한국게임학회 논문지
제3권 제2호 2003년 9월

온라인 3D 게임엔진 개발에 관한 연구

이헌주, 박태준, 김현빈
한국전자통신연구원
가상현실연구부

A Study on Development of an On-line 3D Game Engine

Lee Hunjoo, Park Taejoon, Kim Hyunbin
Virtual Reality R&D Department
Electronics & Telecommunications Research Institute

요 약

국내의 온라인 2D 게임 개발 능력은 국제적으로 강한 경쟁력을 확보하고 있으나, 온라인 3D 게임 분야에서는 자본력과 기술력의 한계로 세계 시장에서 경쟁력 우위의 지속적인 유지가 어려운 상황이다. 본 연구에서 제안하고자 하는 온라인 3D 게임엔진 제반 기술은 외국의 업체에 비해 상대적으로 열세에 놓인 국내 게임 개발업체에 게임엔진 기술을 제공하여 세계적인 게임 경쟁력을 확보할 수 있도록 하는 것을 목적으로 한다. 이를 위하여 본 연구에서는 최신 기술동향을 반영한 3D 온라인게임 제작용 엔진을 위한 핵심 기술을 개발하였다. 제안된 게임엔진은 게임환경을 구성하고 시각화할 수 있는 렌더링 기술, 애니메이션 기술을 통합하고 보다 안정적인 온라인 서비스가 가능한 서버 기능을 제공한다.

Abstract

Domestic game companies are competitive in developing on-line 2D game contents. However, they have difficulties in maintaining the competitive edge in the area of on-line 3D game technologies owing to the limited funds and technologies. The 3D on-line game engine technologies we have intended to propose in this research are highly expected to be the key technologies ensuring the competitive edge over other countries. For this, we propose a full 3D online game engine that reflects the cutting edge technologies trend in the computer gaming industry. Our game engine is integrated with rendering and animation core technologies to visualize the game world, and provides stable network service through the Internet using server technologies.

1. 서 론

온라인게임 기술은 멀티미디어 컨텐츠 산업의 꽃이라고 불리우면서 중요성과 발전가능성에 대해서 강조되고 있다. 국내에서도 연간 매출이 1000억 원이 넘는 온라인게임이 등장했을 뿐만 아니라 시간이 갈수록 온라인게임을 즐기는 사용자가 늘어나고 있는 추세이다. 국내의 수많은 게임개발 업체들은 온라인게임을 주력으로 개발하고 있으며, 게임 사용자들의 요구가 2D를 기반으로 하는 게임에서 3D를 기반으로 하는 게임으로 옮겨가고 있으므로 현재 개발되고 있는 게임 가운데 3D를 기반으로 하는 게임이 상당한 비중을 차지하고 있다. 국내의 온라인게임 개발 수준이 세계적으로 인정을 받고 있으며, 정부는 물론이고 게임개발에 관련된 여러 분야에서 온라인게임 연구개발에 대한 중요성과

필요성에 대해서 계속해서 강조되고 있다.

국내의 온라인게임 분야는 많은 문제점들을 안고 있다. 특히 외국산 게임엔진을 사용함으로써 고가의 라이센스 비용과 판매 시에 로열티를 지불해야 하며, 엔진 자체도 예전 것이므로 비슷한 아류의 게임이 제작되어 경쟁력이 떨어지게 된다. 한편, 이를 극복하기 위해서는 국내 게임 개발업체의 대부분이 게임개발 인력이 부족한 상황에서 게임엔진 개발과 게임 컨텐츠 개발을 동시에 수행해야 한다는 부담을 안고 있다[1].

국내의 온라인 2D 게임 개발 능력은 국제적으로 강한 경쟁력을 확보하고 있으며, 관련 상품의 국내외 시장 점유율도 높으나, 온라인 3D 게임 분야에서는 자본력과 기술력의 한계로 세계 시장에서 경쟁력 우위의 지속적인 유지가 어려운 상황이다. 현재 상용 서비스 중인 온라인 3D 게임들은 SONY사의 Everquest, MS사의 Asheron' s Call과 같은 해외 게임들이 주류를 이루고 있으며, 국내의 경우, 최근 1년간 다수의 온라인 3D 게임이 개발되어 상용화 단계에 있으나 핵심 기술인 게임엔진은 공개되거나 판매되고 있지 않는 것이 현실이다. 국내에서 개발된 게임엔진들은 기술 공개를 꺼리는 국내 업계의 속성 때문에 실질적으로 공개되거나 유통되지 않고 있어 과잉, 중복 투자가 반복되는 실정이다.

본 연구에서는 온라인 3D 게임엔진과 관련한 제반 기술을 개발하여 게임 컨테츠 개발에 활용할 수 있도록 함으로서 외국의 업체에 비해 상대적으로 열세에 놓인 국내 게임엔진 기술의 경쟁력 확보가 가능하도록 하는 것을 목적으로 한다. 본 논문의 구성은 다음과 같다. 2장에서 게임엔진 기술의 국내외 동향을 알아보고, 3장에서는 개발된 온라인 3D 게임엔진에 대하여 엔진별로 기술하고, 상업적으로 성공한 다른 엔진과 비교하여 분석한 내용을 기술한다. 마지막으로 4장에서는 결론과 향후 연구방향에 대하여 기술한다.

2. 기술 동향

2.1 국내외 동향

게임선진국의 개발 업체들의 경우를 살펴보면 게임개발을 위해 엔진을 새로 제작하는 비효율적인 노력을 줄이기 위해 자체 기술 연구를 위한 연구소를 두고 끊임없이 사용자들의 요구를 반영하며 발전시키고 있으며 지속적인 연구를 통해 게임엔진 기술들을 보유하고 전문화 시켜가고 있다[2]. 또한 중소규모의 개발 업체들은 이러한 연구소를 통해 게임엔진 기술력을 도입하여 게임개발을 하고 있으므로 수준 높은 게임 컨텐츠를 제작할 수 있으며 세계시장을 장악할 수 있는 바탕이 되고 있다.

국내의 경우는 몇몇 게임 개발업체에서 자체적으로 개발한 엔진을 보유하고 있으나 폐쇄적인 구조로 공개를 꺼려하고 있으며, 대부분이 개발된 컨텐츠에 특화된 엔진으로 관련 기술을 공유하는 것이 현실적으로 불가능한 상황이다. 그리고 일부의 중소개발업체에서는 많은 시행착오와 함께 3D 엔진과 온라인게임에 필요한 서버 기술력 보유를 위해 과중한 인력 및 자금을 투입하며 개발을 진행하고 있는 상태이다. 따라서 직접적으로 사용자들에게 전달되는 게임성 내지는 게임요소 구현보다는 게임엔진 개발에 더 많은 노력을 쏟을 수밖에 없으며, 이로 인하여 개발되는 게임의 질적 수준을 향상시키는데 상당한 걸림돌이 되고 있는 것이 사실이다. 한편 게임 개발업체 가운데 많은 수가 엔진의 개발이 현실적으로 어려우므로 고가의 라이센스 비용을 지불하며 외산 엔진을 수입하여 컨텐츠 개발에 활용하고 있으나, 국내의 사용자 정서 및 환경에 맞는 컨텐츠를 제작하기에 적합하지 않아 애로를 겪는 사례도 발생하고 있다.

2.2 게임엔진 사례

본 절에서는 현재 기술적인 내용에 대해 문서로 발표된 대표적인 3D 게임엔진의 사례에 대하여 기술한다.

가. 퀘이크 엔진

퀘이크(Quake) 엔진은 게임엔진이라는 용어를 처음 사용한 엔진으로 볼 수 있다. Id Software에서 개발한 엔진으로 John Carmack을 중심으로 하는 프로그래머에 의해 개발되었으며 퀘이크 엔진, 퀘이크 II 엔진, 퀘이크 III arena 엔진의 3 가지 버전이 있다. 보통 퀘이크 엔진이라고 불리는 GPL(GNU General Public License) 엔진 소스는 1999년 공개되었으며, 이 소스를 가지고 게임을 개발한 경우에는 소스를 공개하도록 되어 있다. 퀘이크 엔진은 이전의 둠(Doom)이라는 엔진을 개선한 것으로서 최초의 3D 엔진으로 평가받고 있다. 둠 엔진의 경우에 등장하는 괴물은 3D

와 같이 보이지만 실제로는 2D 스프라이트를 사용하였으며, 카메라의 상하 회전이 불가능하여 실제로는 2½D 게임 엔진이라는 용어를 사용하기도 하였다[2]. 퀘이크 엔진은 십여개가 넘는 많은 게임을 개발하는데 사용되었으며 대부분의 게임이 일인칭슈팅 게임들이었다. 퀘이크 엔진으로부터 라이센스 받아 만들어진 Half-Life라는 게임은 퀘이크의 게임보다 더 많은 인기를 얻었으며 독자적으로 Half-Life 엔진이라는 이름으로 불리기도 하였다.

퀘이크 II 엔진까지는 8비트 컬러를 사용했으나, 퀘이크 III 엔진부터는 32비트 컬러를 사용하여 자연스러운 게임 환경이 가능하게 되었다. 그리고 퀘이크 엔진에는 네트웍 기능이 기본적으로 포함되어 있어서 네트웍에 연결된 PC를 별도의 서버로 사용하거나 하나의 PC가 서버와 클라이언트 기능을 동시에 담당하도록 구성할 수도 있다.

나. 언리얼 엔진

언리얼(Unreal) 엔진은 Epic Games 사의 Tim Sweeny가 주축이 되어 개발한 엔진으로 퀘이크 엔진 이후에 등장한 엔진으로 퀘이크 엔진의 기능을 대폭 개선한 엔진이며 액션 게임엔진으로는 가장 훌륭하다는 평을 받고 있다. 퀘이크 엔진을 참고하여 만들었기 때문에 퀘이크의 기본적인 장점을 갖고 있으며 여기에 여러 가지 문제점을 보완하였다. 초기의 게임은 C 언어만을 사용하여 개발되었고 C++ 언어는 C 언어에 비해 상대적으로 느리다는 일반인의 인식이 있었으나 C++을 채택하였다. 이는 프로그램의 복잡도가 커짐에 따라 프로그램의 보수유지와 생산성을 향상하기 위한 것으로 특히 엔진 기술을 판매하고자 할 경우에 중요한 의미를 갖는다. C++을 택함으로 객체지향적인 접근이 가능하였으며 여러 오브젝트를 사용하는 게임의 특성상 이는 매우 유용한 기능이다. 언리얼 엔진으로 개발된 게임의 텍스춰는 퀘이크 엔진에 비해 현실감이 높다는 평이 있다. 이는 같은 256 컬러 팔레트를 사용하더라도 각각의 텍스춰가 별도의 팔레트를 사용할 수 있도록 하여 16비트 컬러 이상을 사용하는 게이머에게 화려한 그래픽 효과를 보여준다. 이에 비해 퀘이크 엔진의 경우에는 하나의 레벨에서는 모든 텍스춰가 같은 팔레트를 사용하도록 되어 있다.

다. 넷이머스 3D 엔진

넷이머스(NetImmerse) 엔진은 NDL이라는 회사에서 개발한 3D 엔진이다. 이 회사는 렌더링 소프트웨어를 개발하는 회사로 넷이머스 엔진은 그 동안의 3D 소프트웨어 기술을 게임에 사용하도록 만든 엔진이다. 이 엔진은 기본적으로 3D 기능을 강조하여 개발된 것으로, 네트웍 기능이나 인공지능 기능은 특정 게임을 위해 별도로 개발하여야 한다. 또한 특정한 게임 컨텐츠 개발의 결과로 만들어진 엔진이 아니라 처음부터 게임 개발도구로 판매하기 위하여 만들어졌다는 것이 특징이다.

넷이머스 엔진은 실내와 실외 환경을 모두 지원하며, 애니메이션에 있어서는 deformable 애니메이션과, skeleton 애니메이션 등의 기능을 제공한다. 파티클 엔진의 경우에도 눈, particle emitter, particle bomb, particle array, particle cloud와 같은 기능을 지원하고 있다. 그리고 3D 사운드를 지원하며 실내의 음향 반사를 고려하여 사운드를 처리할 수 있다. 충돌 감지에 있어서는 움직이는 오브젝트와 움직이지 않는 오브젝트로 구분하는 계층 구조적인 접근방법을 사용함으로서 충돌 감지에 소요되는 CPU 처리시간을 줄이도록 하였다. 한편, 테리언(terrain)의 LOD(Levle of Detail)에는 progressive mesh 알고리즘을 사용함으로서 카메라의 이동이 있을 경우에도 자연스럽게 처리될 수 있도록 하는 continuous LOD 기능을 제공한다. "Dark Age of Camelot"은 넷이머스 엔진을 이용하여 개발된 게임 가운데 하나이다.

라. 뮤 엔진

국내에서 개발된 3D 엔진으로 온라인게임 뮤에 사용되었다. 접속자의 수가 수만명에 이르고 있어 본격적인 MMORPG 엔진으로 평가받고 있으나 안정성에 일부 문제가 있기도 하였다. 3D 효과를 최대한 이용하여 비, 바람에 따라 나무가 움직이는 등 사실적인 효과를 높였으며 파티클 엔진의 성능도 우수한 것으로 판단되고 있다. 인공지능 기능도 이전의 게임에 비해 충실하다고 평가되고 있으나 게임엔진의 구체적인 내용이 일반적인 내용 외에는 발표된 것이 없다.

3. 온라인 3D 게임엔진

그림 온라인 3D 게임엔진 구성도

본 연구에서 제안하는 온라인 3D 게임엔진인 Dream3D

는 다중참여자용 롤플레잉 게임 제작을 위한 게임엔진으로 [그림 1]과 같은 구조로 이루어진다. Dream3D에서의 주요 세부 엔진으로는 렌더링 엔진, 애니메이션 엔진, 사운드 엔진, 서버 엔진이 있다. 렌더링 엔진은 게임 그래픽 데이터를 게임 진행 상황에 맞게 화면에 출력해주는 엔진이며, 애니메이션 엔진은 캐릭터나 객체의 애니메이션을 담당하고 물리현상을 실시간으로 계산하는 엔진이다. 사운드 엔진은 게임 사운드 데이터를 게임 진행 상황에 맞게 스피커에 출력해주는 엔진이며, 서버엔진은 네트워크를 통하여 다중 동시 사용자를 처리하고 게임 환경의 일관성 유지 및 진행을 담당하는 게임 서버들의 구현을 지원하는 엔진이다. 각 세부 엔진에 대한 자세한 내용은 다음에서 기술한다.

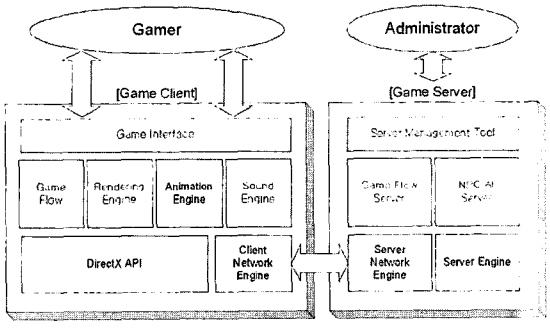


그림 1 온라인 3D 게임엔진 구성도

3.1 렌더링 엔진

렌더링 엔진은 게임 상에 등장하는 각종 그래픽 데이터를 게임 진행상황에 맞게 화면에 실시간으로 출력해주는 모듈이다. 다시 말하여 게이머에게 게임의 진행상황이나 게이머의 입력에 대한 게임 캐릭터의 반응, 그리고 게임 진행 중에 발생하는 각종 이벤트 정보를 화면에 출력하는 모듈이라고 할 수 있다[13].

본 렌더링 엔진의 구성은 [그림 2]와 같다. 입력된 자료는 scene graph로 구성하여 내부 관리된다. Scene graph는 동적인 물체와 정적인 물체에 대해 별도로 구성되며, 실시간 처리를 통해 실제 렌더링될 다각형과 전환행렬(transform matrix), 텍스춰 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. 각 객체가 화면에 나타나게 되는 비율을 계산하여 렌더링 하고자 하는 객체의 표현 정도를 결정하고 이들 중 카메라 시야에서 벗어나거나 다른 물체에 의해 가려지는 삼각형이 view-frustum culling, back-face culling, hidden-surface culling 등의 과정을 거쳐 제거된다. back-face culling은 DirectX에서 기본으로 제공되는 기능을 사용하였고, view-frustum culling은 DirectX에서 제공되는 카메라 관련 파라미터를 이용하여 간단한 행렬 연산을 통해 구현하였으며, hidden-surface culling은 BSP/PVS 기능을 통해 구현하였다.

속도의 향상을 위해 정적환경에 대한 실시간 처리는 카메라 변수가 변경된 경우에만 수행하며, 동적 환경에 대한 실시간 처리는 대상 객체의 자세가 변경될 때마다 수행한다. 선택된 삼각형들은 향후 반투명 혹은 투명 물체의 올바른 화면 출력을 위해 카메라로부터의 거리에 따라 정렬된다.

실시간에 고품질 렌더링을 수행하기 위하여 본 엔진에서는 projective texture mapping 기법을 제공하였다. 또한 본 렌더링 엔진은 거울면 효과와 반사/굴절 효과를 올바르게 처리하기 위하여 환경 맵 기법을 활용하며, 물체 표면의 질감을 보다 사실적으로 표현하기 위하여 범프(bump) 맵 기능을 제공하였다. 범프 맵은 텍스춰를 물체 표면에 단순히 입히는 것이 아니라 물체 표면에 주름이 지거나 올록볼록한 입체감이 느껴지도록 하는 기술이며, 본 렌더링 엔진에서는 엠보싱(embossing) 효과를 지원한다. 환경 맵은 가상환경에서 특정 객체를 둘러싼 환경을 텍스춰로 저장하여 그 객체의 표면에 입히는 기술로, 반사 속성을 가진 물체를 표현하기에 적합하다. 본 렌더링 엔진에서는 주위 환경을 한 장의 원형 텍스춰로 저장하고 객체의 표면의 각 정점의 반사 벡터(reflection vector)로부터 텍스춰 좌표를 계산하여 물체의 표면에 입히는 방법을 사용한다. 실시간에 환경 맵핑을 할 물체를 중심으로 환경 맵을 생성하기 때문에 동적인 물체에도 적용할 수 있다.

반투명 물체의 올바른 출력을 위해 그래픽 하드웨어로 전달되는 다각형들은 이전 실시간 처리 단계에서 카메라로부터의 거리에 따라 정렬된 순서에 따라 처리된다[1]. 이때, 빠른 영상 출력을 위해서는 그래픽 가속 하드웨어의 내부 연산의 지속성을 유지하여 내부 파이프라인이 멈추지 않고 동작하도록 유지하는 것이 중요하다. 이를 위해 본 렌더링 엔진에서는 3차원 객체나 특수효과 등 기타 객체들이 동일

한 버텍스 버퍼(vertex buffer)를 공유하도록 하여 렌더링 시에 불필요한 버텍스 버퍼 전환이 발생하지 않도록 하였으며, 또한 같은 텍스춰를 사용하는 삼각형들이 가능한 함께 모여서 렌더링되도록 하여 내부 텍스춰 및 표면 속성 상태 변화가 최소화되도록 하였다. 이와 더불어 그래픽 메모리와 시스템 메모리간의 텍스춰 스와핑(swapping)을 최소화하기 위해서 텍스춰 압축 및 밉맵(mip map) 생성 기법을 적용하여 텍스춰가 그래픽 메모리에서 차지하는 용량을 줄이도록 하였다.

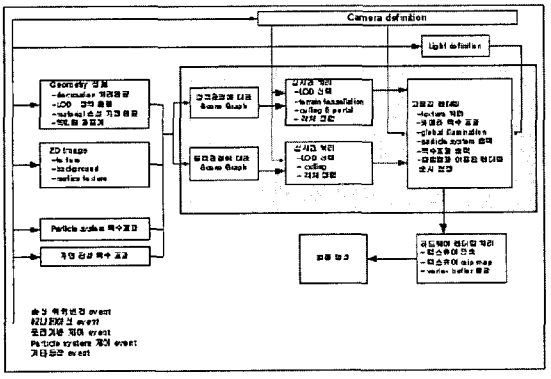


그림 2 렌더링 엔진의 구성

렌더링 엔진의 광원은 4개의 기본 광원과 2개의 응용 광원으로 구성된다. 기본 광원에는 점 광원, 직사 광원, 스팟 광원, 주변 광원이 있고 응용 광원에는 텍스춰 광원, 그림자 광원이 있다. 기본 광원은 DirectX에서 기본적으로 제공되는 광원들로 주변 광원을 제외한 세 광원은 사용될 경우 하드웨어 리소스를 할당받게 되어 렌더링 속도에 영향을 미치게 된다. 주변 광원은 밤과 낮을 표현하는 등 전체적인 삼차원 환경의 밝기를 설정하기 위한 광원이다. 텍스춰 광원은 사용자가 지정한 텍스춰를 광원의 위치에서 뿌리는 영사기 역할을 하는 광원이고 그림자 광원은 지정된 광원의 위치와 방향에서 물체에 그림자를 뿌려주는 광원이다.

각 광원은 그 특성에 따라 고유의 속성들을 갖고 관리된다. 사용자는 광원을 임의로 추가 또는 삭제할 수 있으며, 필요한 경우 각 광원의 속성을 변경할 수 있다. 광원의 위치와 색상도 이 속성에 속하므로 동적으로 속성을 변경함으로써 동적 광원을 구현할 수 있다. 또한 본 렌더링 엔진은 일련의 광원들을 기술할 수 있는 스크립트를 제공하여 게임에서 사용되는 광원들을 파일로부터 읽어 들여 한번에 추가할 수도 있다.

텍스춰 광원은 라이트 매핑(Light mapping)이라고 하는 기술을 광원의 형식으로 구현한 것으로 사용자가 지정한 임의의 텍스춰를 지정된 위치에서 지정한 방향으로 뿌린다. 이 광원은 물리적 광원의 수를 줄이기 위해 원형의 텍스춰를 사용하여 스팟 광원의 효과를 내는 데 사용될 수 있다. 그림자 광원도 마찬가지로 텍스춰를 삼차원 공간에 뿌리는 역할을 하나, 이 경우엔 그림자 맵(shadow map)이라는 텍스춰를 직접 생성하는 과정을 거친다. 본 렌더링 엔진에서 제공되는 기능들은 다음과 같다.

- 3차원 정적/동적 객체 출력 기능
- 그림자 출력 기능(fake shadow, shadow volume 지원)
- 동적 지형 LOD 기능
- 그림자 생성 기능
- 광원/카메라 제어 기능
- 라이트맵 제어 기능
- 기본 텍스츄어 매핑 기능
- 환경맵, 범프맵을 이용한 고품질 텍스춰 매핑 기능
- BSP/PVS(실내 지형), Culling, LOD 등을 활용한 실시간 렌더링 기능
- 특수효과 기능 (빌보드, 렌즈 플레어, 스프라이트, 파티클 시스템, 모션 블러, 안개 등 지원)

렌더링 엔진에서 지원되는 파일 포맷은 GME와 GE5이며 객체의 형태를 표현하기 위한 형식이다. 3D Studio MAX에서 동작하도록 plug-in의 형태로 개발된 익스포터(exporter)를 통하여 3차원 객체의 기하정보와 표면 속성 정보를 GME나 GE5 형식으로 저장할 수가 있다. 본 익스포터를 이용할 경우, 게임 디자이너는 GME나 GE5 형식에 대한 자세한 지식 없이도 스스로 제작한 게임 그래픽 자료를 렌더링 엔진에서 사용할 수 있는 형식으로 전환할 수 있다.

3.2 애니메이션 엔진

애니메이션 엔진[10]은 캐릭터나 아이템 등 애니메이션 오브젝트의 동작을 재생 및 실시간 생성하기 위한 엔진이며, 키프레임(key-frame), 모션 DB 기반의 스키닝(skinning), 관절체 애니메이션, 얼굴표정 제어, 모션 섞음(blending), 모션 보간(interpolation), 역운동학(inverse

kinematics), 버텍스 및 텍스쳐 애니메이션, 계층구조 애니메이션 등의 기능이 제공된다. [그림 3]은 애니메이션 엔진의 구조를 나타낸다. 애니메이션 객체(instance or Actor Instance)는 애니메이션 템플릿(Template or Actor Definition)으로부터 생성되며, 템플릿을 구성하는 요소로는 관절 구조(bone hierarchy)와 몸체(body) 및 행동(Skill)이 있다. 행동은 관절(joint)의 궤적(trajectory or path)의 집합으로 표현된다.

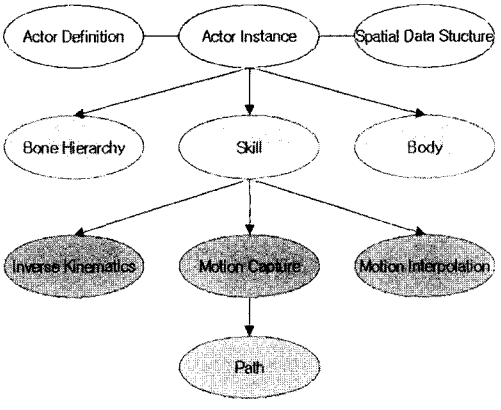


그림3 애니메이션 엔진의 구조

행동은 역운동학과 모션 보간, 모션 캡쳐 데이터를 이용하여 캡쳐된 모션 데이터의 재생뿐만 아니라 실시간 모션 변형을 할 수 있다. 몸체는 캐릭터의 정보 즉, 캐릭터의 텍스쳐, 버텍스, 스키닝을 위한 관절 정보를 담고 있으며 모션 데이터를 적용하여 캐릭터가 애니메이션을 하도록 하였다. 본 애니메이션 엔진에서 사용되어지는 파일 포맷은 AME와 AE5 형식이다. 다음은 애니메이션 엔진에서 제공되는 기능들을 설명한다.

가. 3차원 객체 동작 제어 기능

애니메이터가 직접 키프레임을 인위적으로 만들어서 저장한 데이터일 경우 애니메이션 엔진에서 이를 로딩하여 특정시간에 동작을 표현 하고자할 때, 키프레임 사이를 보간하여 표현된다. 또한 모션 캡쳐된 데이터도 마찬가지로 사용된다.

나. 관절체 동작 제어 기능

피직(physique)은 애니메이션을 적용하려는 관절체와 관절체에 붙어 있는 캐릭터의 메쉬 데이터를 연결시켜 관절체가 움직임에 따라 캐릭터의 메쉬 데이터도 따라 움직이도록 연결하는 작업을 말한다. 스티칭(stitching)은 하나의 뼈에 붙어있는 연속적인 메쉬에 대하여 동작하며 강체 애니메이션에서 하나의 다각형은 그 다각형이 부착된 뼈대를 표현하는 행렬에 의해 변환된다. 스티칭의 경우는 뼈대들에 붙어 있는 정점들이 뼈와 일대일 대응관계를 가진다. 하나의 다각형에 포함되는 점들일지라도 뼈에 일대일 대응하면서 서로 다른 뼈에 붙어 있을 수 있다. 따라서 두 개의 뼈대를 하나의 다각형으로 꿰매는(stitch) 것과 같이 두 뼈대의 움직임으로 인해 생기는 메쉬의 틈을 하나의 다각형으로 덮는 효과를 만들어 낼 수 있다. 스티칭은 유용한 기법이지만 관절부를 많이 구부리면 관절부분의 다각형이 심하게 늘어나며 결과적으로 매우 부자연스러운 모습이 된다. 만약에 팔뚝을 180도에 가깝게 구부리면 위 팔뚝과 아래 팔뚝 사이에 틈을 하나의 폴리곤으로 메워야 하기 때문에 팔꿈치 부분이 자연스럽지 못하고 각이 지어서 보인다. 스티칭의 이러한 문제점을 없애려면 하나의 정점이 하나의 뼈대에 의해서만 영향을 받는 것이 아니라 여러 개의 뼈의 영향을 받도록 하여야 하며 이를 스키닝(skinning)이라고 한다. 이러한 스키닝이 이루어지기 위해서 각 정점들은 자신들에게 영향을 주는 뼈들에 대한 정보, 즉 이름이나 인덱스를 가지고 있어야 한다. 또한 어떤 뼈대에 얼마만큼의 영향을 받을 것인지에 대한 정보도 가지고 있도록 하였다.

다. 고급 동작 생성

애니메이션 엔진의 가장 큰 특징은 실시간으로 모션을 변형할 수 있다는 것이다. 역운동학을 이용하여 캐릭터의 손이나 발이 외부 환경이나 목적으로 하는 경로를 가도록 실시간으로 모션을 변형할 수 있고, 모션 보간을 이용하여 동작간에 연결시 부자연스러움을 최소화하였고, 모션간에 서로 섞어 중간동작을 생성할 수도 있도록 하였다. 모션 보간은 2개의 모션과 이들 모션을 어느 정도의 비율로 보간할 것인지에 대한 가중치를 스칼라 값으로 주고, 그 값에 따라 중간 모션을 생성한다. 가중치의 값은 선형 함수나 시그모이드 함수에 따라 결정될 수 있다.

또한 애니메이션 엔진에서는 3차원 그래픽스를 이용한 얼굴 애니메이션을 위해 얼굴 근육을 이용한 방법을 이용하였다. 이 방법은 얼굴에 미리 근육을 설정하고, 얼굴의 스

킨과 연결시킨 후, 이 근육을 움직이면 얼굴의 표정이 변하는 원리를 이용한 것이다. 본 애니메이션 엔진에서는 근육의 움직임을 뼈대를 이용하여 처리하였다. [그림 4]는 얼굴 애니메이션을 보여주고 있다. 또한 모션 섞음을 이용하여 2개의 동작 또는 3개 이상의 동작을 섞어서 실시간을 표현할 수 있는 기능을 제공하고 있다.

라. 버텍스와 텍스쳐 및 계층적 애니메이션

버텍스의 위치를 편집하여 키 프레임 형태의 애니메이션을 적용할 수 있고, 텍스춰의 UV 좌표를 애니메이션할 수도 있다. 또한 텍스춰 애니메이션이 가능하기 때문에 분수나 폭포, 폭발 장면을 쉽게 표현할 수 있다. 또한 물체간에 연결성를 고려하여 한 물체가 다른 물체에 종속되어 애니메이션이 가능하다. 이러한 애니메이션을 계층적 애니메이션이라고 한다.

위에서 기술한 것과 같이 애니메이션 엔진에서는 3차원 관절체 동작을 제어하고, 고급 동작을 생성하여 자연스러운 캐릭터 애니메이션이 가능하다. 뿐만 아니라 버텍스, 텍스춰, 계층적 애니메이션 기능을 추가하여 캐릭터 외에 아이템의 애니메이션도 쉽게 할 수 있도록 하였다. 또한 기능의 가감이 편리한 구조로 되어 있으므로 향후 업그레이드가 손쉽게 이루어질 수 있도록 하였다.

[그림 4] 얼굴 애니메이션

3.3 사운드 엔진

사운드 엔진은 게임엔진에서 게임 사운드 데이터를 게임 진행 상황에 맞게 스피커에 출력하여 주는 엔진이다. 사운드 엔진을 통하여 적절한 사운드를 재생할 수 있으며 필요시 일시정지나 반복, 재생 등을 수행 할 수 있다. 사운드 엔진에서는 기본적인 재생기능 이외에 특수효과 기능을 이용하여 사운드를 입체음향으로 만들기도 하고, 필요에 따라 특수효과 필터들을 적용하여 새로운 음향을 만들 수도 있다[8]. 사운드 엔진에서 제공되는 기능들은 다음과 같다.

- 기본적인 사운드 렌더링 기능
- 3D 사운드 생성 기능
- 사운드 모델 기능
- 특수 효과 기능

가. 음상 정위와 머리전달 함수

음상 정위 기술이란 이 음상을 공간 상의 원하는 위치에 형성하고 제어하는 기술을 일컫는다. 오디오 기술의 발달 과정에 비추어 음상을 고려해볼 때에 모노 재생방식(Monophony)은 공간 정보없이 점 음상을 형성하는 기술이며, 다음 세대인 2채널 스테레오 재생방식(Stereophony)은 좌우 두 스피커 사이의 공간에 음상을 형성하는 기술이다. 그리고 나아가 최근의 입체음향(3D Sound) 기술은 머리전달 함수(HRTF: Head Related Transfer Function)를 사용하여 3차원 공간상의 임의의 방향에 위치한 입체음상을 생성하는 기술을 일컫는다[8].

시각적인 정보와 더불어 청각적인 정보(즉, 음향신호)의 기록 및 재생 기술이 진보하여 스테레오나 서라운드 방식의 충실한 현장감의 재생이라는 목표에서 좀 더 나아가 음상을 임의의 위치에 둘 수 있는 3차원 음향 공간의 자유로운 재생을 목표로 하게 되었다. 3차원 음향공간을 인지하기 위하여 머리전달 함수를 이용하여 음상을 정의할 수 있다. 머리전달 함수란 음원의 3차원적 위치 변화에 따른 음원과 사람의 고막(혹은 외이도) 사이의 전달함수를 말한다. 3차원 공간의 어떤 위치에서 음이 방사될 때, 청취자는 방사 음원의 위치를 지각하게 된다. [그림 5]는 3차원 공간에서 청취자에 대하여 일정 거리에 위치하는 구면 상의 음원의 배치를 나타내고 있다. 정 중앙에 청취자가 있고, 이때 음원의 방향은 방위각(Azimuth)과 고도각(Elevation)에 의해 지정된다. 인간의 청각 시스템이 수평면에 대하여 음원의 방향을 지각할 수 있는 주요인은 오른쪽 귀와 왼쪽 귀에 들어오는 소리의 세기차와 시간지연차이다. 방향 지각에는 소리의 세기차와 시간지연차 요인 이외에도 여러 요인들이 작용하게 되는데 예를 들어, 청취자의 정면이나 뒤에서 재생

되는 음의 위치나 수직면에서의 음원의 고저를 인지하는 과정에는 입사파가 내이(內耳)에 도달하기까지의 몸통, 머리, 외이(外耳)와의 상호작용과 직접음의 반사와 회절에 의한 음의 스펙트럼적 변화 등 다양한 요인들이 작용하게 된다. 이러한 요인들을 포괄적으로 가지고 있는 것이 머리전달 함수이며, 여기에는 두 귀 사이의 소리 세기차나 시간지연차 이외에도 음원의 위치에 의해 달라지는 사람의 머리 및 귀에 의한 소리의 주파수별 감쇄 특성 등의 정보가 들어 있다.

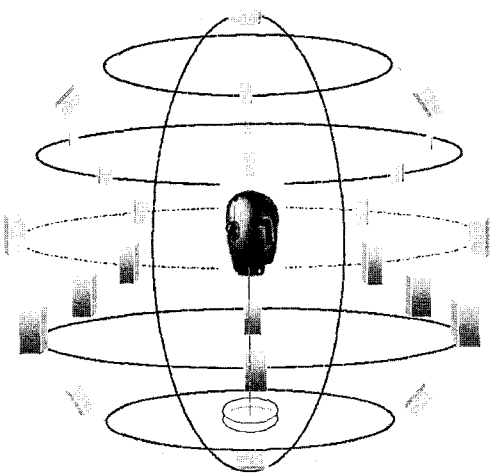


그림 5 머리전달 함수를 얻기 위한 스피커 배치도

방향감 구현의 경우, 특정 방향의 음원으로부터 두 귀까지의 좌우 머리전달 함수의 쌍을 모노 음에 대하여 각각 필터링하는 바이노럴 머리전달 함수 필터링을 수행함으로써 구현이 가능하게 된다. 방향에 따른 머리전달 함수의 필터링은 보통 디지털 신호처리 기법을 사용하여 컨볼루션(Convolution) 연산의 수행을 통해 구현한다. 머리전달 함수 필터링을 FIR 필터로써 구현할 경우에 필터 계수들은 임펄스 응답(HRIR: Head Related Impulse Response) 열의 원소들과 같으므로 측정된 머리전달 함수 임펄스 응답 데이터를 정규화하여 필터 계수로 사용하며, 컨볼루션 연산은 다음 식과 같이 표현된다.

$$y(n) = \sum_{m=0}^{M-1} h(m)x(n-m)$$

여기서 $x(n)$은 음원 샘플, $h(m)$은 머리전달 함수 필터 계수, 그리고 $y(n)$은 필터링된 출력 음향신호를 나타내며, M은 컨볼루션되는 FIR 필터의 탭 수이다.

이러한 연속적인 머리전달 함수(HRTF) 필터링 처리를 거치면 음원 신호는 방향성을 갖게 되며, 이후 이득(gain) 조절을 통한 거리감 제어 과정과 공간감을 부여하는 잔향생성 과정을 통과하여 입체음향으로서 스테레오 헤드폰이나 2채널 스피커로 재생시키게 되는 것이다. 이와 같이 음상정위 기술을 적용하여 재생할 경우에 실제 소리가 발생한 위치에서 음상을 지각하게 하는 것이 가능하게 된다.

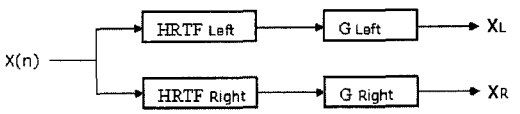


그림 6 단일 모노 음원에 대한 바이노럴 입체음향 블록도

나. 음장 제어

음원을 둘러싸고 있는 실내 공간의 특성에 따라 동일한 음원이라 할지라도 청취자에게 다른 음향효과를 줄 수 있다. 예를 들어, 동일한 피아노 소리라도 콘서트 홀에서 들을 때와 일반 강당에서 들을 때에 청취자는 다른 음향 경험을 갖는다. 이는 실내 공간의 크기, 구조, 재질 등에 의해서 음원에 대한 직접음, 초기 반사음, 잔향 패턴 및 잔향 시간 등이 달라지기 때문이다. 잔향은 음원으로부터 방사가 그친 후에도 천장이나 벽으로부터의 반사가 계속되어 울리는 음으로, 공간감 생성에 주요한 요인임과 동시에 거리감 생성에도 중요한 역할을 한다. [그림 7]과 같이 반사음과 잔향을 인공적으로 제어하여 특정 실내에 음원이 있는 것과 같은 음향효과를 생성하는 기술을 음장 제어기술이라 한다[9]. 음장 제어에 가장 많이 사용되는 것 중의 하나로 Schroeder 잔향기가 있는데, 이 잔향기는 병렬로 연결된 4개의 빗형 필터와 직렬로 연결된 2개의 전대역 통과 필터로 구성된다. 빗형 필터는 특정 주파수가 진동하는 모드를 시뮬레이션하며, 전대역 통과 필터는 잔향 밀도를 증가시키는 역할을 한다. 또한, 잔향 밀도를 더욱 풍부하게 하기 위하여 중첩된 전대역 통과 필터와 저역 통과 필터를 갖는 일반화된 잔향기가 제안되었다.

이외에 실내의 잔향 특성을 음향학적으로 모사하는 음선 추적(Ray Tracing) 방식과 이미지 모델(Image Model) 방식이 있다. 음선 추적 방식은 음원에서 나오는 음은 모든 방향으로 방사되는 성질을 고려하여 음의 에너지 분포를 구하여 가상 음장을 생성하는 방법이고, 이미지 모델 방식은 빛

이 거울에서 반사하는 것과 같이 음파도 실내의 벽면에 부딪혀 한번 반사한다는 전제하에 반사 경로를 구하여 가상 음장을 생성하는 방법이다. 또한 공간전달함수(Room Transfer Function)를 이용하는 방법이 있는데, 이는 특정 공간의 공간적 단서가 내포된 공간전달함수를 측정하여, 이를 단순음과 컨볼루션을 행하여 특정 실내의 음장 특성을 부가하는 방법이다.

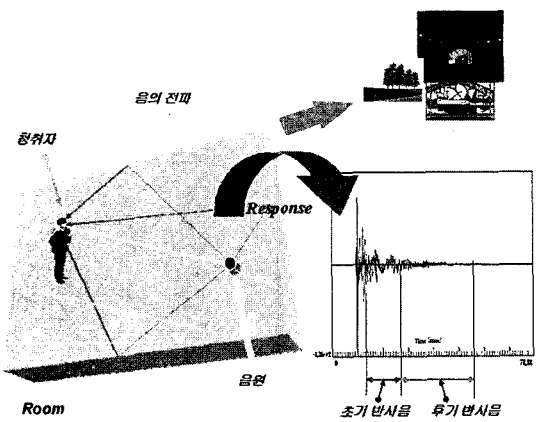


그림 7 음장 제어 기술 및 음장 DB

다. 특수 음향 효과

음향효과는 소리에 색다른 효과를 더해주는 것으로 일반적으로 에코, 딜레이 코러스, 피치 시프트 등 여러 가지가 있다. 영화나 TV, 게임, 음반 등 많은 분야에서 만들어진 스토리들 또는 음악 자체의 느낌을 변화하거나 증폭하기 위해서 여러 가지 음향효과가 쓰이게 된다. 음향효과들은 영상과 함께 쓰여지는 경우가 많다. 화면에 특정장면이 나온다면 이에 적합한 소리를 기존 자료를 활용하여 소리를 넣거나, 아니면 직접 신디사이저를 이용하여 소리를 만들어 여러 효과를 더하여 이용한다. 이렇게 함으로서 우리가 접하게 되는 컨텐츠의 질을 좀더 높일 수 있게 된다. 이러한 특수 효과를 만들 수 있는 도구들은 간단하게는 소리의 크기를 변화시키는 것부터 소리를 울리게 하거나, 떨리게 하거나, 찌그러지게 하거나, 특정 음색을 돌출 시키거나, 주파수 대역별 증폭을 가하거나 하는 일들을 할 수 있게 한다.

라. 구 조

모든 사람이 자신의 바이노럴 신호를 녹음할 수 없기 때문에, 현재는 주로 청각 능력이 뛰어난 음악가나 표준치의 머리 모형을 가진 더미헤드(Dummy Head)에 장착한 마이크로폰을 통하여 바이노럴 신호를 녹음하고 이를 일반 청취자에게 들려주는 방식을 이용한다. 필터링에 의한 방법은 단순음(모노음 또는 스테레오음)을 변형하여 입체음을 생성하는 방식으로, 주로 머리전달 함수가 필터로서 이용된다. 이 머리전달 함수와 단순음을 컨볼루션하면 원하는 공간상의 위치에 음상을 정위시킬 수 있다. 무향실이 아닌 특정 실내에서 측정한 머리전달 함수를 공간전달함수라고 하며, 이를 이용하면 그 실내의 음장 특성을 생성할 수 있다.

사운드 엔진의 구성은 다음 [그림 8]과 같다. 사운드 엔진은 간단하게는 인간의 청각 구조와 유사한 더미헤드를 사용해 좌우 두 귀 채널에 대해 각각 녹음한 음을 헤드폰으로 그대로 재생하면 가능하다. 그러나 [그림 8]에서의 흐름과 같이 머리전달 함수를 이용한 음상 정위, 실내 음장모델을 반영한 음장 제어의 신호처리 과정을 거쳐 입체음향을 생성하고 재생함으로써 구현하였다.

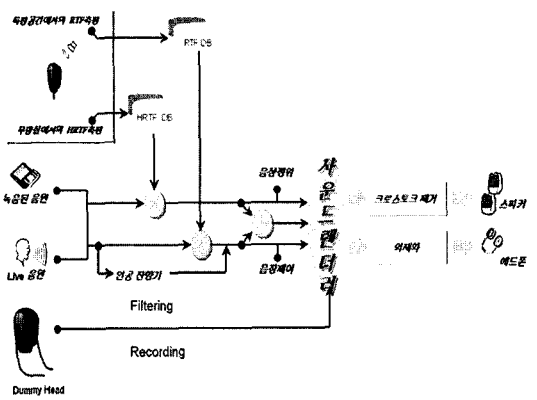


그림 8 사운드 엔진의 구성도

3.4 서버 엔진

서버 엔진은 다수의 사용자들이 온라인게임 내에서 서로 상호작용하며 게임을 수행할 수 있도록 하는 서버 구축을 지원하며, 서버의 최적화를 위한 여러 핵심적인 기능과 알고리즘을 제공하게 된다. 또한, 게임서버들은 지속적으로 서비스를 계속 수행해야 하므로 무엇보다도 안정성이 요구되며, 서버의 과부하, 허용치 이상의 네트워크 지연 등으로 인한 게임 성능 저하가 발생되지 않도록 서버의 성능을 최적으로 유지시켜 게임 사용자들의 불편을 최소화해야 한

다. 게임서버 엔진은 기본적으로 특정 장르의 게임에 종속되지 않고 다양한 장르의 온라인게임 서버를 구현할 수 있는 기본적인 서비스를 제공하는 것을 목표로 하였으며, 주요 기능은 다음과 같다.

첫째, 클라이언트와의 통신을 위해 이벤트 모델을 사용하여 처리하였다. 게임 개발자는 통신 단계를 직접 사용하지 않고, 이벤트 처리 기능을 사용하여 쉽게 프로그래밍 할 수 있도록 하였다. 둘째, 성능 극대화를 통한 다수의 동시 사용자를 지원하도록 하였다. 즉, 다수의 동시 사용자가 하나의 게임월드에서 몰입하여 게임을 즐길 수 있도록 하였다. 셋째, 분산 처리기능을 제공한다. 게임월드를 지역적인 혹은 사용자수의 관점에서 분할하여 다수의 서버가 처리함으로써 게임서버는 확장성(scalability)을 갖게 되며, 원활한 게임 운영을 위해 서버들 사이의 처리 능력을 조절하여 분산한다. 넷째, 끊김 없는(seamless) 게임월드를 구현하였다. 하나의 게임월드를 여러 서버가 관할함을 인지할 수 없도록 자연스러운 이동을 가능하게 하는 분산 서버를 구축하였다.

가. 구 조

게임서버 엔진은 온라인게임의 동작을 서버와 클라이언트의 상호작용으로 간주하며 그 기본단위가 되는 하나의 상호작용은 '이벤트'로 정의한다. 게임서버 엔진 기반 온라인게임의 모든 동작은 클라이언트의 이벤트 발생, 서버로의 이벤트 전달, 서버의 이벤트 처리, 결과 이벤트의 관련 클라이언트로의 전달이라는 4단계로 이루어진다. 일반적인 온라인게임 구현 과정에서 서버와 클라이언트 사이의 이벤트는 기획 과정에서 사전에 결정된다. 이벤트의 자료

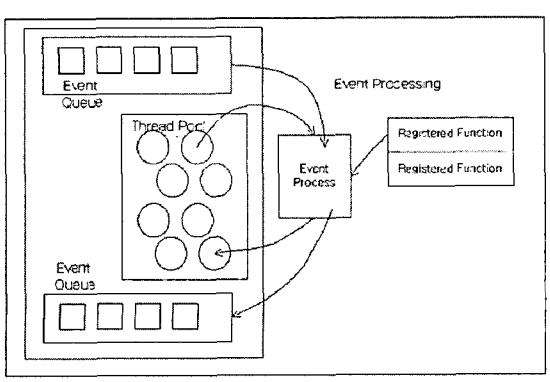


그림 9 네트워크 통신 프레임워크

구조는 클라이언트와 서버가 공유하게 된다. 클라이언트가 어떻게 이벤트를 발생시킬지는 클라이언트 구현에 따라 다르며 게임서버 엔진은 클라이언트의 이벤트 전송, 서버에서의 이벤트 수신 및 처리 루틴으로의 연결, 처리 결과의 관련 클라이언트 전송의 서비스를 제공한다. 각각의 이벤트의 정의, 실제 이벤트 처리 루틴의 구현은 게임서버 엔진의 규칙에 따라 게임 프로그래머가 직접 구현하게 된다.

게임서버 엔진은 I/O completion port와 Thread pooling 메커니즘을 이용하여 시스템의 처리 능력을 극대화시켰다. I/O completion port(IOCP)와 Thread pooling 메커니즘은 Windows NT계열의 운영체제에서 제공하는 서비스로 항상 일정수의 Thread로 하여금 CPU가 허용하는 최고의 성능을 유지하며 동작할 수 있도록 해주며, MS Internet Information Server, MS Exchange Server, MS SQL Server등의 Windows 계열 서버 응용에서 성능향상을 위해 사용되는 메커니즘이다. 게임서버 엔진은 Thread pooling 등과 같은 시스템 자원을 게임 개발자에서 일관된 인터페이스로 제공하며, 게임 개발자는 주어진 인터페이스를 사용하여 게임을 구성하게 된다. 기본적으로 필요한 정보는 이벤트 안에서 제공되며 다른 기본적인 정보, 즉 다른 객체 등에 대한 정보 역시 게임엔진의 런타임 자료구조로 제공된다. 네트워크를 통해 서버에 이벤트가 전달되면 게임엔진은 Thread pool에서 가용 Thread가 있는가를 먼저 확인한다. 가용 Thread가 있으면 계속 진행하고 아니면 잠시 기다린다. 그리고 가용 Thread를 Thread pool에서 꺼내와서 현재 이벤트 큐의 이벤트를 넘겨준다. Thread는 이 이벤트의 처리함수를 수행하고 처리 종료 후에 다시 pool로 돌아간다. 이와 같은 Thread pool을 실제로 구현하기 위하여 IOCP를 사용하였다. 그러나 게임 개발자에게 이러한 구조는 숨겨지며, 단지 등록된 이벤트 처리 함수와 이벤트의 데이터 구조만으로 기본적으로 필요한 게임 진행을 처리하도록 하였다.

나. 게임서버 프레임워크

1) 관심영역 관리

서버는 공유상태의 변화를 사용자에게 알릴 필요가 있을 때 그 변화가 일어난 곳 주위의 특정 영역 내에 있는 사용자에게만 공유상태를 보냄으로써 공유상태의 전체 전송률을 크게 줄일 수 있다. 이때 공유상태를 보내는 영역을 관심영

역(Area of Interest)이라 한다. 서버 엔진에서는 [그림 10]과 같은 셀 기반의 관심영역 관리 방법을 사용한다. 셀 기반 관심영역 관리에서는 게임 공간을 셀이라고 하는 단위 영역들로 나누어서 어느 특정 셀에서 발생된 공유상태의 변경을 그 것 주위에 있는 셀들 내에 있는 사용자들에게만 보내주는 것이다. 그러면, 모든 사용자들의 거리를 일일이 검사할 필요 없이 공유상태를 보내야 하는 사용자에게 바로 보낼 수 있다. 이를 위해서 게임서버는 사용자, 몬스터 등과 같은 공유객체들을 셀 별로 따로 관리하며 각각의 셀들은 자신의 주위의 셀들의 리스트를 가지고 있다.

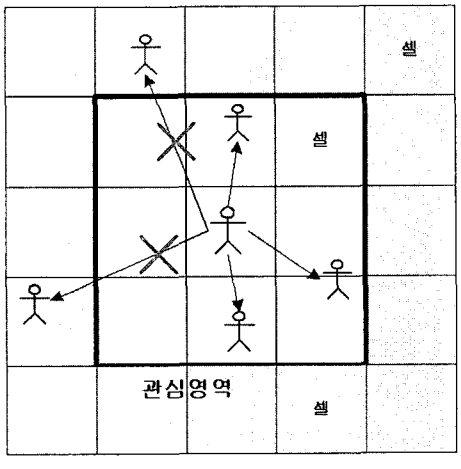


그림10 셀 기반의 관심영역 관리

2) 데드레커닝 알고리즘

게임 상에서 수많은 사용자들은 다양한 동적인 상태들을 서로 공유하고 있기 때문에 이러한 공유상태들을 일관성 있게 유지시켜 줄 필요가 있다. 하지만, 네트워크 상에서 다양하게 분포되어 있는 여러 클라이언트 시스템과 서버 시스템 간의 네트워크 지연과 대역폭의 제한으로 인해 모든 사용자들의 공유상태들이 모두 동일하도록 완벽하게 일관성을 유지시켜주는 것은 거의 불가능하다. 일반적으로 눈으로 보이는 화면상의 거의 완벽한 일관성을 위해서는 초당 30번 이상 공유상태들을 주고받아야 한다. 이 경우에 서버는 네트워크 대역폭의 제한 때문에 게임서버는 많은 사용자들을 수용해줄 수 없을 것이다. 반면에, 공유상태들의 전송률을 지나치게 줄이면 게임서버는 많은 사용자들을 수용할 수는 있겠지만 일관성이 실시간으로 유지되지 않으므로 사용자들이 서로 다른 상태를 가지게 되어 게임을 제대로 진행해 나갈 수 없게 된다.

이와 같이 서버 성능(서버의 최대 수용가능 동시사용자)과 게임상태의 일관성은 둘 다 높을수록 좋지만 서로 역비례 관계에 있기 때문에 양쪽을 모두 고려하여 공유상태들의 전송률을 적절히 유지시켜줄 필요가 있다. 이를 위하여 게임엔진에서는 PC 객체 및 기타 공유객체들의 위치에 대해 데드레커닝 알고리즘(Dead Reckoning Algorithm)[9]을 지원한다.

데드레커닝 알고리즘에서 공유상태를 변화시킨 사용자는 다른 사용자들에게 특정조건이 만족될 때에만 변화된 상태정보를 보내준다. 그리고 변화된 상태정보를 받는 사용자들은 불연속적으로 받은 상태정보를 이용하여 연속적인 값을 만들어 사용한다.

공유상태를 변화시키는 사용자(공유상태를 보내는 사용자)는 과거에 보낸 상태정보(상태값, 변화율 등)를 근거로 하여 추정 알고리즘(Tracking Algorithm)을 이용하여 다른 사용자들이 추정하고 있을 공유상태를 계속해서 추정한다. 그리고 계속해서 추정된 상태값과 실제 상태값과의 오차를 검사하여 이 오차가 미리 정해진 문턱값 이상이 되면 다른 사용자들에게 새로운 상태정보(상태값, 변화율 등)를 보낸다. 따라서 불필요한 상태정보의 전송을 막을 수 있으며 문턱값을 조정해줌으로써 상태정보의 전송률을 제어해줄 수도 있다. 문턱값이 크면 클수록 사용자들 간의 공유상태 오차는 커지지만 상태정보의 평균 전송률은 떨어진다. 반면에 문턱값이 작으면 작을수록 공유상태의 오차는 줄지만 평균 전송률이 늘어나게 된다.

한편, 변화된 공유상태를 받는 사용자는 과거에 받은 공유상태를 근거로 하여 현재의 공유상태를 계속해서 추정하다가 새로운 상태정보를 받으면 공유상태를 보정해준다. 이때 불연속적인 상태값이 변하는 것을 막기 위하여 수렴 알고리즘(Convergence Algorithm)을 사용하여 공유상태를 갱신해준다.

4. 게임엔진의 분석 및 평가

제안된 게임엔진은 객체 기반의 설계를 토대로 엔진을 체계화하고 모듈화하여 엔진에 대한 수정이나 확장이 용이하게 함으로서 활용도를 높였으며, DirectX 기반으로 개발되어 게임에 최적화된 빠른 속도 및 최신의 3D 기술을 반영한

업그레이드가 용이하다는 장점을 가지고 있다. 특히 렌더링 엔진은 다양한 객체 출력 및 지형에 영향을 받지 않는 동적 그림자 생성이 가능하며 텍스춰를 활용한 다양한 연출, 동적 광원 제어, 파티클을 이용한 다양한 특수효과 연출 기능 등을 제공한다. 또한 복잡한 인도어(indoor) 환경까지 실시간으로 처리가 가능하도록 되어 있다. 애니메이션 엔진은 모션의 실시간 처리를 통하여 자연스러운 애니메이션이 가능하며, 물리 현상에 대한 현실적인 시뮬레이션이 가능하도록 되어 있다. 사운드 엔진은 2 채널 기반 및 스트리밍 기반으로 실시간 3D 사운드 생성이 가능하며, 최적화된 알고리즘에 의하여 CPU 부하가 최소화되도록 하였다. 서버 엔진은 데드레커닝 알고리즘, 쓰레드 풀링(thread pooling) 등을 사용하여 통신 및 메시지 처리에 있어서 우수한 성능을 가지며, 셀 기반의 관심영역 관리 및 공유객체 관리를 사용하여 상태관리 및 이벤트 처리에 유리하도록 되어 있다. [표 1]은 제안된 엔진인 Dream3D와 상업적으로 성공한 대표적인 엔진인 Unreal, LithTech, NetImmerse를 중심으로 주요 기능에 대하여 분석을 실시한 결과를 나타낸다.

	세부기능(Dream3D)	비교		
		Unreal	LithTech	NetImmerse
렌더링 엔진	ㅇ 3차원 객체 출력기	O	O	O
	ㅇ 게임 환경 출력기	O	O	O
	ㅇ 광원 및 그림자 출력기	O	O	O
	ㅇ 카메라 제어기	O	O	O
	ㅇ 특수효과 출력기	O	O	O
	ㅇ 텍스춰어 처리기	O	O	O
애니메이션 엔진	ㅇ 애니메이션 객체 처리기	O	O	O
	ㅇ 공격, 스킨 처리기	O	O	O
	ㅇ 키프레임 기반 모션 처리기	O	O	O
	ㅇ 얼굴 애니메이션 처리기	x	x	x
	ㅇ 다양한 객체 애니메이션 처리기	O	O	O
	ㅇ 물리 엔진	O	x	O
서버 엔진	ㅇ 서버 쓰레드 풀 관리기	x	x	x
	ㅇ 클라이언트/서버 이벤트 처리기	x	x	x
	ㅇ 셀 기반 관심영역 관리기	x	x	x
	ㅇ 데드레코닝 알고리즘 구현	x	x	x
	ㅇ 분산 서버 최적화	x	x	x
사운드 엔진	ㅇ 2채널 기반 3D 음상 정위기	Δ	Δ	Δ
	ㅇ 3D 음장 제어기	x	x	x
	ㅇ 다이나믹 사운드 믹서기	O	O	O
	ㅇ 음원모델/음장모델 처리기	Δ	Δ	Δ
	ㅇ 특수음향 효과 처리기	O	O	O

표 1 게임엔진의 기능 분석 (O:지원, Δ:일부지원, x:지원안함)

5. 결 론

본 연구에서는 다중참여자용 롤플레잉 게임 제작을 위한 온라인 3D 게임엔진을 제안하였다. 제안된 게임엔진은 렌더링 엔진, 애니메이션 엔진, 사운드 엔진, 서버 엔진 등의 세부 엔진으로 구성되어 있으며 다음과 같은 특징을 가지고 있다.

- 다수의 동시 사용자 환경의 Full 3D 온라인게임 지원
- 그래픽 카드와 DirectX 8.x에 최적화된 3D 그래픽 기능
- 모션 DB 기반의 자연스러운 애니메이션 지원
- 2 채널 기반의 실시간 3D 사운드 생성
- 확장성이 용이한 다중/분산 서버 방식 채용
- 객체 기반 설계를 토대로 한 엔진의 체계화, 모듈화
- 다양한 장르의 게임 컨텐츠 개발 가능

제안된 게임엔진 기술을 게임 컨텐츠 개발에 활용한다면 개발기간 단축 및 비용 절감의 효과를 얻을 수 있을 것이다. 또한 개발된 요소 기술들은 게임 분야 뿐만 아니라 애니메이션, 영화 등 멀티미디어 컨텐츠 관련 분야에서도 폭넓게 적용될 수 있으리라고 본다.

그 동안의 게임기술은 게임 플랫폼에 따라 분리되어 성장하였고, 영역도 확연히 구분되었다. 예를 들면, 게임 콘솔의 그래픽과 제작 기술은 PC 기반의 게임 그래픽 기술과 전혀 다른 형태를 취하고 있어서 플랫폼간의 기술적 장벽이 높은 편이었다. 그러나 게임 플랫폼별로 다소 차이가 있지만 한 게임당 수십억원이 들어가는 게임 제작 예산의 증가와 한 종류의 게임 플랫폼에서 상업적으로 성공한 게임 타이틀이 다른 플랫폼에서도 성공하는 경우가 많은 시장적 요인으로 멀티플랫폼 지원 게임 기술이 중요시될 것이다. 제안된 게임엔진은 향후에 PC 환경 뿐만 아니라 콘솔 등을 함께 지원하는 멀티플랫폼 지원 게임엔진으로 확장되어 개발될 필요가 있으며, 상업용 컨텐츠 개발을 통한 검증이 이루어져야 할 것이다.

참고문헌

[1] 이헌주, 김준애, 임충규, 김현빈, "온라인 3D 게임엔진 표준화", 한국정보처리학회지, 2002년 5월호.

[2] D. H. Eberly, "3D Game Engine Design: A Practical Approach to Real-Time Computer Graphics", Morgan Kaufmann Publishers, 2001.

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

[4] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Mesh Optimization", SIGGRAPH 1993, pp 19-26, 1993.

[5] H. Hoppe, "Progressive Meshes", SIGGRAPH 1996, pp 99-108. 1996.

[6] P. Lindstorm, D. Koller, W. Ribarsky, L. F. Hidges, N. Faust, and G. A. Turner, "Real-Time, Continuous Level of Detail Rendering of Height Fields", SIGGRAPH 1996, pp 109-118, 1996.

[7] Tae-Joon Park, Soon Hyoung Pyo, Chang Woo Chu, and Byoung Tae Choi, "Design and Implementation of a Rendering Engine for Developing Computer Games", Japan Korea Computer Graphics Conference 2001 proceedings, 2001.

[8] D. R. Begault, "3-D Sound for Virtual Reality and Multimedia", New York, Academic Press Inc., 1994.

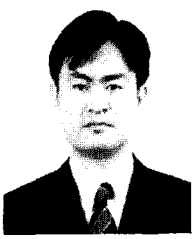
[9] M. R. Macedonia, M. J. Zyda, D. R. Pratt, P. T. Barham, "NPSNET: A Network Software Architecture for Large-Scale Virtual Environments", Presence: Teleoperators and Virtual Environments, Vol. 3, No. 4, 1994.

[10] Institute of Electrical and Electronics Engineers, International Standard, ANSI/IEEE Standard 1278-1993, Standard for Information Technology, Protocols for Distributed Interactive Simulation, March 1993.

[11] Andrew Rollings, Dave Morris, "Game Architecture and Design," Coriolis, 2000.

[12] A. Watt, F. Policarpo, "3D Games:Real-time Rendering and Software Technology", Addison-Wesley, 2001.

[13] 박태준, 표순형, 추창우, 최병태, "3D 게임용 렌더링 엔진의 제작", HCI 2002 학술대회 논문집, 2002.



박 태 준

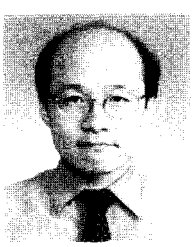
1992년	한국과학기술원 전산학과 학사
1994년	한국과학기술원 전산학과 석사
1999년	한국과학기술원 전산학과 박사 (컴퓨터그래픽스 전공)
1999년~2000년	텔아비브 대학교 객원연구원
2000년~현재	한국전자통신연구원 선임연구원
2003년~현재	한국전자통신연구원 가상현실연구부 영상컨텐츠연구팀장
관심분야	컴퓨터그래픽스, 가상현실, 3차원 게임, 영화 및 애니메이션, 영상 특수효과

e-mail: ttjjpark@etri.re.kr

이 헌 주

1991년	중앙대학교 컴퓨터공학과 학사
1993년	중앙대학교 컴퓨터공학과 석사
1998년	중앙대학교 컴퓨터공학과 박사
2001년~2002년	Iowa State University (Post-Doc.)
1998년~현재	한국전자통신연구원 선임연구원
관심분야	인공지능, 가상현실, 게임엔진

e-mail: hjoo@etri.re.kr



김 현 빈

1985년	중앙대학교 학사 (응용통계학 전공)
1988년	중앙대학교 대학원 석사 (응용통계학 전공)
1996년	Okayama Univ. 대학원 박사 (전산통계학 전공)
1991년3월~1993년3월	Nagoya Univ. (연구원)
1984년11월~현재	한국전자통신연구원 (부장/책임연구원)
관심분야	가상현실, 게임엔진, HCI

e-mail: hbkim@etri.re.kr