Journal of Digital Contents Society Vol. 8 No. 4 Dec. 2007(pp. 579-586)

온라인 3D 게임의 엔진 테크놀러지 디자인

최학현*, 김정희**

요 약

온라인 3D 게임 엔진 개발은 엔진 세부설계와 개발, 알파테스트, 수정 및 보완, 베타테스트, 출시 등의 개발 공정과정을 거치며, 온라인 3D 게임 엔진은 서버 엔진과 클라이언트 엔진으로 나뉜다. 서버 엔진은 Linux 에서 개발되며 비대칭 다중 서버 구축을 위한 서버구축 파트, 분산처리 파트, DB 구축파트로 각각 구성하고 클라이언트 엔진은 윈도우즈에서 DirectX 를 이용하여 개발하며 그래픽 파트, 사운드 파트, AI 파트, 객체처리 파트 파트로 각각 구성하고 엔진 공통기술인 네트워크 기술을 분석하여 3D엔진기술을 기반으로 엔진구조 및 구성하는 방식을 제안하였다.

Engine Technology Design of On-line 3D Game

Hak-Hyun Choi*, Jung-Hee Kim**

Abstract

The process of developing a online 3D game engine involves designing details of the engine, alpha test, adjustment, supplementation, beta test and introduction, and an online 3D game engine is made of a server engine and client engine. The server engine, which is made of server construction part, distribution center and DB construction part for building asymmetric multiple servers, is developed by Linux, and the client engine, which is made of a graphic part, sound part, AI part and object management part, is developed by using DirectX in Windows, and it proposes an engine configuration and method based on 3D engine technology by analyzing the network technology, which is a common engine technology.

Keywords : 3D Game, On-line Technology, 3D Engine Design

1. 서론

게임 엔진이란, 그래픽과 사운드 등의 소프트웨어 기술을 결합해 게임에 적용하기 쉽도록 미리 만들어 놓은 핵심 프로그램으로, 해외의 경우 '퀘이크'와 '언리얼'이 대표적인 게임 엔진으로 자리 잡고 있으며, 개발사들의 엔진에 대한 아웃소싱도 일반화되었다. 국내서는 게임콘텐츠 개발인력이 게임 엔진까지 개발, 엔진개발에 대한 전문성이나 노하우가 결여되고 있다는 지적을 받아왔으나, 최근 게임엔진의 시장성이 검증되면서 전문 개발사들이 잇달아 등장하였다. 한편, 관련 정부기관들도 각각 주관하에 온라인게임 엔진 개발 사업이 추진되고 있고, 게임엔진 등 기반기술 개발 사업이 추진되고 있다. 따라서 본 논문에서는 온라인게임 엔진 중 서버 엔진의 네트워크 파트, 비대칭 다중 서버 구축을 위한 서버 구축 파트, 분산처리 파트, DB 구축파트, 클라이언트 엔진의 그래픽 파트, 사운드 파트, AI 파트, 객체처리 파트, 엔진 공통기술인 네트워크 기술 파트, 엔진 시스템 구조 파트를 분석을 통하여 온라인게임 엔진기술을 기반으로 엔진설계에 관해 고찰한다.

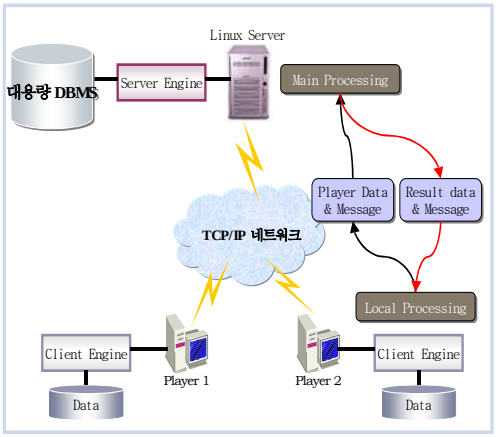
※ 제일저자(First Author) : 최학현
접수일자:2007년09월18일, 심사완료:2007년09월28일
* 서울여자대학교 미디어학부
chh@swu.ac.kr
** 성균관대학교 대학원
▣ 이 논문은 2006학년도 서울여자대학교 컴퓨터과학연구소 교내학술연구비에 의해 지원되었음.

2. 게임 엔진 기술

2.1 전체 시스템 구조

온라인 게임의 전체 시스템 구조는 완벽한

2-Tier Client/Server 구조로 서버와 클라이언트는 각각의 엔진을 이용하여 게임진행을 위한 다양한 프로세싱을 하고 필요한 데이터를 적절히 분산하여 처리한다. Linux 기반의 서버는 클라이언트들의 데이터 처리, 메시지 전달, 동기화 등의 주요 프로세싱을 담당하고, Windows 기반의 클라이언트는 I/O를 비롯한 많은 핵심 프로세싱을 담당한다(그림 1).



(그림 1) 전체 시스템 구조

2.2 엔진기술

게임을 개발하기 위한 핵심기술로 전체 시스템 구조를 처리하는 서버쪽의 객체지향 엔진개발 기술, 대용량 데이터처리 기술, 분산처리 기술 등이 있고 클라이언트 쪽의 그래픽 처리기술, 3D 동영상 제작 기술, 사운드 처리기술 등이 있으며 공통적으로 네트워크 처리기술을 포함한다.

가. 서버 엔진기술

① 객체지향 엔진개발 기술(엔진의 확장성 및 재사용성 극대화)
② 대용량의 공간, 비공간 데이터 처리기술(대용량 DBMS를 이용한 저장, 검색 및 처리)
③ 비대칭 다중서버를 위한 분산처리 기술(맵에 따른 서버분할로 서버기능 극대화)
④ 사용자 인증과 데이터 보안처리 기술(서버 Hacking 및 데이터 조작방지)

나. 클라이언트 엔진기술

① 3D 그래픽 처리기술(3D Max를 이용한 캐릭터와 배경 제작 및 Alpha Blending 효과 처리)
② 3D 동영상 제작기술(Motion Capture 사용으로 부드러운 영상처리)
③ MP3를 이용한 사운드 처리 기술(MP3 포맷의 높은 압축률과 Quality를 가진 사운드 처리)
④ 인공지능처리 기술(NPC 및 Monster들의 인공지능 강화)
⑤ 최적화 알고리즘 구현 기술(A* 알고리즘, 개체충돌, 압축 등의 처리)

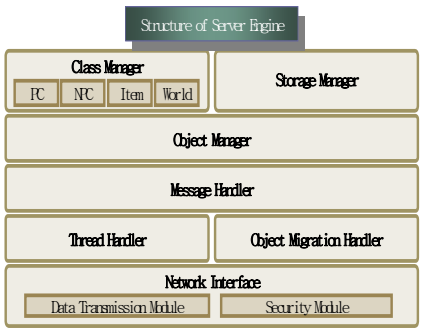
다. 엔진 공통기술

TCP/IP 기반의 실시간 네트워크 처리기술(소켓을 이용한 Client/Server 통신처리)

2.3 엔진설계

가. 서버엔진

전체시스템의 핵심이 되는 서버엔진은 다음과 같다(그림 2, 3).



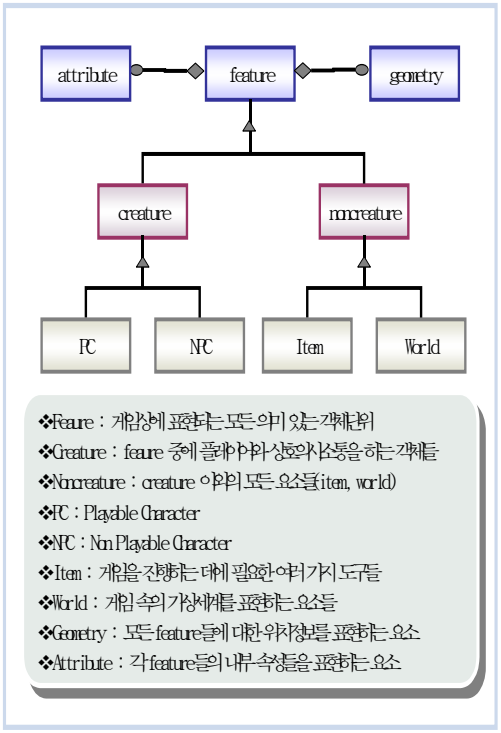
(그림 2) 서버엔진 구조

Class Manager	본 게임 속에 존재하는 모든 객체를 크게 PC, NPC, Item, World로 클래스화 하여 정의하고 이를 처리한다.
Storage Manager	클래스화 된 객체저장, 플레이어에 대한 정보저장, 관리 및 효율적인 색인방법을 이용한 검색을 담당한다.
Object Manager	본 게임 속에 있는 모든 객체를 통제하는 부분으로 Message Handler에서 넘어온 데이터를 처리한다.
Message Handler	플레이어로부터 받은 정보를 해석하고 Object Manager를 호출하여 메시지를 넘겨준다.
Thread Handler	다수의 플레이어를 관리하기위해 Thread를 생성하여 플레이어간의 동기화 및 데이터공유 문제 등을 처리한다.
Object Migration Handler	비대칭형 분산서버 구조를 사용함으로써 생기는 객체 이전현상을 처리하는 부분으로 객체이전은 플레이어가 게임 중에 서버를 옮길 때 발생한다.
Network Interface	실제 클라이언트와 서버간의 통신을 담당하는 부분으로 크게 데이터 전송파트와 보안파트로 나누어진다.

(그림 3) 서버엔진 구성요소

(1) 객체지향 데이터 모델링

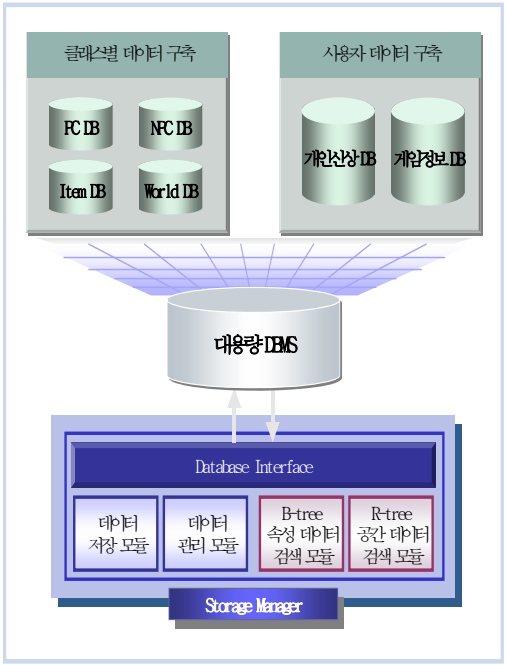
객체지향 엔진을 개발하기위해 먼저 구현 가능한 모든 객체 데이터들에 대한 모델링을 하여 클래스로 구현한다. 이러한 객체 지향적인 개발은 엔진의 확장성과 재사용성을 극대화함으로써 플레이어들이 게임을 만들어나가는 온라인 롤플레잉 게임에 가장 적절하다. 모든 객체는 하나의 Feature로 표현이 되는데 이는 공간적인 속성인 Geometry와 비공간적인 속성인 Attribute들로 이루어지고 Feature로부터 상속을 받는 Creature / Noncreature와 또 각각으로부터 상속을 받는 플레이어 정보를 표현하는 PC, NPC, Item과 게임 속의 가상세계를 표현하는 World로 구분됩니다. 이러한 모델링에 대한 모식도는 다음과 같다 (그림 4).



(그림 4) 데이터 모델링

(2) 대용량 DBMS를 이용한 데이터 처리

서버를 운영하는데 필요한 대용량의 모든 데이터를 효율적으로 구축하고 사용하기 위해 객체 지향적인 데이터베이스 구조가 필요하다. 이에 따라 클래스별로 기본데이터를 구축하고 사용자 데이터를 구축한다. 대용량 데이터베이스에서 필요한 데이터를 가져와서 클라이언트들의 요구를 빠르게 처리하기 위해 성능이 뛰어난 색인 검색방법을 사용한다. 서버엔진의 Storage Manager에서 이 부분을 담당하게 되며 비공간 데이터에 대해서는 B-tree 색인방법을,공간데이터에대해서는 MBR(Minimum Boundary Rectangle)을 이용하는 R-tree 색인방법을 이용하여 검색한다(그림 5).

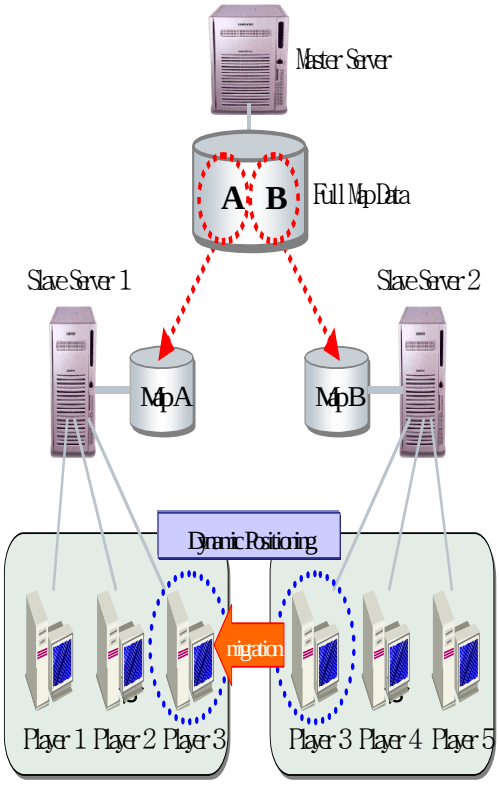


(그림 5) 데이터 처리

(3) 다중 사용자를 위한 비대칭형 분산서버 구조

현재 온라인 롤플레잉 게임에서 가장 중요한 점은 동시에 많은 사용자가 서버에 집중되어 발생하는 서버성능 저하 및 마비에 대한 대처 방안이다. 이를 해결하기 위해 저희 게임에서는 비대칭형 분산서버 구조를 이용한다. 이 구조는 각 게임서버에서 맵 데이터를 일정크기로 나누어 가지고 각 Map에 접속하는 플레이어들을 관리하는 형태이다. Master 서버는 초기 접속을 받아들여 해당 서버에 연결시키고 다른 모든 Slave 서버들을 관리하는 역할을 하며, 각 Slave 서버는 현재 자신에게 접속해 있는 플레이어들

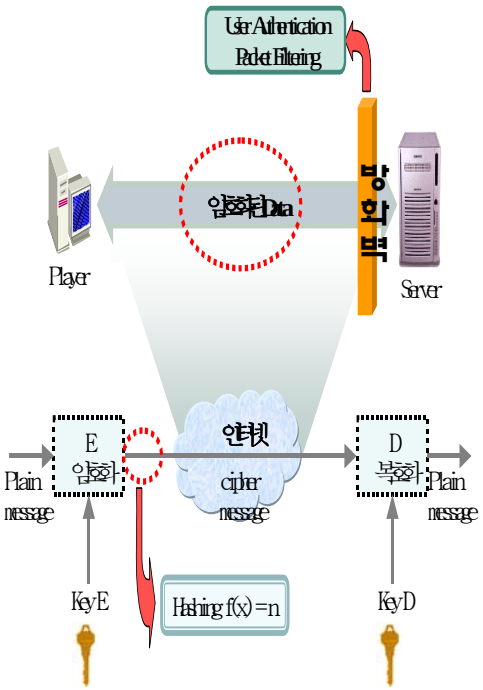
을 관리한다. 플레이어가 다른 서버에 있는 Map으로 이동할 경우에는 플레이어의 모든 정보를 이동할 서버로 이전시키는 객체이전 현상(Object Migration)이 발생한다. 이와 같은 구조는 맵에 따라 서버가 분리되기 때문에 시나리오와 서버의 확장성이 매우 높고 각 서버간의 의존성을 최소화하여 서버간의 동기화 부담을 줄일 수 있는 장점을 갖고 있으므로 전체적인 운영의 효율성을 높일 수 있다(그림 6).



(그림 6) 비대칭형 분산서버 구조

(4) 서버 및 데이터 보안

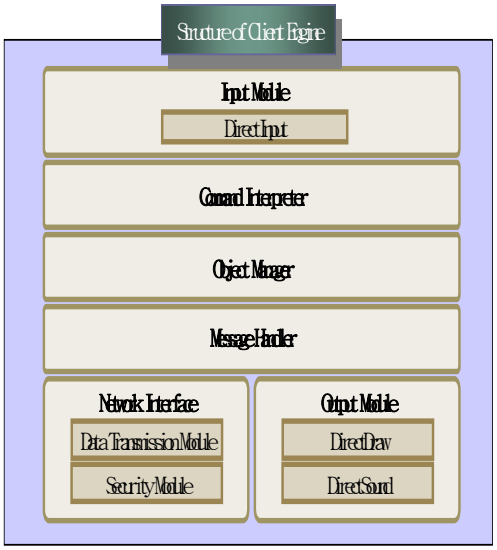
현재 온라인 롤플레잉 게임에서는 서버해킹은 물론이고 각종 데이터 조작 및 변조행위가 빈번하게 일어나고 있다. 이를 대비해 각 서버에 대한 임의의 접근을 막기 위해 방화벽을 설치하고 데이터 전송 중에 발생하는 변조, 위조를 막기 위해 다단계의 암호화 방법을 이용한다. 이로서 서버자체 보안은 물론 데이터보안까지 할 수 있게 된다. 데이터 암호화는 대칭키 암호화 방식과 해시함수를 동시에 사용한다. 대칭키 암호화는 암호화 키와 복호화 키가 동일한 것으로 암호화 및 복호화 속도가 빨라 메시지 전송을 주로 하는 온라인 롤플레잉 게임에 적용하기에 가장 적합하다. 해시함수는 대칭키 암호화 방식에서 키의 노출로 인한 문제점을 막기 위해 대칭키 암호화 방식으로 암호화된 메시지를 다시 변환하게 된다. 이러한 2단계 암호화 방식을 사용함으로써 스니핑, DOS(Denial of Service)등의 해킹 및 로컬 데이터 변조 등을 차단하여 서버와 클라이언트 상호간에 신뢰할 수 있는 데이터 전송이 이루어지게 된다(그림 7).



(그림 7). 서버 및 데이터 보안

나. 클라이언트엔진

게임의 클라이언트 엔진은 캐릭터나 배경에 대한 복잡한 속성 및 공간 데이터 계산을 자체 처리 함으로서 플레이어에게 빠른 응답속도를 지원합니다. DirectX를 이용하여 플레이어의 입력으로부터 직접 처리할 수 없는 부분은 최소화하여 서버로 데이터를 전송하고 결과를 받아 처리한다(그림 8, 9).



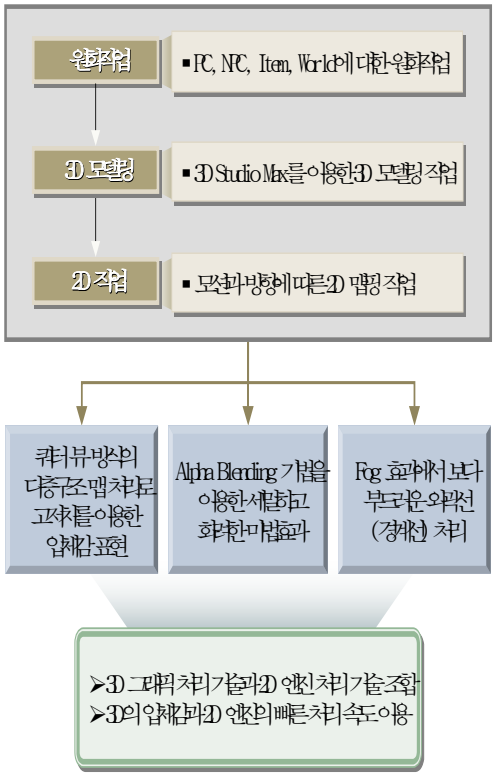
(그림 8) 클라이언트엔진 구조

Input Module	플레이어로부터 마우스나 키보드 입력을 받아 DirectInput을 이용하여 처리한다.
Command Interpreter	Input Module에서 받은 모든 명령이나 메시지를 해석하고 Object Manager에게 전달한다.
Object Manager	플레이어의 입력을 클래스화 된 객체단위로 처리한다.
Message Handler	Object Manager에서 처리된 정보를 서버에 전송하기 위해 규정된 protocol에 맞게 메시지를 처리한다.
Network Interface	실제 클라이언트와 서버간의 통신을 담당하는 부분으로 크게 데이터 전송파트와 보안파트로 나누어진다.
Output Module	플레이어의 입력에 대한 결과를 DirectDraw, Direct3D 및 DirectSound를 이용하여 처리한다.

(그림 9) 클라이언트엔진 구성요소

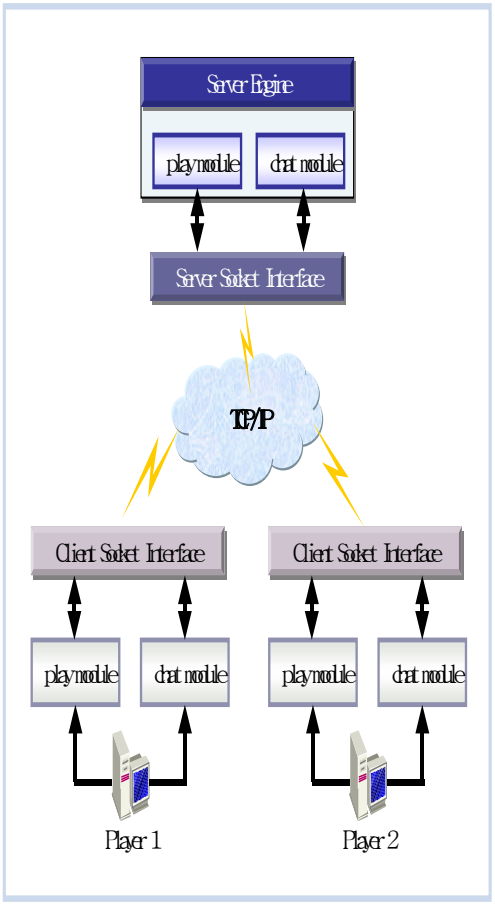
다. 3D 방식을 이용한 그래픽 처리

게임은 3D로 입체적이고 사실적인 게임화면을 보여준다. 모든 캐릭터와 배경은 원화작업 후에 3D Polygon 처리와 랜더링 작업을 거쳐서 3D로 표현된다. 캐릭터는 세밀한 프레임 처리로 사실적이고 부드러운 동작표현이 가능하며 배경은 쿼터 뷰 방식의 다층구조로 현실감 있는 고저차를 표현한다. Alpha Blending 기법을 사용하여 세밀하고 화려한 마법 효과를 나타내고 Fog 효과에서 보다 부드러운 외곽선 처리를 한다. 게임은 다양한 컬러와 라이트들의 조합과 3D 그래픽 처리 기술의 이용으로 2D 엔진처리의 단점을 극복해서 Full 3D 로 화려하고 세밀한 그래픽을 보여준다(그림 10).



(그림 10) 그래픽처리

플레이어들의 데이터를 서버에 전송하고 처리결과를 받아오기 위해 Socket Protocol을 이용한다. 서버 엔진과 클라이언트 엔진은 각각의 소켓으로 만들어진 Network Interface를 통해 서로 통신한다. 부가적으로 이 부분에서 데이터 압축으로 전송량을 줄이거나 보안 프로토콜을 이용하게 된다. 통신부분은 크게 Play Module과 Chat Module 두 가지로 나뉜다. Play Module은 실제 게임진행시에 필요한 데이터를 처리하는 부분이고, Chat Module은 플레이어들간의 대화를 처리하는 부분이다. Play Module에서의 처리결과는 같은 영역 안에서 접촉이 이루어지는 플레이어들에게 전송되며, 플레이어간의 대화내용은 영역 안의 모든 플레이어에게 Broadcasting 된다(그림 11).



(그림 11) Client-Server 통신처리

라. 시스템 요구사항

누구나 쉽게 인터넷을 통해 접속하여 간편하게 프로그램을 다운로드 받거나 배포 CD를 통해서 설치, 실행할 수 있으며 일반 성능의 PC에 최적화 되어 개발된다. DirectX는 기본으로 설치되어 있어야 하며 빠른 데이터 전송처리를 위해 ADSL 이상 또는 LAN 환경이 요구된다.

3. 3D 게임엔진 구성 방식

3D그래픽 데이터들은 평면 이미지가 아니다. 여러 개의 폴리곤으로 이루어진 하나의 입체이므로 컴퓨터 화면대 입체의 크기가 거리에 따라서 변하게 된다. 또한 카메라를 움직이면 물체의 보는 각도에 따라서 생김새가 변화함으로 2D그래픽처럼 일반적으로 처리하기가 힘들다. 3D그래픽은 2D그래픽처럼 컴퓨터상의 해상도의 수치(640*480, 800*600, 1024*768) 단위를 좌표나 크기의 기준으로 이용하지 않고 물체의 텍스쳐(이미지)와 물체를 1부터 0사의 비율값으로 처리할 수 있다.

(1) 그래픽 플러그인(Garaphic Plugin)

2D그래픽 DB는 2D Graphic Case Tool을 사용하여 그래픽 이미지 파일을 얻어서 바로 가공해서 쓰지만 3D그래픽 DB는 3D Graphic Case Tool 자체에서 게임엔진이 원하는 형태로 DB를 가공해서 Export할 수 있다. 이러한 3D Graphic Case Tool에 작업한 내용을 자신의 게임엔진에 맞게 Export/ Import해줄 수 있게 플러그인을 설계할 수 있다.

(2) 캐릭터 저작(Character Editor)

3D그래픽 DB를 플러그인(Plugin)으로 통해서 얻었다면 그 DB만을 가지고 다시 재수정 작업을 할 수 있게 해주는 도구이다. 3D그래픽 도구로 제작된 캐릭터 혹은 3D 물체가 게임에서 불러들일 때 개발자가 의도한 것과 다른 경우가 생긴다. 이러한 이유는 3D 그래픽 도구로 제작한 결과 소스들은 상당히 복잡한 내용들과 DB들이 섞여 있는 것에 반해서 게임에서는 3D 형태나 질감을 이루는 DB등 몇 가지만 이용하므로 3D그래픽 도구가 처리해준 특별한 기능들도 함께 게임엔진에서 구현되지는 않기 때문이다. 그래서 3D그래픽 도구에서 제작한 결과물과 그 결과물을 Plugin을 통해서 Export한 DB를 게임에서 불러 왔을 때의 차이점을 게임을 직접 실행하기 전에 이 Character Editor를 이용하여 미리 확인해 볼 수 있고 필요한 부분을 수정도 가능하게 하는 것이 에디터의 목적이다. 또 3D그래픽 도구를 사용하지 않고 Character Editor를 사용하여 직접 3D물체들을 만들 수 있게 설계할 수 있다.

(3) 레벨 저작(Level Editor)

2D 그래픽 게임에서의 배경은 단순히 게임을 이해시키는 평면 이미지지만 3D 그래픽 게임에서는 배경이 공간(입체)으로 이루어져 있기 때문에, 수많은 3D기술들이 필요하다. 레벨 저작에서

Full Map을 사용하는 경우는 많이 없다. 주로 타일처럼 영역을 자르거나 임의의 크기로 영역을 분할 시켜 놓는다. 그렇게 제작하는 이유는 공간에다가 수많은 물체들을 입체 형태로 설치를 해야 하고 각 설치된 물체들과 상호간의 충돌처리를 해야만 한다. 특히 높이 차가 있는 게임속 지형을 배경으로 삼을 경우 고도에 따라서 캐릭터들이 지면 높이에 따라 위치도 조정해 주어야 함으로 캐릭터들과 배경은 언제나 서로 충돌을 느끼는 것이다. 이렇게 되었을 때 모든 지역을 하나의 캐릭터와 비교해서 충돌 테스트를 할 경우 컴퓨터가 무리하게 연산을 하게 되어 컴퓨터 H/W에 막대한 연산 부하를 주게 된다. 이것을 피하고자 영역을 분할해서 컴퓨터의 처리능력 부담을 줄일 수 있다. 영역을 분할하는 기본 처리는 카메라를 통해서 보이는 곳 이외의 모든 부분은 컴퓨터가 처리하지 못하게 하는 것인데 배경만 해도 수만개에서 수백만개의 폴리곤들로 이루어져 있다. 이것을 일일이 다 체크하기에는 역시 컴퓨터 H/W에 무리가 따르므로 전체 레벨을 몇 개의 블록으로 나누어 관리하면 해결이 쉽다. 즉 레벨을 어떻게 분할해야 하는지와 어느 곳에 어떤 입체물들과 캐릭터들을 설치해야 하는 지의 결과들을 게임속에서 실행해보기 전에 Level Editor를 통해서 체크하고 작업할 수 있게 설계할 수 있다.

*** 3D 게임엔진 구성 방식**

- 3D Render Create
- 3D Font Create
- Object Model Create & System
- Animation Create & Define
- Animation Key Frame Create
- Time Information Create
- Camera Path
- Camera Control Technique
- Camera Special Effect Technique
- Orthographic Projection Mode Create
- Perspective Projection Mode Create
- Free View Mode Create
- FOV(Field Of View) Handle
- Vertex Control & Effect
- Texture Mapping Technique
- Ray Tracing Technique
- Radiosity Technique
- Multi Texture Mapping Technique
- Environment Mapping Technique
- Bump Mapping Technique
- Shadow Mapping Technique
- Minmap Technique
- Light Technique
- 기타 Effect System

4. 결론

3D 게임엔진으로는 “Genesis 3D” 라는 전체 소스가 오픈되고 무료인 공개 엔진이 있으나 분석에 적합한 엔진이 별로 없다. 2D 게임엔진은 대부분이 엔진이라고 하기보다는 일부분 기능별 라이브러리라고 보는 게 적당하고 또한 특정게임에 맞춰서 개발되어 있기 때문에 활용에는 문제가 많다. 상용엔진은 대부분 소스가 오픈되지 않으며, 공개엔진은 기능성이나 호환성에서 적합하지 않다. 몇 개의 Mud 엔진이 오픈되어 있으나 초창기 텍스트 머드 엔진 위주라서 역시 적합하지 않으며 보통 게임 소스를 보면 수만 라인 이상이다. 따라서 기존 공개 소스를 분석하는 데는 시간도 너무 오래 걸리고 그만큼 효과를 보기 어렵다고 생각된다. 따라서 국산게임에 맞게 새로운 엔진기술을 정립하는 전략이 필요하다고 보고, 본 논문 연구를 통해 앞으로 상품성과 경쟁력 있는 3D 게임엔진을 만들 수 있으려면 어떠한 요소기술과 설계방식들이 요구되어진다는 것을 알게 되었다. 그것들을 잘 응용할 수 있다면 시장성 있는 3D 게임엔진을 개발할 수 있을 것이라 기대한다.

- 다양한 분야의 온라인 3D게임 서버 제작
- 온라인 3D게임 서버의 손쉬운 운용
- 게임 개발자가 3D게임 콘텐츠 개발에 집중
- 온라인 3D게임 서버의 안정성 및 성능에 대한 지속성
- 다양한 온라인 3D게임 콘텐츠 서비스를 통한 사용자의 증가 효과
- 3D게임서버의 체계적이고 쉬운 구축을 통한 경제적, 시간적 비용의 절감
- 3D게임 상품화를 통한 세계시장 석권이 용이하며 기술적 파급효과가 큼

참 고 문 헌

[1] 최학현・이명학, 게임 프로그래밍, 진한M&B, 2006.
[2] John David Funge, AI for Games and Animation: A Cognitive Approach, A.K. Peters, 1999.
[3] Robert Huebner, "Adding Languages to Game Engine", Game Developer, Sep 1997.
[4] Michael van Lent, John Laird, "Developing an Artificial Intelligence Engine", Game Developer Conference Proceeding, March 1999.
[5] Steve Rabin, "Designing a General Robust AI Engine", Game Programming Gems, Charles River Media, 2000.
[6] Steve Rabin, An Architecture for RTS Command Queuing , Game Programming Gems II, Charles River Media, 2001.
[7] Paul Tozour, Influence mapping , Game Programming Gems II, Charles River Media, 2001.
[8] Paul Tozour, Strategic Assesment Techniques , Game Programming Gems II, Charles River Media, 2001.
[9] William van der Sterren, Terrain Reasoning for 3D Action Games , Game Programming Gems II, Charles River Media, 2001.
[10] Steven Woodcock, Flocking with Teeth , Game Programming Gems II, Charles River Media, 2001.
[11] Mickey Kawick, Real-Time Strategy Game Programming, Wordware Publishing Inc., 1999.
[12] Kevin Hawkins, Dave Astle, Andre Lamothe, Open GL Game Programming, Prima Tech, 2001.
[13] Adrian Perez, Dan Royer, Advanced 3-D Game Programming Using DirectX 7.0, 2000.
[14] David H. Eberly, 3D Game Engine Desgin, Morgan Kaufmann, 2002.
[15] Tomas Moeller, Eric Haines, Real-Time Rendering, AK Peters, 2003.
[16] Alan Watt, Fabio Policarpo, 3D Games, Addison Wesley, 2004.

최 학 현

2005년 : 고려대학교 영상전공 (박사수료)
2007년 : 서울대학교 문화콘텐츠 (GLA)
2007년 : 문화콘텐츠연구센터 (책임교수)
현 재 : 서울여자대학교 미디어학부
관심분야 : 디지털콘텐츠(Digital Contents)

김 정 희

2001년 : 숙명여자대학교 (학사)
2007년 : 성균관대학교 (박사과정)
2007년 : 문화콘텐츠연구센터 (객원연구원)
현 재 : 성균관대학교 대학원
관심분야 : 디지털콘텐츠(Digital Contents)