국내의 온라인게임개발 수준이 세계적으로 인정을 받고 있으며, 정부는 물론이고 게임개발에 관련된 여러 분야에서 온라인게임개발에 대한 중요성과 연구개발 필요성에 대해서 계속해서 강조되고 있다. 그러나,..... 현재 국내의 게임개발 업체의 대부분이 게임개발 인력이 부족한 상황에서 게임엔진개발과 게임 컨텐츠 개발을 동시에 수행해야 한다는 부담을 안고 있다.

국내 온라인 3D 게임 엔진의 현황

김경식(호서대), 김현빈(ETRI), 유채곤(대덕대), 임충재(동서대),
이만재(아주대), 최성(남서울대), 최학현(동아방송대)

요 약

본 논문에서는 국내에서 접근 가능한 몇가지 온라인 3D 게임 엔진에 대해서 조사 비교하였다. 이러한 비교분석은 국제적인 경쟁력을 가지는 국산 3D온라인 게임엔진개발이라는 과제의 기초자료로 사용될 수 있을 것이다.

I. 서론

국내 외 게임개발업체들의 관심이 온통 온라인게임으로 집중되고 있다. 이미 그 중요성과 발전가능성에 대해서는 널리 알려져 있었지만, 수익성문제와 기술력부재로 인해 도전분야로 남아있었던 것이 사실이다. 그러나, 국내에서도 연간 매출이 1000억 원이 넘는 온라인 게임이 등장했을 뿐만 아니라 시간이 갈수록 온라인게임을 즐기는 사용자가 늘어나고 있는 경향이다. 이에 국내의 수많은 게임개발업체들이 일반적인 패키지게임을 버리고 온라인게임 개발 쪽으로 개발방향을 선회하고 있다. 또한, 게임을 즐기는 사용자들의 요구가 기존의 2D를 기반으로 하는 게임에서 3D를 기반으로 하는 게임 쪽으로 옮겨가고 있기 때문에 현재 개발되고 있는 게임 중에서 3D를 기반으로 하는 게임이 상당한 비중을 차지하고 있다.

게임을 개발하는데 있어서 게임 엔진이라는 것은 게임 프로그래밍[1-3] 의 핵심코드로서 재사용성이 높은 화면 그리기, 기본 인터페이스 등의 코드를 말한다. 게임의 내용에 따라 물리적인 특성과 인공 지능적 코드를 포함하기도 한다 [4-12].

국내의 온라인게임개발 수준이 세계적으로 인정을 받고 있으며, 정부는 물론이고 게임개발에 관련된 여러 분야에서 온라인게임개발에 대한 중요성과 연구개발 필요성에 대해서 계속해서 강조되고 있다. 그러나, 현실적으로 국내 온라인게임개발 업체들이 상당히 많은 문제점을 안고 있다. 외국산 게임 엔진을 사용할 경우 고가의 라이센스비와 판매 시 카피 당 로열티를 지불해야 하며 이 엔진 자체도 옛 것이므로 비슷한 아류의 게임을 만들 뿐이라 경쟁력이 약하다. 따라서 현재 국내의 게임개발 업체의 대부분이 게임개발 인력이 부족한 상황에서 게임엔진개발과 게임 컨텐츠 개발을 동시에 수행해야 한다는 부담을 안고 있다.

이러한 시점에서 국내. 외에서 사용되고 있는 3D온라인 게임 엔진에 대한 비교분석은 국제적인 경쟁력을 가지는 국산 3D온라인 게임엔진개발이라는 과제

의 기초자료로 사용될 수 있을 것이다. 현재 국내.외의 이 분야의 연구동향을 살펴보면 다음과 같다.

국외 연구 동향

게임선진국의 개발 업체들의 경우를 살펴보면 게임개발을 위해 엔진을 새로 제작하는 비효율적인 노력을 줄이기 위해 자체 기술 연구를 위한 연구소를 운영하고 있으며 지속적인 연구를 통해 게임 엔진기술들을 보유하고 전문화 시켜가고 있다. 또한 중소규모의 개발업체들은 이러한 연구소를 통해 게임엔진 기술력을 도입하여 게임개발을 하고 있으므로 수준 높은 게임컨텐츠를 제작할 수 있으며 세계시장을 장악할 수 있는 바탕이 되고 있다.

국내 연구 동향

현재 RealTime Computer Graphic 3D Rendering 엔진의 경우 국내에서 자체개발 기술을 보유하고 있기는 하지만 폐쇄적인 개발 구조로 인하여 국내기술의 전반적인 발전에 가장 큰 저해 요소가 되고 있으며 몇몇 개발회사를 제외한 중소개발 업체들은 대부분 많은 시행착오와 함께 3D 엔진과 온라인 게임에 필요한 서버기술력 보유를 위해 과중한 인력투입은 물론이고 자금투입 또한 상당한 비중을 두고 개발이 진행되고 있는 상태이다. 때문에 직접적으로 사용자들에게 전달되는 게임성 내지는 게임요소 구현보다는 게임엔진 개발에 더 많은 노력을 쏟고 있으며, 필연적으로 개발되는 게임의 질적 수준을 향상시키는데 상당한 걸림돌이 되고 있는 현실이다.

2. 온라인 3D 게임엔진 현황

1. 국내엔진

현재 온라인 3D 게임엔진을 보유하거나 개발하고 있는 국내 업체들이 많지만 공개를 허락한 대표적인 2개의 엔진을 소개한다.

가. 가이블 3D 엔진

(1) 특징

가이블 3D 엔진의 가장 두드러진 특징은 웹 3D 동영상지원이 우수하다는 것이다. 실시간 3D렌더링 엔진의 대표격인 Quake와 비료를 해보면 Linear 렌더링 방식을 채택하고 있는 Quake엔진에 비해 Multipath-shading 렌더링 방식을 채택하여 3D렌더링 성능에 있어 우수하다.

그리고 기존의 BSP방식을 사용하고 있는 Quake에서는 Static 공간분할의 특성상 그림자의 이동이 어렵고 벽을 부수는 등의 표현은 어려운 제약이 있는 반면에 가이블 엔진에서는 Dynamic 공간분할을 사용하며 그러한 단점을 보완하고 있다. 3D Modeling 도구에서의 Data추출 면에서도 월등한 성능을 보여주고 있다. 3D MAX의 예를 들어보면 최소의 data만을 추출하고 나머지 부분에 대한 처리는 엔진에서 담당하기 때문에 light, meterial 등 질감 표현에서 훨씬 빠른 속도를 자랑한다. 또한, 엔진 자체의 프로그래밍 부분에 있어서도 wrapping, class hierarchy 등 C++ 사용기법이 Quake엔진에 비해 우수한 특징을 갖고 있다.

(2) 주요 기능

가이블 3D 엔진을 modeling, animating, rendering, special effects, bitmap분야로 나누어서 설명하고자 한다.

① Modeling

· Polygon modeling : vertex들이 모여 삼각형 구조를 갖는 다각형을 페이스(face)라고 하는데 이 face가 모여 이루는 다각형을 polygon이라고 하

고 , 이 다각형 을 중심으로 만드는 mesh · Patch modling : 패치(patch)라는 옷감 종이 성질을 가진 오브젝트 하나로, 만들고자 하는 면의 굴곡을 만든 후 , 모서리에 새로운 패치들을 이어서 오브젝트를 만듦, 넙스(nurbs)의 강점을 이어받고 폴리곤(polygon)의 투박함을 없앤 최강의 모델링 기법

- In-out mesh modeling : mesh를 face, edge, vertex를 추가하거나 , 이동 또는 face를 선택하여 extrude 하여 모델링
- Subdivision Surface modeling : 폴리곤 모델링을 한 후 넙스의 곡선처럼 부드럽게 만들어 주는 모델링
- Terrain modeling : 2차원 쉐이프 오브젝트를 연결하여 3차원 지형을 만듦
- Character modeling : 사람이나 동물 또는 어떤 특정 캐릭터를 넙스(nurbs), 폴리곤(polygon), 패치(patch)와 같은 모델링 방식으로 만들어 내는 작업
- Low polygon modeling : 폴리곤의 수를 적게 사용하여 모델링 한느 방식. 기본적으로 사람과 같은 캐릭터는 600개 ~ 800개 정도의 폴리곤을 써서 모델링
- Mesh optimization : Realtime3D에서 쓰이는 mesh는 폴리곤 수를 적게 제작함. 그래서 mesh를 모델링할 때 보통은 low polygon medeling을 함. 모델을 최적화하고 필요 없는 폴리곤은 없애주어야 함. 이렇게 모델링 마지막에는 mesh의 최적화

② Animating

- Nor-Linear animation : 키프레임과 키프레임사이의 장면(Scene)들을 Aniamation 할 때 키프레임(Key Frame)과 키프레임(Key Frame)사이의 프레임에 대해서 선형적으로 변화시키는 것이 Linear Animation이고 2차내지 3차식-베지어곡선(Bezier Spline)이나 TCB곡선으로 변화시키는 것
- Motion tracking : scene에 존재하는 모든 오브젝트의 내니메이션 키 값을 수정 편집할 수 있는 기능
- forward kinematics:관절구조물(Articulated Structure)의 모든 관절(joint) 각(angle)이 주어진 상황에서 관절 구조물의 끝부분(end effector)의 위치를 계산하는 절차(Process)
- Inverse kinematics : 관절 구조물의 끝부분의 위치가 주어진 상태에서 모든 관절의 각 계산
- Facial espression : 얼굴표정 애니메이션. 캐릭터의 립 싱크(Lip Sync)애니매이션, 표정 애니메이션이 들어감. 3dsmax 에서는 morpher를 이용하여 표정과 캐릭터의 립싱크 애니메이션을 제작
- Motion capture : 모션 캡쳐는 가상캐릭터(Virtual Character, 혹은 Virtual Actor , Virtual talent)와 실제 연기자가 필요하며, 실제 연기자의 몸 여러 부분에 센서를 부착하고 그 동작을 데이터화하여 가상 캐릭터(3D 모델)가 같은 동작으로 애니메이션
- IK single chain handle : 본즈(bones)는 사람과 같이 관절을 갖는 오브젝트를 위해 뼈대를 심는 기능으로 이것은 움직이는 오브젝트를 보다 쉽게 애니메이션하기 위해 사용되는 기능. 이본을 보다 쉽게 애니메이션 시키기 위해 사용되는 것이 IK single chain handle controller
- Animation interpolation : 애니메이션의 키 값들을 보간하여 부드럽게 연결시켜주는 알고리즘
- Keyframe animation : 데이터 용량을 줄이기 위해 각 애니메이션의 필요한 키 값 생성

해 각 애니메이션의 필요한 키 값 생성

- Time-based animation : 시간을 동기화하여 mp3나 음성 스트리밍에 맞춰 애니메이션
- Transform animation : Trnaslate, Roatate, Scale 애니메이션을 보간방법을 이용하여 애니메이션
- Deformation bones : 3DS MAX에서 구현된 bone, Scale 애니메이션을 보간방법을 이용하여 애니메이션
- Skinning : Bone과 biped를 심은 오브젝트를 관절의 끊김없이 부드럽게 애니메이션
- Face morping : 얼굴의 움직임(입, 눈꺼풀, 토 등)이 자연스럽게 움직이도록 하며 립싱크 가능
- Real shadow : 그림자를 사실적으로 표현
- Dynamics : 사실감을 부여하기 위한 운동 역학 적용

③ Rendering

- Flat shading : 3D 상에 존재하는 Object의 면의 밝기(Intensity)를 부여하는데 있어서, 한 면 내부늬 모든 점들에 대해 같은 밝기를 주는 기능. Object 면의 Normal Vector의 Light Source와 Object 면위의 점으로 이루어지는 벡터사이의 내적을 이용해서 두 벡터가 마주보고 있는 각도, 즉 zero면 가장 밝은 부분, 두 벡터사이의 각이 직각이면 가장 어두운 부분. 여기서 Flat이란 말이 붙여진 이유는 한 면에 있는 점들에 대해서는 같은 정도의 밝기가 부여되어 평평하게 부임
- Gourand shading : Object를 구성하는 면 내부의 모든 점들에 대해 각각 다른 세기의 밝기를 부여하여 보다 실감나도록 보이게 하는 방법. 각 면을 구성하는 Vertex들의 Normal Vector를 산술 평균하여 얻어낸 후 Flat Shading에서와 같은 방법으로 빛의 세기를 부여하여 빛의 세기 값으로 보간(Interpolation)해서 각각 다른 세기의 밝기를 줌
- Phong shading : Normal Vector값을 가지고 면 내부의 점에 대한 Normal Vector값을 얻어냄 Metal rendering 메탈 표면에 광원의 반사나 물체의 반사를 보여줌
- Texture mapping : 2D 이미지를 3D 모델의 표면에 입힘
- Bump mapping : 알파 채널을 사용하여 질감을 사실적으로 표현
- Alpha transparency : 텍스처 맵의 알파 채널로 물체를 투명 또는 반투명으로 보여줌
- Toon shader : 모델의 외선을 실루엣 아웃라인으로 렌더링하는 기법. 3D 모델을 만화적으로 표현
- Face Material : 오브젝트에서 특정면에서 대한 색감이나 텍스처의 속성을 각각 다르게 할당
- UV texture mapping : 3DS MAX의 UV texture 코드 좌표를 읽어 텍스처를 모델에 입는 기술
- Light mapping : 광원의 위치와 광원에 영향받는 3D객체의 빛의 양을 계산하여 텍스처 맵으로 제작하여 모델에 입힘
- Dynamic lighting : 광원들을 생성하고 그에 따른 실시간 계산으로 물체들에 입체감을 줌
- Vertex smoothing : 표면에 vertex의 normal값을 연산하여 표면을 부드럽거나 평평하게 보여줌
- Level of detail : 카메라가 멀리 있는 물체는 폴리곤이 적은 오브젝트로 가까운 물체는 많게 하는 기법
- Mipmap : level of detal 기법으로 택스처의 jitter를 없애주는 것
- Volume Fog : 사실적인 표현을 나타내기 위해 서로 다른 크기의 노이즈를 갖는 안개 모양 표현

④ Special Effects

· Particle system : 눈, 비, 폭파 등을 표현하기 위해 작은 크기의 입자 오브젝트를 구성하는 시스템 제작

· Deformation : 만들어진 dnosfo의 오브젝트에 scale, twist, teeter, bevel 등의 변형을 가함

· Image processing : 화면에 출력된 영상에 대해 색 표현(color representation), 포인트(point processing), 영역(area processing), 기하학(Topology processing) 등을 처리

⑤ Bit Map

· Anti-aliasing : 보간 알고리즘으로 산을 부드럽게 표현

· Dithering : 새로운 색상을 주기 위해 칼라 픽셀의 배열 변환

· Alpha brending : 초기 픽셀의 색상을 최종 픽셀의 기존값에 의거하여 알파 채널을 부가하는 방식으로 혼합하는 기법

⑥ 현재개발중인 기술

· BSP/PVS render management : 공간 분할, 한 공간 안에 있는 것을 다 계산하여 속도가 느려지기 때문에 눈에 보이는 것만 계산하여 보여줌

· Light Maps for static geometry(pre-computed lighting with soft shadow) : 속도 향상을 위해서 실제 라이트를 사용하는 것이 아니라 라이트 맵이라는 텍스처로 사용하여 표현

· Vomuletric fog with fog maps : 안개 효과, 화면 전체에 군데군데 뭉쳐져 있는 부피가 있는 안개

· Collision detection : 충돌체크 ,서로 부딪히는 관계를 정의 예를 들면, 캐릭터가 거리를 걷다가 나무가 앞에 있는 경우 부딪히지 않고 가도록 계산

· Dynamic shadows : lightmap or stencil shadow volumes, 물체에 그림자를 주고 그에 따른 실시간 계산으로 물체들에 입체감을 줌

· Physics Engine : 물리학 엔진, 공을 벽에 던진다면 던졌을 때의 힘, 공이 부딪힌 각도 , 방향 등을 계산하여 떨어졌을 때 공이 떨어지는 방향, 찌그러지는 정도를 나타냄

· Mathematic Engine : 수학 엔진, 원심력, 중력가속도, 탄성계수 등의 물리학적 요소들을 수학적으로 계산하여 처리

· 인공지능(AI) : 하나의 캐릭터가 움직일 때 다른 캐릭터가 따라 다니는 것 또는 미로 찾아내기 등 캐릭터의 움직임을 인공지능으로 처리하도록 개발

· 디바이스 컨트롤링(컨트롤 함수) : 마우스, 키보드, 조이스틱 등 입력 매체 제어 함수 개발

· 그밖에 네트워크 엔진(서버 포함), 스크립트 언어, 사 운드 개발(3D 사운드)

나. MU 엔진

7억원을 들여 개발하여 지난 5월 25일부터 시범서비스에 들어간 뮤온라인 풀 3D게임은 이제까지 나온 온라인 게임들이 대부분 그림자 정도가 표현되는 3D의 초기였던 반해 처음부터 자체 개발한 3D그래픽 엔진을 사용하고 있다. 시범서비스 한지 두 달이 채 안되어 20만 명이 넘게 회원에 가입했다. 동시접속 자는 1만4000명으로 항상 서버 한계가 꽉 차 있을 정도로 폭주중이다. 리니지의 경우 접속자수가 가입회원에 100/1정도임을 고려할 때 뮤의 회원대비 동시 접속자 수는 대단한 수치이다. 이 게임에서 갑옷, 무기, 방어구 등의 아이템이 10만여 개나 나온다. 이를 조합하여 자신만의 독특한 게임 캐릭터를 꾸밀 수 있다. 또한 풀3D인 만큼 여러 각도에서 무기를 볼 수 있어 현실감이 강하다. 기존의 캐릭터에 비해 크기가 3~4배 정

도 큰 것도 장점이다.

(1) 장점

① 행동

· 캐릭터의 자유도(앉기, 인사, 약올리기, 브라보등 다양 한 행동가능)를 높였고 캐릭터가 마우스의 반응에 따라 해당 위치를 보도록 했다.
· 종족마다 각각의 개성 있는 기술 구사와 함께 10만 가지의 아이템 조합으로 다양한 캐릭터조합이 가능하다.
· 간단한 인터페이스와 수요와 공급에 따라 게임의 상거래 물가도 변화한다.
· 전문게이머를 위한 핫키도 지원하며 음성도 지원한다.
· 몬스터들에게 인공지능을 탑재하여 일격 다인 공격, 순간이동, 마법구사, 무리생활, 3D높이 개념을 살린 공중과 지상을 오가는 몬스터등 다양한 설정가능

② Graphic

· 풀3D로 렌더링(2D보다 훨씬 입체적이고 현실감 있게 유저들에게 다가간다.)
· 2D방식의 경우 갑옷이나 아이템 장착시의 경우 메모리가 많이 소모되지만 3D의 경우에는 그렇지가 않다.
· 아바타를 구현할 때도 2D의 경우 여러 가지 제약이 있어서 자유롭게 구현하기 어렵다. 하지만 3D의 경우 입체감과 현실감을 비롯 많은 점에서 2D보다 현실적으로 표현할 수 있다.

③ System

· 효과가 입증된 방식은 그대로 가져가면서 보다 쉬운 인터페이스로 가는 방식채택
· 이모티콘의 경우 게임 안에서 자기 아바타와의 일체감을 더욱 강조하기 위한 수단으로 사용 속도감 증대(공격시 딜레이 최소화, 빠른 이동속도, 단순 공격이 아닌 찌르기부터 올려치기까지의 다양한 공격 모드 지원)

(2) 단점

① 서버안정화

② 데이터손실의 최소화 방안

2. 국외엔진

가. GENESIS3D 엔진

차세대 그래픽 카드(Transform and Lighting, TnL)가 나오기 전까지 넓고 복잡한 지형을 연출하기 위해 많이 사용되었던 BSP기법을 기반으로 하는 게임 엔진은 퀘이크(Quake)를 비롯하여 큰 인기를 얻어왔다. 그러나 TnL그래픽 카드의 출현으로 BSP기법만으로는 더 이상 최적화된 엔진으로 남기가 어려워진 게 현실이다. 하지만 이 Genesis3D엔진의 큰 의미는 게임 엔진을 제작하는데 아주 좋은 템플릿(Template)역할을 할 수 있다는 것이다.

전반적으로 지형, 캐릭터, 그래픽 드라이버, 동적 월드 객체 그리고 많은 자원관리 등등 3D 엔진을 제작하는데 있어서 좋은 참고 자료가 될 것이다. 그리고 Genesis3D의 엔진 소스를 활용한 여러 가지 툴을 분석함으로써 앞으로 게임제작에 필요한 툴을 제작할 때 좋은 예가 될 것으로 기대한다.

(1) 주요 기능 및 특징

① 지원 기능

· 라디오시티 광원(Radiosity lighting)
· 월드 객체 표현을 위한 물리학 지원
· 광원 애니메이션을 위한 광도의 선행 계산 연산

과 부식효과 지원
- 월드 렌더링의 가시성 테스트를 위한 빠른 이진 공간 분할 기법 사용 (Binary Space Partition, BSP)
- 동적 RGB광원효과
- 동적 그림자효과
- 동적 안개효과
- 동적 거울효과
- 동적 물 효과
- 동적 애니메이션 텍스쳐 제공(합성, 모핑, 등)
- 월드표현을위해 부분적으로 포탈영역(Portal area)제공
- 구형으로 매핑 된 하늘과 수평선 제공
- 3D 사운드 제공
- 사용자 확장가능한 파티클시스템(Particle system)제공

② MAP Editor
- 삼차원 환경에 최적화된 지형 생성
- 애니메이션 월드 객체 생성을 위한 키 프레임 시스템(Keyframe system)포함
- 물리적으로 제어되는 객체들 간의 상호작용을 위한 물리 객체 생성

③ Character
- 부드러운 스킨 캐릭터
- 최적화 성능을 위한 자동 가시성 테스트
- 3D Studio MAX에서 제공하는 캐릭터와 애니메이션툴 지원
- 높은 질의 음영처리를 위한 Smoothing group 지원
- 무제한의 텍스쳐와 색상 지원
- 다양한 재질 애니메이션(색조, 애니메이션, 합성 그리고 모핑)

④ Character Animation
- 계층적 또는 비 계층적 본(Bone) 애니메이션 지원
- 서로 다른 캐릭터간의 애니메이션 데이터 공유
- 애니메이션 정보의 부분 또는 전체적인 합성 기능
- 최소화된 애니메이션 데이터

⑤ Driver 지원
- Glide
- Direct3D
- Software driver(하드웨어 가속기 없는 시스템을 위한)
- AMD optimized software driver

(2) TOOL 구성

① Editor
- 3D 지형 편집
- 텍스쳐 매핑
- 광원
- 레벨 데이터 관리
- 텍스쳐 관리

② 엑터 스튜디오(Actor studio)

3D Studio MAX의 파일을 Genesis3D 파일로 변환시키는 역할을 한다.

③ 엑터 빌더(Actor builder)

여러 애니메이션 정보들을 하나로 통합한 Actor파일을 생성하기 위한 커맨드 명령을 제공한다.

④ 엑터 뷰어(Actor viewer)

Actor Builder에서 만들어진 Actor파일을 보여주는 프로그램으로 Actor안에 들어있는 모션들을 볼 수 있게 해준다.

⑤ 텍스쳐 (Texture packer)

게임에서 사용되는 텍스쳐들을 하나로 통합관리 해주며, 추가/삭제가 간편하다.

나. UNREAL 엔진

현재 존재하는 엔진 중에 가장 훌륭한 엔진이라는 평을 듣고 있는 UNREAL 엔진의 주요 기능 및 특징을 살펴 보고자 한다.

(1) 주요 기능 및 특징

① Editing Tools

UNREAL은 매우 강력한 실시간 level design 툴을 기초로 3차원 배경을 디자인한다. 이러한 UnrealEd는 UNREAL 엔진에 완전히 매칭되어서 이 툴에서 디자인한 모습 그대로, 게임에서 보이는 환경과 디자인을 얻을 수 있다. 예를 들어 빛과, Texture, 그리고, 지형적 구조, 효과 등의 모든 것을 레벨을 만드는 순간 바로 볼 수 있게 디자인되었다.

UnrealEd에서는 매우 쉽게 게임을 돌려서 만들어 놓은 레벨을 테스트 해 볼 수 있다. 또한 Texture browers는 texture를 선택하고, 교체하는데 사용되고, Surface properties, surface의 위치, 크기, 그리고 특수효과를 수정할 수 있으며, Actor properties는 actor를 Visual Basic tool처럼, 작성 할 수 있다.

② Mesh Animation

UNREAL에 사용되는 모든 Mesh Animation은 3D STUDIO MAX, Soft Image, Power Animator또는 모션 캡쳐로부터 얻은 애니메이션을 .DXF로 import 하여 사용한다. Mesh Animation으로 높은 프레임 비율로 Fluid Animation이 지원된다. 15fps면 게임이 가능하게 되고 40fps가 되면, 부드러운 게임을 할 수 있다. 또한 tweening 기능을 지원하여, 플레이어와 몬스터의 액션 사이의 흐름을 부드럽게 연결시키는 애니메이션이 가능하다.

③ Artificial Intelligence

매우 발전된 수준의 creature나 bots의 Artificial intelligence를 작성할 수 있게 스크립트 언어가 설계되어있다. bots는 모든 움직임과, 무기, 그리고 인벤토리를 어떻게 이용하는지 알고 있다. 즉 A-Life로서 AI를 작성할 수 있게 되어있다. UnrealEd를 살펴보면, 객체 지향적인 스크립트의 언어를 이용하여, AI를 보다 확장성 있게, 그리고 맞춤식 인공지능을 작성 할 수 있게 되었다. 또한 Path node를 기반으로 한 navigation 시스템은 보다 복잡한 길을 찾는데, CPU 시간을 최소화하여 찾을 수 있게 한다.[7-12, 14]

④ Digital Sound System

3D 위치와, 거리, 스테레오, 도플러효과를 고려한 음을 지원하게 되어있다. 돌비 서라운드 시스템을 지원하며, 그렇게 된다면, 360도로 펼쳐진 채널에서 소리를 들을 수 있다. 또한 UNREAL의 이러한 Digital Surround System은 완전히 UnrealScript에 의해 조정될 수 있게 되어있어 Sound 설계자가 쉽게 사운드를 만들고, 수정할 수 있다.

⑤ Digital Music System

CD audio와 실시간 디지털 믹스 음악을 지원한다. 게임 디자이너는 높은 수준의 음질과 속도를 자랑 할 수 있는 CD 음악과, 유동적인 음악과, 인터넷을 통한

이동가능성을 장점으로 하는 디지털 믹스 음악을 선택하여, soundtrack으로 사용할 수 있다. .mod, .s3m, .xm등과 같은 유형을 포맷을 지원하며, 우리가 알고 있는 대부분의 포맷을 지원한다.

⑥ Lighting

멀티컬러 라이팅를 지원하게 되며, raytrace와 envelop 라이팅이 지원된다. 라이팅 역시 UnrealEd에서 수정하여 바로 볼 수 있기 때문에, 쉽게 수정할 수 있다. 이러한 라이팅 효과는 플레어나 코로나 현상 등을 지원하여 보다 멋진 특수효과를 구현하는데 도움을 준다.

⑦ Game Programming

게임 프로그램은 크게 C++와 UnrealScript 프로그램으로 나누어진다. 모든 프로그램은 객체지향적설계로 게임의 컨셉(player, monster, inventory, triggers)에 맞게 작성된다. UnrealScript는 state와 state수준의 함수들과, 시간에 기초한 latent 함수와, networking을 지원한다. 이 스크립트 언어에서 높은 수준의 AI, 길찾기, 그리고 Navigation System을 프로그래밍한다.[3]

⑧ Networking

UNREAL은 지금까지 나온 게임들 중 가장 발전된 모습의 네트워크를 지원한다. 매우 작은 수준의 LAN 게임과, 큰 스케일의 서버에 기초한 인터넷 게임도 지원한다. 게이머들은 UNREAL 서버들을 돌아다니며 쉽게 접속할 수 있다. 또한 새로운 레벨이나 텍스쳐, 사운드, 모델, 스크립트 등의 새로운 컨텐트 들은 자동으로 다운로드 받아서, UNREAL 세계로 불러들여 사용한다. 자바스타일의 클라이언트 사이드 스크립트 언어로, 최소한 28.8K 모뎀의 속도로도, 게임을 무난히 즐길 수 있다.

지금까지 UNREAL 엔진에 대해서 간략하게 살펴보았다. UNREAL 엔진은 많은 부분에서 볼 때, 현재 최고의 엔진을 자랑한다. 뛰어난 그래픽과, 빠른 네트워크, 그리고, 고 수준의 인공지능을 구현 할 수 있는 스크립트엔진 등의 면만 보더라도, UNREAL은 현재 최고의 3D엔진 중 하나이다. 그에 더해 하드코딩을 최소화하고, 객체 지향 설계에 충실하여, 확장성을 높이고 재사용성을 늘려, 보다 좋은 게임을 개발하고자 할 때 좋은 본보기가 된다.

다. Turbine 엔진

(1) 개요

Turbine 엔진은 마이크로소프트에 의해 출판된 Asheron's Call이라는 MMORPG 게임에 사용된 엔진으로 Turbine Entertainment 사에 의해 개발되었다.(그림 1) 아직은 Asheron's Call 이외의 다른 게임에 사용된 예는 없다. 최근 버전을 2.0으로 개선하였다. 실제 서비스를 진행한 몇 안되는 MMORPG 엔진으로 3D 그래픽 기술과 컨텐츠 제작 환경을 함께 지원한다. 서버는 분산Fault-tolerant 시스템으로 관리자의 개입을 최소화하도록 구현하였다. 또한 World 컨텐츠를 구현하기 위한 시스템을 엔진 내부에 포함하고 있다.

(2) 서버 기술

엔진의 특징은 매우 큰 region을 경계면이 없이 하나로 구현하였다는 것이다. 예를 들어 Asheron's call에 등장하는 Dereth라는 섬은 마을, 산, 호수, 개울, 다리, 폭포 등을 포함한 500 평방마일 크기를 하나로 다루고 있다. 이러한 크기의 World 를 구현하기 위해 분사 서버를 사용한다. 이보다 더 큰 region을 넘는 경우에는 별도의 서버에서 다루게 되며 수 초 정도의 이

동시간을 필요로 한다. 2.0 버전의 경우 1.0 버전의 경험을 살려 개정한 코드로 매우 안정적이라고 알려 져 있다.

(3) 클라이언트의 그래픽 기술

클라이언트는 사용자에게 최선의 게임의 느낌을 주도록 한다. 따라서 인터넷을 사용함에 따른 시간지연을 최소화하도록 하고 있다. 또한 최첨단의 3D 기술을 사용하여 사용자의 시각적인 만족을 높이도록 한다. DDD(Dynamic Data Downloading) 기술을 사용하여 온라인 World에 변화가 있을 경우 이를 자동적으로 업데이트하는 기술을 사용한다. 사용자 인터페이스는 스크립트를 사용하여 프로그래머 아닌 개발자도 쉽게 Look and Feel을 변경할 수 있도록 하고 있다. Turbine 2.0 엔진은 Direct 3D 8.0을 기준으로 새로 코딩하였으며 하드웨어 T&L, multi-texturing, vertex blending과 같은 첨단의 기술을 사용한다. 사용자 플랫폼의 성능 차이가 많은 것을 감안하여 모든 애니메이션이나 multi-resolution mesh와 같은 3D component를 scalable 하도록 설계하였다. 애니메이션 시스템 역시 skeletal animation을 사용하며 시스템의 성능이 낮을 경우를 대비하였다. RPG 게임에서는 avatar의 외모가 게임 진행에 아주 중요한 의미를 갖기 때문에 다양한 칼라와 모습을 택할 수 있도록 하였다.[13, 14]

(4) Dynamic Data Downloading

MMORPG에서 DDD는 매우 중요한 위치를 차지한다. 새로운 creature나 NPC가 world에 추가될 경우 텍스쳐, 모델, 애니메이션, 사운드, 게임 규칙을 새로 클라이언트에 전달하여야 한다. 대부분의 온라인 환경은 별도의 접속시간에 patch를 받아서 처리한다. patch를 사용하는 방법은 오랜 공백기간 후에 참여하는 사용자에게 수 Mbyte 이상의 다운로드를 요구하게 되며 사용자를 불편하게 한다. Turbine 엔진은 DDD 기법을 표준 patching 방법으로 사용한다.

Asheron's call 게임에서는 이벤트를 만들어 world에 대규모의 수정을 가한다[6]. 이러한 world의 지속적인 발전으로 한 번 참여한 사용자는 계속 온라인 게임에 참여하게 된다. 이러한 이벤트를 통하여 게임의 줄거리를 진행시키고 한편으로 게임 밸런스를 개선하기도 한다.

또한 새로운 지역을 추가하여 새로운 탐험을 유도한다. DDD를 사용하기 때문에 비록 아직은 CD를 사용하여 게임을 판매하지만 최소한의 다운로드 또는 타 게임 CD의 번들로부터 게임을 시작할 수 있다. DDD에서는 꼭 필요한 데이터만을 게임 진행의 배후에서 다운로드하도록 한다.

(5) 게임 아트 저작도구

3D 모델, 애니메이션, 텍스쳐와 같은 게임 아트는 Maya, Photoshop과 같은 도구를 사용하여 제작한다. plug-in 기능을 사용하여 컨텐츠 제작에 필요한 기능을 추가하였다. 예를 들어 Maya작업을 하면서 게임 엔진에서 어떻게 렌더링 되는지를 볼 수 있도록 하고 있다. 3D 모델과 애니메이션은 Maya를 사용하나 게임아트는 대부분 Light Wave를 사용하여 처리한다. 필요한 경우 3D max를 사용할 수도 있다.

(6) 컨텐츠 제작 및 World 규칙

Turbine 엔진은 정교한 수준의 이동 및 충돌 확인 시스템을 엔진에 모듈로 포함할 수 있도록 한다. 이러한 기능을 게임 디자이너가 요구하는 데 따라 변경할 수 있는 구조를 택하였다. 엔진은 전반적으로 모듈 구조를 택했기 때문에 서버, 데이터베이스, 충돌, 네트웍 모두 정의된 API를 사용하도록 한다. 이러한 모듈 사

용은 코어 엔진의 구현과 상관 없이 게임을 개발할 수 있도록 한다. 새로운 게임 룰이 필요할 경우 상대적으로 적은 시간에 구현이 가능하다.

새로운 게임 시스템을 구현하기 위해 자체의 인터프리터 형식의 스크립트 언어를 사용한다. 이 언어는 단순한 문법과 객체지향 기능을 사용하고 있으며 네트웍메시지 처리, data serialization 등을 처리하며 기능을 확장할 수 있다. 서버와 클라이언트에서 수행되는 이 스크립트 언어는 최첨단의 게임 개발에 필요한 복잡도를 대폭적으로 낮추는 데 크게 기여하고 있다.

새로운 컨텐츠를 제작하고 이를 게임 world에 업데이트 하는 작업은 온라인 게임에서 가장 많은 시간을 필요한 기능으로 이를위해 그래픽 작업환경을 제공한다.

(7) 정리

Turbine 엔진을 사용한 Asheron's Call은 이미 약 2년의 서비스 경험을 갖는 몇 안되는 MMORPG 3D 게임 엔진이다.[15,16] 여기에서 사용된 DDD 기술, 스크립트언어 기술, 컨텐츠 제작 및 관리 기술을 본 프로젝트에 도움을 줄 수 있을 것이나 구체적인 내용을 알 수 없어 아쉬움이 남는다.

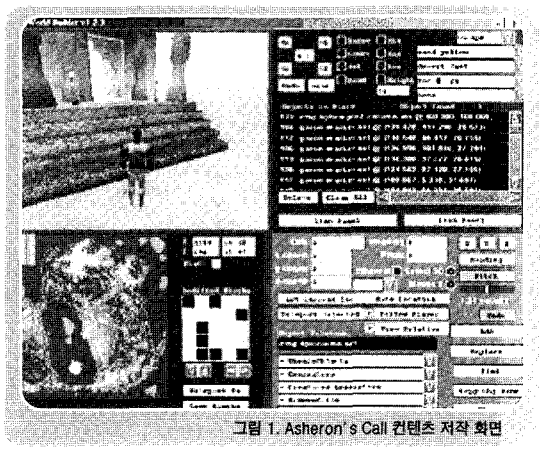

그림 1. Asheron's Call 컨텐츠 저작 화면

라. Quake Engine 분석

(1) 주요 특징

- OpenGL 기반으로 제작되었고 리눅스등 여러 플랫폼 에 이식.
- 제한적으로 공개된 소스로 변형된 퀘이크(MOD)를 만 들 수 있음.
- 풀 소스가 공개 되어있음 (ID 의 프로그래머인 존 칼 막이 크리스마스 선물로 공개했다.) ftp://ftp.cdrom.com/pub/idgames/idstuff/
- 크기 맵을 처리해주는 부분과 Texture mapping과 Lightmap을 함께 사용하여 맵 Mapping을 처리해주는 Portal
- 캐릭터 애니메이션 처리해주는 부분 캐릭터의 충돌을 효과적으로 체크해주는 부분
- 화려한 특수효과 및 물리학 엔진
- Network 부분에서 다른 플레이어와 게임을 진행중 리얼타임 효과적으로 처리하여 Network 상에서 게임 을 더욱 증가시킴
- 퀘이크 엔진에서 맵을 처리하기 위해 BSP 알고리즘을 사용하고 있는데, 순수한 BSP 알고리즘에서는 많은 제한을 가지고 있으나 퀘이크 엔진에서는 약간의 변형을 주어 이러한 제한을 극복 (고전 게임 중에서 1인칭 액션 게임인 DOOM에서는 문이 수직으로만 열리는 것을 알 수 있다. 즉 DOOM 에서는 순수 BSP 알고리즘만을 사용하여 맵을 처리하여 주었다. 하지만 퀘이크 에서는 위의 알고리즘을 변형하여 문을 좌우로 열리게 하였다.)
- 부분적인 안개 효과를 사용하여 지하층을 더욱 효과적 으로 표현

(2) 주요 기능

① Client Server

Quake는 그림 2에서 보는 바와 같이 클라이언트-서버 프로그램이다. 모든 게임플레이어와 시뮬레이션은 서버에서 실행되고, 모든 Input, Output은 클라이언트에서 일어난다. 즉, 클라이언트는 특별한 터미널일 뿐이다.

각 클라이언트는 각 프레임에 대한 키보드, 조이스틱, 마우스의 입력을 모아서 서버로 전송한다. 서버는 모든 클라이언트로부터의 입력을 받아서 정해진 시간(fixed timeslice) 동안 게임을 실행하고 결과를 클라이언트로 전송한다. 클라이언트는 다음 프레임동안 결과를 표시한다[17, 18].

클라이언트-서버 기술은, 여러 플레이어 사이의 동기 문제를 단순화시키므로 멀티 플레이어 게임에 있어서 명백한 이점을 가지고 있다. 클라이언트-서버는, 디자인을 모듈화할 수 있고 디버깅을 단순화 시켜주는 하나의 통신 채널을 사용할 수 있으므로 싱글-플레이어 모드인 경우에도 유용하다.

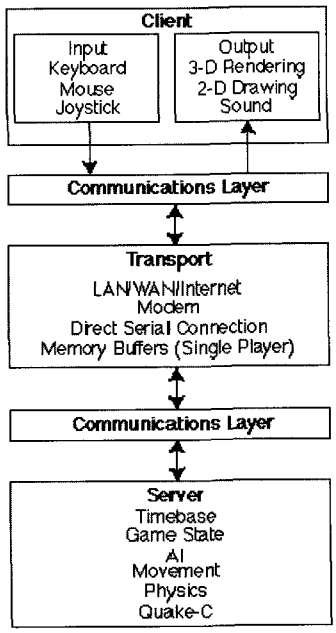

그림2. Quake의 구조

② 통신

클라이언트-서버 구조의 관심 있는 이슈(issue)는 인터넷 플레이이다. 퀘이크는 처음부터 멀티플레이 게임을 고려해서 설계되었다. 그 이전에는 커다란 주목을 받지 못한 인터넷 플레이가 많은 관심거리를 만들어 냈다. 통신 지연(communication latency)은 LAN이나 심지어는 직접 연결된 모뎀 플레이 경우 보다 심했고, 패킷 전송 역시 신뢰할 만한 정도가 아니었다. 개발 초기에, 퀘이크는 신뢰성 있는 패킷 전송 방법만을 사용하였다. 즉, 패킷이 전송되고 그에 대한 ack를 받는 방식이다. 만일 ack이 오지 않으면, 재전송을 하여 ack를 받을 때까지 중지된다. 전송되는 데이터의 사이즈를 줄이기 위해서, 현 상태 정보 자체가 아닌 변화된 내용만을 주고받았기 때문에 모든 데이터는 누락되는 패킷없이 반드시 전송되어야 했다. 만일 어느 한 데이터라도 놓치면 그 다음에 오는 데이터는 모두 소용없어지기 때문이다.

이런 패킷 전송의 문제점은, 패킷이 드롭되었을 경우, 드롭된 사실을 발견하는데 까지 긴 시간이 걸리고(최소한 클라이언트와 서버 사이 한 번의 roundtrip 시간), 그러므로써 결국 드롭된 패킷을 재전송하는 것 역시 시간이 오래 걸린다는 것이다.(최소한 또 한번의 roundtrip) 만일 클라이언트로의 ping time이 200ms라면(PPP 연결 상에서 최적의 시간임) 드롭된 패킷이 수 백ms 동안 멈춰 있게된다. - 그 시간이면 매우 긴 시간이다.

위와 같은 방법대신 현재, 퀘이크는 점수(score)나 레벨 변화와 같은 정보에 대해서만 reliable packet을 사용한다. 플레이어의 위치나 오브젝트와 같은 현재의 게임 상태는 그 정보 자체가 각 timeslice동안 전송된다. 즉, 변화값이 아니고 신뢰성 있는 전송 메커니즘을 이용하지 않는다. ack가 없고 서버나 클라이언트는 자신이 보낸 패킷이 제대로 도착했는지 알 수 없

다. 각 갱신 정보는 각 클라이언트에 대한 모든 정보를 담고 있다. 그러므로 패킷 하나를 읽어 버린다는 것은 한 번의 server time slice동안 세상이 freezing 되어 있음을 의미한다. 서버 타임 슬라이스는 1초에 10이나 20번 정도로 온다. 그러므로 드롭된 패킷은 단지 100ms 동안만 지연시키는 결과를 가져온다. 그리고 그 정도 시간이면 받아들일 만한 수준이다.

③ 지연(Latency)

클라이언트-서버 구조는 플레이어의 동작 사이에 잠재적인 큰 latency를 초래한다. 동작은 서버로의 roundtrip이 발생하고, latency는 LAN상에서는 거의 발생하지 않지만, 인터넷 상에서는 수백 ms가 된다. 긴 latency는 게임하는데 많은 지장을 준다. 즉 플레이어가 실제로 점프하기 전에 몇 피트를 더 나아가서 함정에 빠졌을 수도 있다. 이런 문제는 서버와 병행으로, 혹은 peer-to-peer 구조를 이용하여 다른 클라이언트와 병행하여 게임의 전부 혹은 일부의 로직을 클라이언트에서 실행하도록 하여 좀 더 빠른 응답을 얻을 수 있도록 하는 방법을 고려하게 한다.

빠른 응답은 좋으나 클라이언트 측에서 시뮬레이팅 하는데 약간의 심각한 문제가 발생한다. 그 중 한가지는 커뮤니케이션과 게임 로직을 상당히 많이 어렵게 만든다는 것이다. 서버에 하나의 마스터 시뮬레이션만 있는 것이 아니라, 동기를 맞춰야 하는 여러 개의 시뮬레이션이 존재하게 되기 때문이다. 여러 이벤트 중 단지 하나의 결과만이 허용된다. 즉, 클라이언트 시뮬레이션을 사용하게되면, 각 클라이언트에 있는 시뮬레이션끼리 충돌이 났을 경우 누구의 결정이 옳은지 결정하고 잘 못된 시뮬레이션을 취소시키는 메커니즘이 존재해야 한다. 어쩔 수 없이 패러독스가 존재한다. 예를 들어, 한 플레이어가 적을 향해서 로켓을 발사해서 명중했지만 그 적이 다시 살아나는 것을 볼 수도 있다. Quake에서는 lag이 존재하지만 paradox를 가지고 있지는 않다. 그래서 실제 경험처럼 느끼도록 해준다.

Quake에서는 lag을 없애기 위해서 다음과 같은 한 가지 방법을 사용한다. 플레이어가 다른 방향으로 돌면, 서버가 그 입력을 처리하는 것을 기다리지 않고 바로 실행한다. 방향을 돌리는 것과 관련해서는 paradox나 동기화 문제 같은 것들이 발생하지 않는다. 그러면서도 게임을 좀더 실감나게 만든다.(minor paradox가 가끔 발생)

④ 서버(Server)

Quake서버는 게임의 timebase와 상태를 관리하고 object 이동과 몬스터의 인공지능 등을 실행한다. 서버에 있어서 가장 흥미로운 점은 데이터 기반의 확장이다. 각 레벨(현재의 "world")은 object 위치와 형태, 벽의 위치 및 기타 등등에 의해서 묘사되고 그러한 정보는 디스크로부터 로드된 데이터 베이스에 저장된다. object와 몬스터의 행동은 Quake-C와 같은 내장 interpreted 언어로 기술된 함수에 의해서 프로그램되고 제어된다. 새로운 레벨을 만들 수 있을 뿐만 아니라, 새로운 게임 요소를 추가할 수도 있다. 사람을 쫓아다니는 로켓, 플레이어에게 달라붙어서 "나 여기 있다"라고 소리치는 alerter도 만들 수 있다. 이러한 것들은 C나 어셈블리 코드를 사용하지 않고서 만들 수 있다. 이러한 유연성은 퀘이크를 창조성있는 플랫폼으로 만들뿐만 아니라 프로그램 을 재 컴파일하지 않고서도 바꿀 수 있도록 해준다. 사실, 레벨과 Quake-C 프로그램은 Quake 없이도, 다시 로드되고 테스트될 수 있다.

⑤ 클라이언트(Client)

서버와 통신 레이어는 Quake에 있어서 중요한 요소

이다. 그러나 플레이어가 실제로 상대하는 것은 클라이언트이다. 그리고 3-D 엔진을 가지고 있는 부분 또한 클라이언트이다. 클라이언트는 키보드, 마우스 그리고 조이스틱 입력, 사운드 믹싱과 2-D drawing(메뉴와 같은), 상태 바 그리고 텍스트 메시지를 다룬다. 그러나 이런 것들은 비교적 다루기 쉽다. 3-D는 액션이 있는 곳이다. Quake의 3-D 엔진에서 시도한 2가지 중요한 사항은 다음과 같다. 모든 방향에서 진정한 3-D viewing을 허용(DOOM의 2.5-D와는 다름)하고, lighting과 정교한 픽셀 배치로 좀더 visual quality를 높이는 것이다. 물론 언제나 성능은 좋아야 한다.

서버에서는 3-D 엔진이 데이터 기반이었다. 데이터를 drawing하는 것은 두 가지 범주로 나뉘어 진다. World와 Entity이다. 각 레벨은 벽과 각 면들에 어떤 텍스쳐를 입혀야 하는 지에 대한 정보를 가지고 있다. Quake 데이터베이스는 또한 triangle mesh와 엔터티를 기술하는 텍스쳐를 포함한다. 원래는 모든 것을 단일 렌더링 파이프라인으로 그릴 계획이었다. 그러나 큰 벽과 몇 백개의 작은 폴리곤으로 이루어진 몬스터를 단일 파이프라인으로 좋은 성능을 얻을 수 없다는 것으로 판명되었다. 그래서 world와 entity는 전혀 다른 code path로 그려진다.

World는 BSP(Binary Space Partitioning) 데이터 구조로 저장된다. BSP 트리의 구조는 설명하기에 좀 복잡하므로 여기서 상세하게 들어가지는 않겠다. (만일 관심이 있다면, Dr. Bobb's Sourcebook 1995년 5월/6월, 7월/8월, 11월/12월호를 참조하라) 퀘이크의 목적에 따라, BSP 트리는 매우 유용한 2가지 일을 한다.

front-to-back 혹은 back-to-front 순서로 폴리곤 집합 traverse를 쉽게 할 수 있고, 공간을 convex volume으로 분할한다.

완전한 폴리곤을 그려야 한다면 Back-to-front order가 편리하다. 뒤에서부터 앞으로 폴리곤들을 그릴 수 있기 때문이다. 이러한 과정은 "painter's algorithm"으로 알려져 있다. 그러나 퀘이크에서는 폴리곤을 front-to-back으로 그린다. 좀더 자세히 설명하면, 모든 폴리곤들의 edge를 글로벌 리스트에 놓고 이 리스트를 rasterize한 후, 단지 앞에서 보이는 부분만 그린다. 이 방법의 큰 장점은 world에 있는 각 픽셀을 단지 한번만 drawing 함으로써, 소중한 drawing 시간을 줄일 수 있다는 것이다. 즉, 겹치는 어느 한 폴리곤 위에 겹쳐서 표시하는 일이 발생하지 않는다.

그러나, edge 리스트는 view pyramid에 있을 수 있는 수 천 개의 폴리곤들을 다룰 만큼 빠르지 않다. 그 많은 edge들을 edge list에 넣으면, 처리하고 sort해야 할 데이터가 너무 많아서 frame rate이 매우 늦어질 것이다. 그래서, PVS(potentially visible set)을 계산하는 BSP트리의 convex-partitioning 특성을 사용해서 고려할 폴리곤의 개수를 제한하였다. 한 레벨이 Quake포맷(맵이 생성될 때, 유틸리티 프로그램에 의해 한 번 실행되는 분리된 전처리 단계)으로 처리될 때, BSP 트리가 레벨로부터 생성되고, 그 BSP 트리의 각 convex subspace에 대해서 visibility calculation이 실행된다.

leaf가 하나 있다고 할 때, 그 유틸리티는 그 leaf에서 어느 곳으로부터도 볼 수 있는 subspace를 계산하고 그 정보는 BSP 트리에 저장된다. 즉, 아래 층 부엌에서는 어느 위치에서도 침실을 볼 수 없다고 한다면, 침실 폴리곤은 부엌 leaf의 PVS로부터 생략된다. 부엌 코너에서 거실을 볼 수 있다면, 부엌 leaf의 PVS는 거실 폴리곤은 볼 수 있는 것으로 기록한다. leaf를 위한 PVS는, 플레이어가 그 leaf의 어느 곳에 서있던지 고려할 필요가 있는 모든 폴리곤을 리스트로 가지고 있게 된다.

한 레벨에서 수 천 개의 폴리곤을 처리하기보다는, 렌더링 시간에 단지 현재 leaf의 PVS에 있는 수 백 개의 폴리곤들을 다루기만 하면 된다.(자르기, 변형하기, 투영하기 edge 리스트에 삽입) edge 리스트가 다룰 수 있는 레벨로의 로딩을 줄인다. PVS와 그 edge 리스트는, 매우 다양한 장면에서 빠르고 일관성 있는 성능을 가질 수 있도록 한다.

- PVS에 관한 한가지 : 계산 량이 매우 많을 수도 있다. PVS 4개의 프로세서가 있는 Alpha 시스템에서 15분 혹은 20분 정도가 걸릴 수 있다. 다행히도, Pentium Pro 시스템은 그 작업을 잘 처리할 수 있을 만큼 충분히 빨라지고 있고 그 코드도 더 빨라 질 것이라는데 의심할 여지가 없다. 그러나 PVS의 파워를 얻기 위해서는 그에 따른 대가를 치러야 한다.

edge 리스트가 모든 edge에 대한 처리를 한 번 완료하고 나면, 스크린을 정확하게 한번씩 덮는 span의 집합을 갖게된다. 이 리스트를 rasterizer에게 전달하는데 텍스쳐는 원근을 고려하면서, 적당한 비트맵을 그 span에 매핑시킨다. (이러한 작업은 정확성을 얻기 위해서 막대한 divide를 요구한다) 그러기 위해서는 16픽셀마다 divide를 하고 그러한 점들 사이에 보간(Interpolate)을 해야 한다.(이것이 어느 방향에서나 진정한 3-D를 볼 수 있도록 하기 위한 중요한 단계이다) DOOM의 조잡한 섹터 lighting과는 달리, Quake에서의 Lighting은 진짜 광원을 고려하고 그림자 처리를 한다. 이것은, 16-pixel 그리드 상에서 light sample을 가지고, 각 폴리곤마다 별도의 lighting map을 사용하며 텍스쳐가 메모리 버퍼에 그려지는 것처럼 grid에 따라서 각 폴리곤을 위한 텍스쳐를 prelight함으로써 구현된다.(기본적으로는 텍스쳐 맵이지만 color 대신 light값을 가지고 있다) 실제 텍스쳐 매핑은 이 prelit 텍스쳐로부터 이루어진다. 텍스쳐 매핑 시에는 어떠한 lighting도 발생하지 않는다. 이것이 일반적인 lighting과 어떻게 다른지 혹은 왜 바람직한지 설명하기엔 공간이 부족하다. 단지 그렇게 하는 것이 상세하고 고품질의 lighting과 좋은 성능을 낸다고 얘기할 수 있을 뿐이다.

⑥ 엔터티(Entities)

DOOM은 몬스터나 다른 움직이는 물체를 나타내기 위해서 flat poster를 사용했다. Quake에서의 큰 발전 사항 중의 하나는 진정한 3-D object인 폴리곤 entity로 전환했다는 것이다. 그러나 이것은 새로운 문제를 일으켰다. Entity가 수 백 개의 폴리곤으로 이루어질 수 있으며, 한 번에 여러 entity들이 보일 수 있다는 것이다. 그러므로 그러한 entity를 빨리 그리는 것이 우리의 주요한 사항 중의 하나이었다. 각 entity는 꼭지점의 집합과 그 꼭지점들을 지나는 삼각형의 mesh로 이루어져 있다.

entity에 있는 모든 꼭지점들은 하나의 집합으로 변형되어 투영된다. 그리고 perspective-correct 텍스쳐 매핑보다는 affine(linear)를 사용해서 모든 삼각형들이 그려진다. Affine은 더 빠르고 entity 폴리곤들은 매우 작고 멀리 떨어져 있어서 affine의 불완전성은 눈에 띄지 않는다. 또한 entity drawer는, world drawer가 long span drawing을 위해서 최적화되어 있는 것과는 달리, 작은 triangle을 위해서 최적화 되어있다.

그러나 entity들은 z-buffering으로 그려진다는 큰 차이가 있다. 즉, 그려지는 각 pixel의 거리는 덮어서 그려지는 pixel의 거리와 비교되고(z-buffer라 불리는 메모리 영역에 저장된다), 새로운 pixel이 더 가까울 경우에만 그려진다. 이것은 entity들이 어디로 이동하건 혹은 어떤 각도에서 보여지던지 간에 entity를 어색하지 않게 sort 하게 한다. 물론 그에 따라 감수해야 할 것은 있다.

z-buffering은 non-z-buffering 보다 느리다. 그리고 z-buffer는 보이는 world pixel 들을 Quake성능의 10%의 비용으로 match하도록 초기화되어야만 한다. 그러나 그 비용은 z-buffering의 단순함과 정확함에 의해서 얻을 수 있는 빠른 것 이상이다. 그것은, entity들을 제대로 그리기 위하여 복잡한 clipping과 sorting 연산을 수행해야만 하는 것으로부터 우리를 구했다. 그리고 모든 환경에서 결점 없는 drawing을 할 수 있도록 하였다.

우리는 현재의 leaf를 위해서 PVS에 있는 entity들을 단지 그리기만 하면 되기 때문에, PVS는 entity 성능을 향상시켜 준다. 다른 entity들은 존재하지 않으므로(클라이언트의 입장에서) 서버는 PVS 바깥에 존재하는 정보는 전송하지 않아도 된다. 이것은 drawing load를 줄여주는 것뿐만 아니라, 모뎀이나 인터넷을 통해서 전송되는 정보의 양도 줄여준다.

마지막으로, 우리는 퀘이크에서 연기 효과나 큰 폭발과 같은 효과를 갖길 원하였다. 그러나 표준 스프라이트(폭발의 핵심은 스프라이트임에 불구하고)나 폴리곤 모델로 어떻게 빠르게 처리할지 알 수 없었다. 해결책은 거리에 따른 스케일을 지원하는 많은 square와 색깔 있는 z-buffer된 사각형들을 이용하는 것이었다. 이러한 것들은 "particle"이라 불린다. 로켓 뒤에서 뿌려지는 수백개의 particle들은 놀랍게도 불꽃이 길게 뻗치는 것처럼 보인다. 특히 노란색으로 시작해서 빨간색으로 희미해지고 마지막으로 회색으로 되는 경우 더욱 그렇다. 그들은 그들이 진짜 3-D object를 표현하는 것처럼 눈이 믿게 한다.

dynamic lighting(폭발과 muzzle flash가 세상을 밝히도록 함)과 더불어 Particle, 신뢰성 없는 패킷 전송 방식은 Quake엔진에 최근 추가된 것들이다. 이러한 특징들은 well-render된, 그러나 약간 어색한 World를 만들었다. 메뉴와 같은 세부적인 것, 많은 최적화 그리고 버그 수정과 더불어 이러한 사항은 Quake를 기능적인 3-D, multiplayer 엔진으로부터 기술적인 도약을 이룰 수 있게 만든 "두 번째 90%"이었다.

(3) 검토 의견

Quake3 엔진은 언리얼 토너먼트 엔진과 함께 현재 최고의 3D 게임엔진으로 통한다. 32비트 텍스처를 비롯해 각종 최신 기술을 모두 이용할 수 있을 뿐 아니라 대부분의 3D 카드에서 완벽하게 실행된다. 성능과 안정성은 높게 인정되고 있다. 3-D 게임 엔진 중 라이센싱의 역사가 가장 오래되었으며, Heretic, Hexen, Hexen II, Heretic II, Soldier Of Fortune, Kingpin, Half-Life, Half-Life: Opposing Force, Half-Life: Counter-Strike, Daikatana, Sin, Heavy Metal : Fakk 2, and Star Trek: Elite Force 등의 게임이 Quake엔진을 사용하여 상업적으로 성공한 제품들임을 보더라도 성능면에서는 높은 완성도를 보이고 있다.

3. 맺음말

본 연구에서는 국내에서 접근 가능한 몇가지 온라인 3D 게임 엔진에 대해서 조사 비교 분석하였다.

본 논문이 온라인 3D 게임을 개발하고 있는 국내 업체와 3D 온라인게임 엔진을 연구하고 있는 학계 등에 도움이 되는 자료가 되기를 바란다.

4. 참 고 문 헌

[1] Mickey Kawick, Real-Time Strategy Game Programming, Wordware Publishing Inc., 1999.

[2] Kevin Hawkins, Dave Astle, Andre Lamothe,

OpenGL Game Programming, Prima Tech, 2001.

[3] Adrian Perez, Dan Royer, Advanced 3-D Game Programming Using DirectX 7.0, 2000.

[4] John David Funge, AI for Games and Animation : A Cognitive Approach, A.K. Peters, 1999.

[5] Robert Huebner, "Adding Languages to Game Engine", Game Developer, Sep.1997. (19971003/ huebner_01.htm)

[6] Michael van Lent, John Laird, "Developing an Artificial Intelligence Engine", Game Developer Conference Proceeding, March 1999, pp 577- 587.

[7] Steve Rabin, "Designing a General Robust AI Engine", Game Programming Gems, Charles River Media, 2000, pp. 221-236.

[8] Steve Rabin, An Architecture for RTS Command Queuing , Game Programming Gems II, Charles River Media, 2001.

[9] Paul Tozour, Influence mapping , Game Programming Gems II, Charles River Media, 2001.

[10] Paul Tozour, Strategic Assesment Techniques , Game Programming Gems II, Charles River Media, 2001.

[11] William van der Sterren, Terrain Reasoning for 3D Action Games , Game Programming Gems II, Charles River Media, 2001.

[12] Steven Woodcock, Flocking with Teeth , Game Programming Gems II, Charles River Media, 2001.

*Unreal

[13] Tim Sweeney, Unreal Networking Architecture, http://unreal.epicgames.com /Network.htm, 1999.

[14] Tim Sweeney, UnrealScript Language Reference, http://unreal.epicgames.com /Network.htm, 1998.

*Asheron's Call

[15] Toby Ragaini, "Postmortem: Turbine Entertainment's Asherons Call,"Game Developer's Magazine,April, 2000.

[16] 게임 클리닉! 에쉬론즈 콜, 도서출판 컴퓨플, 2000.

*Quake

[17] Michael Abrash, Quake's 3-D Engine: The Big Picture, http://www.gamedev.net /reference /articles/article987.asp

[18] Olivier Montanuy, Quake Specs v3.4, http:// www.gamers.org/dEngine/quake /spec /quake-spec34/