 이용한 그림자를 표현해 줄 수 있다. 동적인 객체들을 표현하기 위해 그래픽카드에서 지원되는 난반사와 전반사광을 사용할 수 있다. 충돌 검사와 파티클 시스템이 구현되어 있고, 기본적인 물리현상이 구현되어 있어서 간단한 물리 시뮬레이션이 가능하다. 메시들에게 움직임을 줄 수 있고, 귀엽고 따뜻한 느낌의 3D 효과를 주는 카툰렌더링이 사용되었다. 텍스처로 사용될 수 있는 파일은 TGA 와 JPG 두 개의 파일만 가능하다. 엔진은 완전하게 플러그인 형식으로 제어가 된다. 즉 게임을 만들거나 게임에서의 객체를 만들 때에는 플러그인을 만드는 형식으로 만들면 된다. 지형처리 방법으로 BSP 와 PVS를 동시에 사용하였다.

3.1.5 Nebula Device

Radon Labs[13]에서 개발된 Nebula Device 는 C++ 클래스들의 모음인 객체지향적 구조를 가지고 있다. 객체들이 가지고 있는 렌더링에 관한 정보들이 D3D 와 OpenGL 에서 동시에 사용될 수 있도록 저장이 되어 있어서 상호간에 변환하여 렌더링할 때 아무런 제약을 받지 않는다. 하드웨어에서 지원되는 Transform and Lighting(T&L) 기능을 사용하기 때문에 버텍스

변환, 빛의 확산효과, 클리핑, 숨은 면 제거등 의 기능을 하는 변환 함수들이 존재하지 않는다. 멀티텍스처를 지원하고, 버텍스 세이더 대신에 버텍스 블렌딩을 지원하며 간단한 코드 작성과 자동적인 밉맵 생성을 위해 D3DX 라이브러리를 사용한다. 그리고 무게감 있는 버텍스 스킨을 이용한 실시간 본 애니메이션이 가능하다.

이 엔진에서는 픽셀 포맷 변환 때문에 텍스처 로더가 심하게 느리다. 픽셀 포맷 변환 방법은 픽셀 변환을 최적화시키는데, 만약 충분하지가 않다면 텍스처의 포맷을 가지고있는 디스크 캐쉬를 포함을 함으로써 이 문제를 해결한다. 이러한 방법 때문에 실시간 렌더링 에서는 디스크에서 정보를 읽어들여야 되기 때문에 느리게 된다.

3.2 실시간 3차원 공개 게임 엔진 비교 분석

본 논문에서 소개한 실시간 3 차원 공개 게임 엔진들을 [표 1]요소들을 토대로 비교분석 하였다.

	Quake2	Genesis 3D	Crystal Space	Fly3D	Nebula Device
광원효과	yes	yes	yes	yes	
다이나믹 라이트		yes	yes	yes	
밉매핑	yes	yes			yes
culling system	BSP	BSP	BSP Octree Portal c-buffer	BSP PVS	
LOD		yes	yes	yes	yes
particle system	yes	yes		yes	yes
shader					yes
Bones animation		yes			yes
스크립트	C++		C++	C++	Tcl/Tk

[표 1] 실시간 3차원 공개 게임 엔진 비교 분석표

4. 결론 및 향후 전망

본 논문에서는 실시간 3 차원 공개 게임 엔진에 대해서 간단한 소개와 함께 게임 엔진으로서 중요한 요소를 기준으로 비교분석 하였다. 게임 엔진을 제작하거나 연구하는 학교 및 기관 연구소 혹은 개인들에게 잘 알려지지 않은 공개 엔진에 대한 내용을 언급함으로써 좀 더 많은 개발자들의 게임 엔진 연구에 많은 도움이 되었으면 한다. 3 차원 게임 엔진을 이용한 다양한 방면으로의 접근이 시도될 것이라고 예상이 된다. 특히 캐릭터에 대한 사실적인 표현에 바탕을 둔 애니메이션 엔진과 디지털 방송매체에 큰 영향을 줄 것이고 실시간 지형 렌더링을 통한 지형 좌표 시스템에도 영향을 줄 것이 예상된다.

이후에는 공개 게임 엔진과 상용화 엔진에 대한 비교분석대상을 좀더 세분화하여 자료를 정리할 예정이다.

참고문헌

[1] Tomas Moller, Eric Haines, " Real-Time Rendering" , A K Peters.ltd, 1999
[2] Alan watt, " 3D Computer Graphics" , Addison wesley, 1999
[3] Andre Lamothe, " Black Art of 3d Game Programming" , Waite Group, 1995
[4] Mark Deloura, " Game Programming Gems" , Charles River Media
[5] http://spica.u-aizu.ac.jp/soe/magazine/soe9.html
[6]http://home.san.rr.com/thereindels/Mapping/Mapping.html
[7] Quake2, 2001 id software, 2001, http://www.quake.com
[8] Geforce 2, 2001 NVIDIA Corporation, 2001, http://www.nvidia.com
[9] Genesis3D, Eclipse Entertainment, 1998, http://www.genesis3d.com
[10] Crystal Space, http://crystal.sourceforge.net
[11] Fly3d, http://www.fly3d.com.br
[12] Robot Dynamics And Control, Mark W.Spong, M. Vidyasagar, Wie Wiley, 1989
[13] Nebula Device, Radon Labs, 2002, http://www.radonlabs.de