공학석사 학위논문

네트워크 기반 협업 설계 시스템 구축을 위한 Java 3D 모델링 데이터 뷰어에 관한 연구

A Study on Java 3d Modeling Data Viewer Systems for Collaborate Design System in Network Platform

지도교수 정 재 현

2003년 2월

한국해양대학교 대학원

기계공학과

김 정 욱

본 논문을 김정욱의 공학석사 학위논문으로 인준함

위원장 공학박사 조 종 래 인

위 원 공학박사 최 형 식 인

위 원 공학박사 정 재 현 인

2003년 2월

한국해양대학교 대학원

목 차

Abstract ····· iii

표 목차 ····· v

그림 목차 ····· v

제 1 장 서론 ····· 1

1.1 연구의 필요성 및 배경 ····· 1

1.2 연구내용 ····· 2

제 2 장 시스템 구성 및 연구 내용 ····· 3

2.1 시스템 개요 ····· 3

2.2 연구내용 ····· 5

2.2.1. 시스템의 구축 ····· 5

2.2.1.1 시스템 실행 환경 ····· 5

2.2.1.2 실시간 시각화 모듈 ····· 6

2.2.1.3 Java 3D의 장점 ····· 7

2.3 Java3D에서 지원하는 OBJ 파일을 이용한 로더 ····· 8

2.3.1 OBJ 파일의 분석 ····· 9

2.3.2 로더 패키지 인터페이스와 기본 클래스 ····· 11

2.4 Java3D에서 지원하지 않는 ASE 파일을 이용한 로더 구현 ····· 11

2.4.1 ASE 파일의 내용보기 ····· 12

2.4.2 ASE 파일분석하기 ····· 13

2.4.3 노드정보 읽기 ····· 14

2.4.4 바이너리 변환 ····· 16

2.4.5 메모리 로드 ····· 16
2.4.6 바이너리 파일 디코더 ····· 18
2.4.7 뷰어로의 렌더링 ····· 20
2.4.8 JNI를 이용한 Java 3D로의 호출 ····· 23
2.5 사용자 작동 모듈 ····· 23
2.6 엔지니어링 데이터베이스 구현 ····· 29
2.6.1 엔지니어링 데이터베이스 시스템의 종류와 특징 ····· 30
2.6.2 엔지니어링 데이터베이스개발의 기법과 도구 ····· 31
2.6.3 엔지니어링 데이터베이스 계획의 단계 ····· 31
2.6.4 클래스별 속성 정의 ····· 34
2.6.5 데이터 사전 항목 작성 ····· 35
2.6.6 MySQL를 이용한 시스템 구성 ····· 35
2.7 2계층 방식의 네트워크를 이용한 연동 ····· 36
2.7.1 JDBC ····· 37
2.7.2 RMI ····· 38
2.7.3 네트워크 기반 엔지니어링 데이터베이스 시스템의 구현 ··· 39
제 3 장 결론 ····· 42
참고 문헌 ····· 44

A Study on Java 3D Modeling Data Viewer System for Collaborate Design System in Network Platform

Jeong-Wook Kim

Department of Mechanical Engineering
Graduate School, Korea Maritime University

Abstract

This study is on the development of the modeling viewer for the collaborating design on networking environments using java 3d. Many years ago, most of designer had to plan with the drafting tools like compass, ruler and calculator for product design. But after adding the computer system, the design project by CAD system has been the standard in industrial works. At this time, CAD includes 2D drafting, 3D modeling and visualized simulation for the engineering. And in the future and now, for knowledge based information system are including.

In this study for the developing the modeling viewer in collaborating design, the first module to reading CAD model file, the second to managing the read data in DB, and the third to exchanging of data are programmed using java 3d in client-server networking. With these module, the basic step to constructing collaborating design system is applying.

표 목차

표 1. Java 3D에서 지원하는 모델링 데이터 파일
표 2. OBJ 파일의 포맷
표 3. ASE 파일의 포맷
표 4. 노드 정보를 읽기위한 aseParse() 함수
표 5. 컨버트된 파일을 메모리에 생성하기 위한 과정
표 6. 바이너리 파일 디코더인 binReader() 함수
표 7. 뷰어로 렌더링 하기위하 모듈
표 8. MySQL에서 생성된 속성 테이블

그림 목차

그림 1. 모델링 데이터 로더의 구조
그림 2. 전체 시스템 구조
그림 3. Scene 구성을 위한 모듈
그림 4. Java 3D 클래스 계층 구조도
그림 5. 정육면체의 모델링
그림 6. 3D Studio MAX를 이용한 주전자 모델링
그림 7. 사용자 작동 모듈 구성도
그림 8. 모델링 데이터 뷰어 실행화면
그림 9. 마우스 왼쪽 버튼을 이용한 좌우 회전
그림 10. 마우스왼쪽 버튼을 이용한 상하 회전
그림 11. 마우스 중간 버튼을 이용한 확대/축소
그림 12. 마우스 오른쪽 버튼을 이용한 이동

그림 13. 제작한 프로그램의 장면 그래프
그림 14. 데이터 베이스 계획의 단계
그림 15. 2-계층 구조
그림 16. 3-계층 구조
그림 17. 모델링 데이터의 검색과 속성 보기
그림 18. 검색된 모델링 데이터 파일의 상세 속성 보기
그림 19. 모델링 데이터 파일의 상세 보기2

제 1 장 서론

1.1 연구의 필요성 및 배경

소비자의 다양한 요구, 빠른 시장 환경의 변화와 경쟁업계와의 경쟁심화로 인하여 경쟁력 있는 제품을 빠른 시간 내에 개발하기 위하여 노력하는 것은 중요한 쟁점이다. 이의 한 방법론으로 종래의 순차적 개발 방법에서 동시 공학적 개발 방법으로 변하고 있다.

일반적인 제조업에서의 제품 개발 과정은 CAD를 통한 디자인, 디자인 결과에 대한 시뮬레이션, 목업 형태의 모델 제작, 시제품 개발 등의 과정을 거치게 된다. 이 과정에서 각 단계에서의 개발자 및 의사 결정권자간의 원활한 의사 전달은 실제 개발 과정을 효율적으로 구성하는 데에 있어서 필수적이다. 특히, 전자 제품이나 자동차와 같은 제품은 이전에 비하여 점점 라이프 사이클의 주기가 짧아지고 있으며, 이를 위하여 제품 디자인 과정을 효율적으로 구성하는 것은 필수적이다. 또한, 자동차와 같은 수천 내지 수십만에 달하는 많은 부품들로 구성된 제품에 있어서, 디자인 과정에서 이러한 부분품을 조립하고, 수정하는 작업은 제품을 평가하는 필수적인 과정이 된다.[1],[2]

이러한 요구에 의해 가상현실기술을 활용한 디자인은 CAD, 모델 제작과 같은 전반적인 디자인 단계의 효율성을 극대화시킬 수 있다. 가상현실기술을 이용한 CAD 정보의 실시간 3차원 제시는 개발자뿐만 아니라 평가자 및 비전문가에게 쉽게 이해될 수 있다. 자동차와 같이 복잡한 대규모 모델의 경우, 수 억개 단위의 폴리곤을, 전자제품에 있어서도 수백만 내지 수천만 단위의 폴리곤을 갖는 CAD 모델에 대한 실시간 삼차원 시각화는 효율적 디자인 과정을 구성하는 필수적인 요소가 된다.

또한 가상현실기술은 디자인된 제품 및 그 일부분에 대한 직접 조작을 가능하게 하여, 개발 검토 과정에서 발생한 오류를 바탕으로 수정된 제품을 개발하는 제품 개발 과정을 효율적으로 구성한다.[3],[4] 특히 실제의 시제품이나 목업을 제작하

지 않고서도 가상의 공간에서 조립하고 수정할 수 있어, 그 형상이나 기본적 기능에 대한 검토를 손쉽게 하며, 그에 대한 수정과 그 결과의 확인이 손쉽게 이루어진다. 이와 같이 복잡한 모델링 데이터에 대한 실시간 삼차원 영상생성과 가상공간에서 제공하는 조립과정은 제품개발단계에서 빠른 Prototyping을 가능하게 하는 필수적인 기술이다.

대규모 모델링 데이터를 사용한 실시간 조립 과정은 CAD디자인, 시뮬레이션, 목업 제작, 검토, 수정으로 이루어지는 제품 디자인 과정에 있어서 실시간에 Virtual Prototype을 통하여 제품 개발과정을 단축할 수 있으며 궁극적으로는 기업의 경쟁력을 향상시킬 수 있게 된다. 아울러, 가상공간에서의 협업 디자인이 되면 디자인 단계에서 기술자, 생산자, 소비자, 기획자, 광고 담당자의 의견이 반영될 수 있어서 보다 효율적인 빠른 개발과정을 구성한다.

이러한 추세에도 불구하고 기존에 존재하고 있는 상용화된 어플리케이션의 경우 시스템이 체계화 되고 전문화되어 있다는 장점은 가지지만, 초기 투자 비용이 많이 들며 시스템의 규모가 필요 없이 커 중소기업에서는 섣불리 도입할 수 없는 단점을 가지고 있다. 그래서, 본 논문에서는 프리웨어로 제공되고 있는 Java 3D API, 리눅스 그리고 MySQL 등을 이용하여 중소기업에서도 저가의 비용으로 구현이 가능한 협업 설계시스템을 구현해보고자 한다.

1.2 연구내용

본 연구에서는 Java 3D를 이용한 모델링 데이터 뷰어시스템과 엔지니어링 데이터 베이스 시스템을 구축하는 것을 목표로 한다. 그 주요 내용은 첫째, 3D로 설계된 모델링 데이터 파일을 분석하고 로더를 사용하여 로딩하여 시각화 하는 기술과 둘째, 시각화된 모델링 데이터 파일을 회전, 확대/축소, 이동 등의 사용자 조작에 따라 작동하는 사용자 작동 모듈이 구현되어있으며 셋째, 모델링 데이터 파일을 저장, 삭제, 수정 등을 통해 관리하기 위한 엔지니어링 데이터베이스, 그리고 마지막으로 2-계층 방식의 네트워크 기반에서 RMI와 JDBC를 이용한 서버-클라이언트간의 데이터 교환을 위한 기술이 포함되어있다. 본 연구의 주요 내용을

요약한 것으로 다음과 같다.

1) 3D로 설계된 모델링 데이터 파일을 시각화하기 위한 시각화 모듈 구성
2) 마우스 키보드 등을 통한 사용자 작동 모듈
3) 데이터베이스를 통한 데이터 관리
4) 2-계층 방식의 네트워크와의 연동을 위한 데이터 교환 기술

제 2 장 시스템 구성 및 연구 내용

2.1 시스템 개요

본 연구에서 개발하는 Java 3D 모델링 데이터 뷰어는 가상현실환경을 기반으로 3D 모델링 데이터에 대한 관리와 검색을 가능하게 하는 시스템을 구축하는 것을 목표로 한다. 이를 위하여 3D CAD 모델로 구성되는 가상환경을 구축하고, 그 가상환경에서 시각화하고 사용자 작동을 가능하게 하는 것이 그 주된 목표이다.

그리고, 이러한 시각화 기술에 더하여 모델링 데이터 파일의 정보를 관리하기 위한 엔지니어링 데이터베이스 시스템을 개발한다. 이를 위하여, 전반적인 시스템은 기본적인 가상현실환경을 구성하는 커널 부분과, 사용자 작동을 위한 응용부분 그리고 모델링된 데이터 파일을 관리하는 부분으로 구성한다.

커널 부분은 모델링 데이터 파일을 시각화하기 위한 구현 부분으로써 실시간 시각화와 가상환경 상에서의 사용자 작동을 위한 기본적인 프레임의 역할을 한다. 커널부에서는 화면 표현을 위해 모델링 데이터 파일을 로더를 사용하여 생성한다.

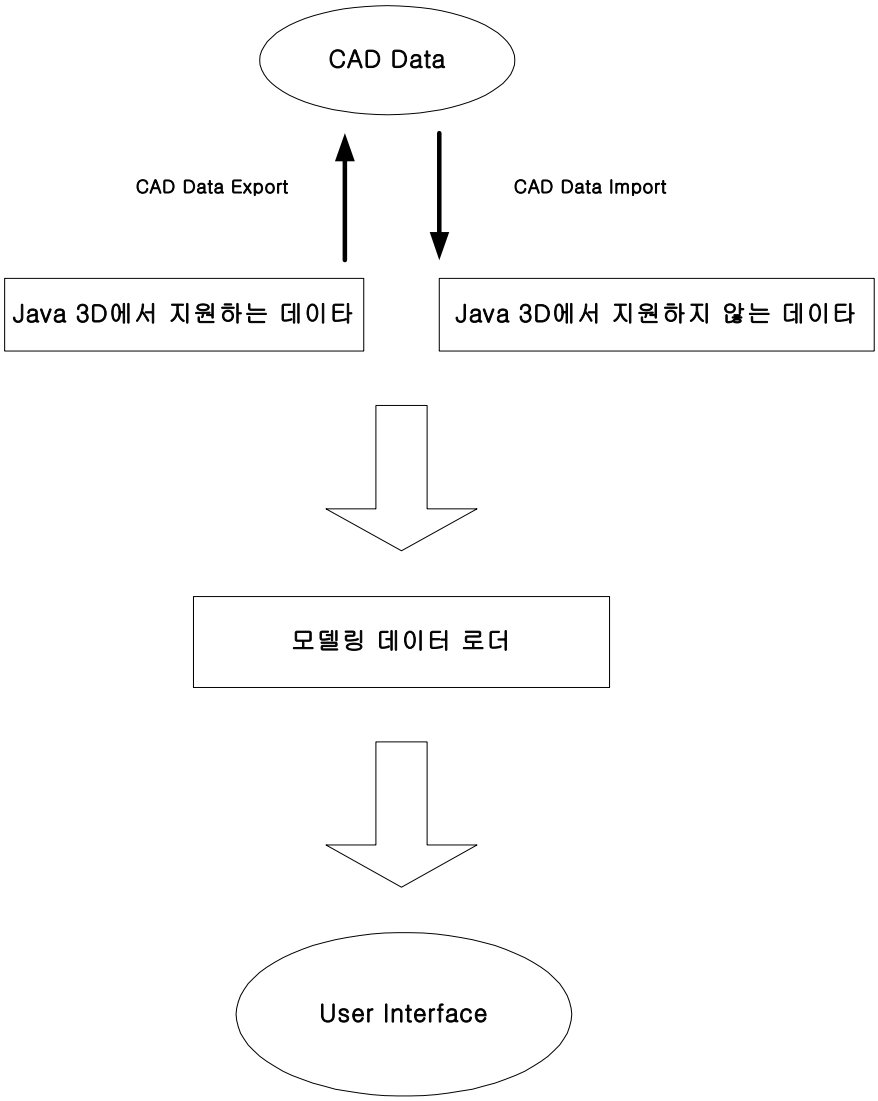

그림 1 모델링 데이터 로더의 구조

그림 1에서와 같이, 본 연구에서는 Java 3D API에서 기본적으로 지원하는 모델링 데이터 파일을 위한 로더 클래스를 이용하는 방법과, 이의 확장을 위하여 Java 3D에서 지원하지 않는 모델링 데이터 파일을 로더하는 구현 방법으로 나뉜다. 전자의 방법은 OBJ파일을 사용하여 구현하였고, 후자는 3D Studio MAX에서 사용하는 ASE파일을 이용하여 구현하였다.

사용자 작동을 위한 부분은 본 연구에서 요구하는 부분이 Java 3D에서 지원하

는 Behavior클래스들을 이용하여도 충분하므로 이를 사용하여 구현하였다.

모델링 데이터 파일의 관리를 위한 엔지니어링 데이터베이스의 구현은 도면의 저장, 삭제, 조회, 수정 등의 간단한 기능으로 이루어졌으며, 네트워크 기반으로 작동하기 위하여 Linux 5.2를 서버 측에 설치했으며 엔지니어링 데이터베이스를 위해 MySQL 3.23.49를 이용하였다.

그림 2에서 보는 바와 같이 2-계층 기반의 네트워크의 구현은 Java RMI와 JDBC를 이용하였고, 또한 네트워크를 위해 TCP/IP로 연결된 클라이언트 측의 윈도우 상에서 이를 테스트할 수 있도록 개발되었다.

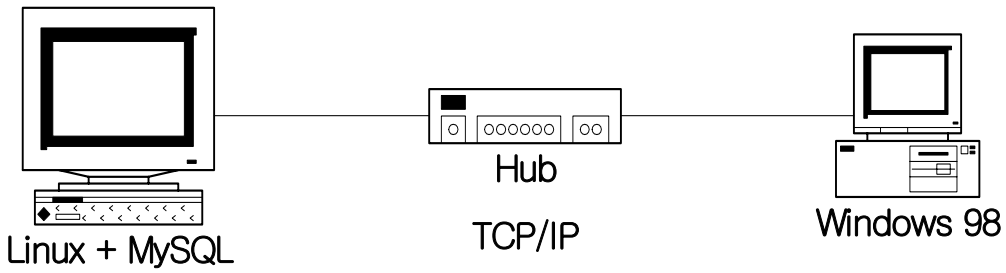

그림 2 전체 시스템 구성도

2.2 연구내용

2.2.1. 시스템의 구축

2.2.1.1 시스템 실행 환경

본 연구에서 구현될 시스템은 서버와 클라이언트로 구성된 2-계층방식의 TCP/IP로 연결된 네트워크 환경에서 실행된다. 먼저 서버로 이용될 개인용 컴퓨터는 Linux 5.2가 설치되어있으며 MySQL 3.23.49를 이용하여 엔지니어링 데이터베이스를 구축하고, 연결된 네트워크를 통해 윈도우 98이 설치된 클라이언트

컴퓨터에서 서버의 접속을 통한 모델링 데이터 파일 뷰어를 구현하고자 한다.

2.2.1.2 실시간 시각화 모듈

실시간 시각화 모듈이란, 3D 모델링 데이터를 로더하는 모듈이다. 이 모듈은 전체 시스템에서 가장 중요한 최적화된 화면 표현을 생성하는 모듈이다. 보통 상용화된 어플리케이션의 경우, 자체 포맷의 모델링 데이터를 가지고 있으며, 이 데이터 파일을 쉽게 로더 할 수 있다. 하지만, 이러한 데이터의 구조는 어플리케이션을 제작한 회사 측에 소유권이 있으며, 그러한 소유권을 획득하기 위해서는 상당한 비용이 필요하게 된다. 그러나, 본 연구에서는 그러한 상용화된 모델링 데이터 포맷 중에 그 형식이 개방된 포맷을 이용하였으며, 그 역할은 그림 3과 같다.

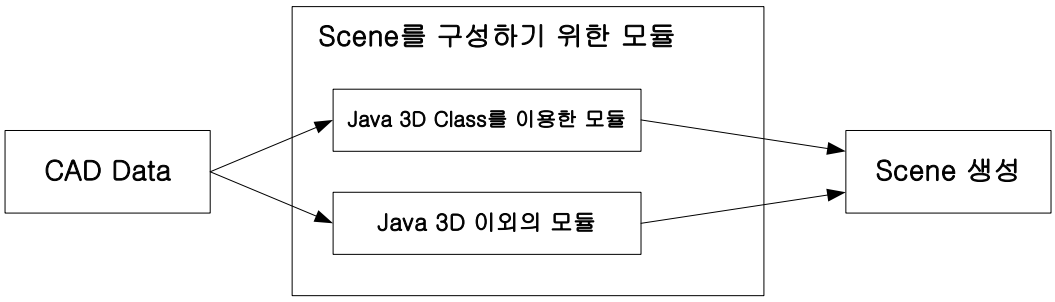

그림 3 Scene 구성을 위한 모듈

본 연구에서는 Scene 구성을 위해 각 모듈별로 다음과 같이 표현하였다.

1) 3차원 형상 정보

본 시스템의 형상은 폴리곤 메쉬로 구성된다. 이러한 폴리곤 메쉬는 메쉬를 구성하는 점과 점들로 구성된 폴리곤들의 집합으로 표현된다. 이렇게 표현된 폴리곤들은 사용자의 사용자 작동 모듈에 의해 동작되어지는 과정에서도 변화하지 않으며, 실시간 렌더링 모듈을 효율적으로 구성하는데 이용된다.

2) 모델링 데이터 로더

실시간 시각화 기능을 갖는 모듈이다. 이 모듈은 복잡한 3D 모델에 대하여, Java 3D에서 사용가능한 데이터 파일을 로더하는 방법과 Java 3D에서 지원하지 않는 데이터 파일을 분석하여 Java 3D와 OpenGL을 이용하여 로더하는 방법으로 나누어진다.

2.2.1.3 Java 3D의 장점

본 연구에서 사용된 Java 3D 1.3 SDK는 Open Inventor와 VRML에 비하여 발전된 특징을 가지는데, 가장 큰 특징은 가상현실을 염두에 둔 설계라고 할 수 있다. 3D 그래픽 API 뿐만이 아니라 삼차원 음향(sound) API가 포함된 음향 라이브러리, 빛을 위한 범위(scoping) 모델, 대용량의 가상 세계를 만들기 위한 고해상도의 좌표계를 지원하며 기하 도형의 기본 원소로 삼각형을 사용하여 상위 레벨의 오브젝트를 추상화할 수 있도록 하였다. 자연스러운 애니메이션과 렌더링을 위하여 이중 버퍼링(Double-buffering), 트루-칼라(true-color), Z-buffer를 지원한다.

가장 우수한 점은 다른 모델링용 저작도구의 경우 자체 파일 포맷을 가지고 있어서 자체 저작도구를 이용해 모델링하거나 다른 저작도구를 사용하여 파일을 변환해 이용한다. 그러나 Java 3D의 경우 수많은 3D 모델링 데이터에 대한 로더를 제공하여, 별도의 변환 없이 그대로 이용할 수 있다. 이러한 점은 설계자가 자신들의 손에 익은 툴을 그대로 사용할 수 있으므로 개발 시간을 단축시킬 수 있는 장점을 가지고 있다.

Java 3D를 이용함으로써 얻게 되는 장점은 크게 세부분으로 요약할 수 있는데 첫째, OpenGL이나 DirectX 등의 다른 Graphic API에 비해 더 최근에 발표되었으므로 이들의 단점을 많이 보완하여 객체지향 프로그래밍이라는 특징을 가지고 있으며, 웹 환경에서 수행이 가능하며, OpenGL이나 DirectX의 Low-API를 이용해 코드를 생성해내기 때문에 속도도 빠르며, 높은 수행능력을 가지고 있다. 둘째, Java 언어 자체의 특징인 플랫폼 독립성을 이어받아 윈도우, 리눅스 그리고 유닉스 등의 환경에 구속받지 않고 동작할 수 있으며, 셋째, 본 연구의 내용 중에 가

장 중요한 부분인 모델링 데이터의 로더를 위해 다양한 로더를 제공한다는 점이다.

그러나, 현재까지 나와 있는 Java 3D API는 그래픽 프로그램에서 자주 이용되는 NURBS을 이용한 곡면생성, 광 추적(ray tracing)을 지원하지 않으나 향후 이러한 기능들이 Java 3D에 포함될 것으로 예상된다.

그림 4는 Java 3D에서 지원하는 클래스 계층 구조도를 나타내고 있다. 본 연구에서는 주로 come.sun.j3d.utils 클래스를 사용하여 구현되었으며, 이 클래스는 모델링 데이터 파일 로더, Scene Graph 생성, Geometry 등의 기능을 지원하고 있다.[5],[6]

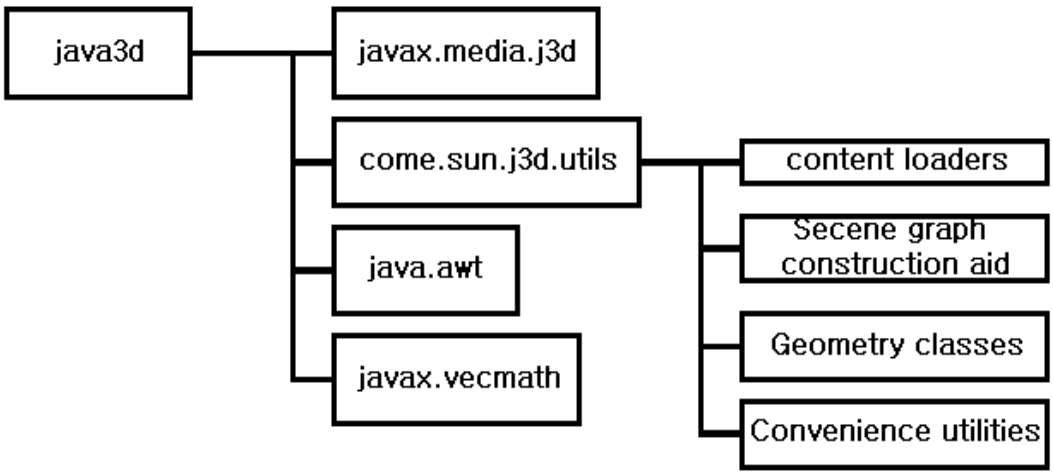

그림 4 Java 3D 클래스 계층 구조도

2.3 Java 3D에서 지원하는 OBJ 파일을 이용한 로더

Java 3D에서는 기본적으로 지원하는 로더 클래스를 이용하여 CAD 또는 3D 모델링 어플리케이션에서 생성한 삼차원 장면 파일을 읽어 들이고 표현할 수 있다.

이때 기존의 삼차원 장면의 일부만을 선택적으로 포함할 수 있으며, 다른 Java 3D 코드로 확장도 가능하다. com.sun.j3d.loader 유틸리티 패키지는 다른 응용프로그램으로부터 생성된 파일로부터 그 내용을 읽어 Java 3D 응용프로그램에

적재(load)하는 기능을 제공한다. 그림 4는 Java 3D에서 제공하고 있는 패키지들과 클래스들의 계층 구조도이다.

삼차원 장면을 표현하는 데는 다양한 파일 형식이 존재하기 때문에 현재 Java 3D가 제공하는 파일 형식이외에도 여러 가지 파일이 있을 수 있다. 이러한 이유로 Java 3D는 파일을 로더하기 위한 실제 코드를 가지는 것이 아니라 파일을 로딩할 수 있는 인터페이스만을 제공한다. 이 인터페이스 정의를 이용하여 Java 3D의 개발자는 다른 로더 클래스와 동일한 인터페이스를 가진 새로운 로더 클래스를 개발할 수 있다. 표 1은 Java 3D에서 지원하는 모델링 데이터 파일의 종류를 도표로 표시한 것이다.[5],[6]

파일형식	설명
3DS	3D Studio
COB	Caligari TrueSpace
DEM	Digital Elevation Map
DXF	AutoCAD Drawing Interchange Format
IOB	Imagine
LWS	Lightwave Scene Format
NFF	WorldToolKit NFF Format
OBJ	WaveFront
PDB	Protein Data Bank
SLD	Solid Works(prt and asm files)
VRT	SuperScape VRT
VTL	Visual ToolKit
WRL	Virtual Reality Modeling Language

표 1 Java 3D에서 지원하는 모델링 데이터 파일

2.3.1 OBJ 파일의 분석

Java 3D에서는 많은 모델링 데이터를 지원하지만, 본 연구에서는 가장 일반적으로 많이 사용되는 OBJ 포맷을 이용하여 구현하였다. OBJ 포맷의 경우 좌표와 속성이 직접 텍스트 형식으로 구성되어있다.

다음의 표 2는 그림 5의 정육면체를 모델링한 후에, 생성된 OBJ파일을 에디터를 이용하여 호출한 것이다. v는 정점을 나타내며, 좌표순서는 x, y, z 좌표값을

나타낸다. 정육면체의 꼭지점이 8개이며, 6개의 면이 있는데 이들을 나타내고 있다. f 1 3 4 2라고 하면, 1, 3, 4, 2 번째의 정점을 조합한 면을 나타내는 것이다. 이와 같이 OBJ파일은 텍스트 형식으로 돼있어 분석이 쉽고 에디터를 통한 수정도 가능하다. 텍스트 형식이기 때문에 오브젝트가 복잡하면 파일의 크기가 커지는 단점이 있다. 이를 보완하기 위해서 Java 3D에서는 기하도형압축을 지원하고 있다. 기하도형을 압축하기 위한 ObjectFileCompressor라는 클래스를 지원한다.

그러나, 변환된 파일은 바이너리 파일이므로 에디터로 열어봐도 그 의미를 알 수 없다. 이러한 압축의 방법은 파일의 크기를 줄일 수 있다는 장점이 있으나, 텍스처 좌표는 무시되어 렌더링된 장면을 얻을 수 없으므로 주의 하여야 한다.

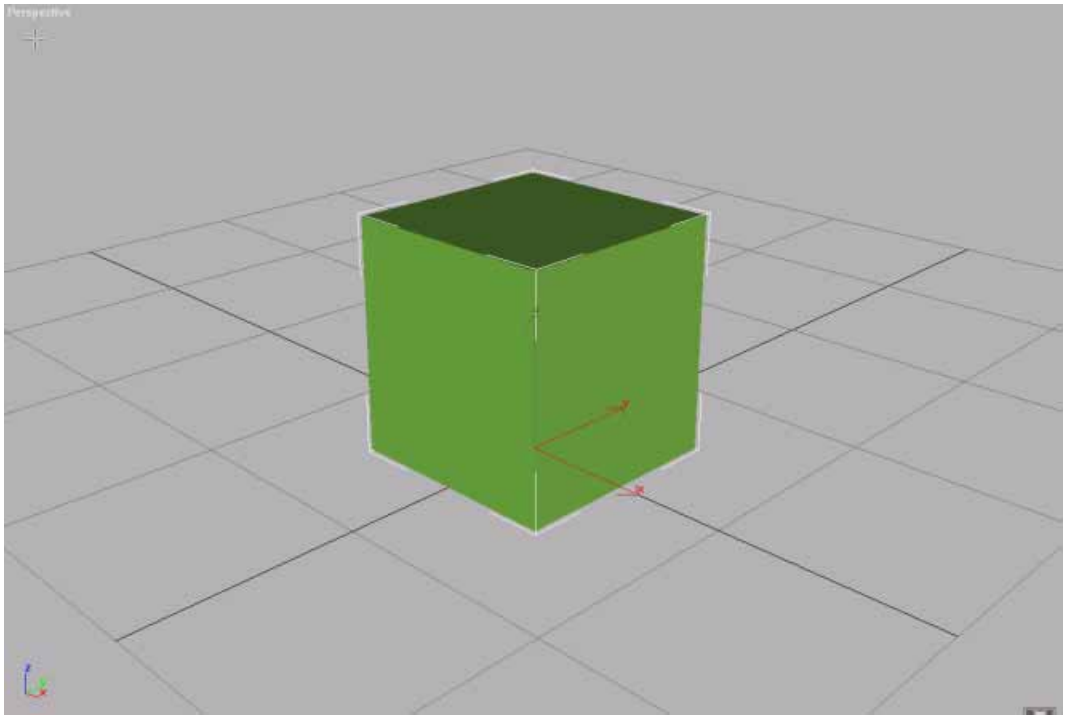

그림 5 정육면체의 모델링

```
v 1 1 1          #vertex 1
v 1 1 -1         #vertex 2
v 1 -1 1         #vertex 3
v 1 -1 -1        #vertex 4
v -1 1 1         #vertex 5
```

```
v -1 1 -1        #vertex 6
v -1 -1 1        #vertex 7
v -1 -1 -1       #vertex 8
f 1 3 4 2
f 5 7 8 6
f 1 5 6 2
f 3 7 8 4
f 1 5 7 3
f 2 6 8 4
```

표 2. OBJ 파일의 포맷

2.3.2 로더 패키지 인터페이스와 기본 클래스

모델링 데이터를 로더하기 위해 Java 3D에서 지원하는 로더가 다양한 이유는 설계자가 로더를 쉽게 쓰도록 하기 위해서이다. 로더 클래스는 로더를 작성하는 어려움을 덜기 위한 로더 인터페이스의 구현으로, 인터페이스는 다양한 로더 클래스가 그 인터페이스에 있어 일관성을 가지도록 하는데 그 목적이 있다.

예를 들어보면 삼차원 파일을 로딩하는 프로그램은 로더와 장면 그래프를 모두 사용한다. 로더는 파일의 내용을 읽고, 파싱한 후 화면을 생성한다.

장면 오브젝트는 로더에 의하여 생성된 장면 그래프를 저장한다. 서로 다른 형식의 파일 역시 적절한 로더 클래스를 사용하여 하나의 Java 3D 프로그램으로 결합 될 수 있다.

2.4 Java3D에서 지원하지 않는 ASE 파일을 이용한 로더 구현

Java 3D에서 지원하지 않는 포맷의 경우는 C/C++과 OpenGL을 이용하여 로더를 구현하고, JNI를 이용하여 구현된 메서드를 로더하는 방법을 사용하였다. 여기서, 사용된 ASE 포맷은 3D Studio MAX에서 지원하는 데이터파일이다. 물론 ASE 외에도 다양한 포맷이 존재하지만, 여러모로 쉽고 사용이 편리하여 장점이 많다.

2.4.1 ASE 파일의 내용보기

아래의 표3은 그림 6에서 보는 것과 같이 3D MAX에서 생성한 주전자 모델링 파일을 ASE파일로 변환 한 것이다.

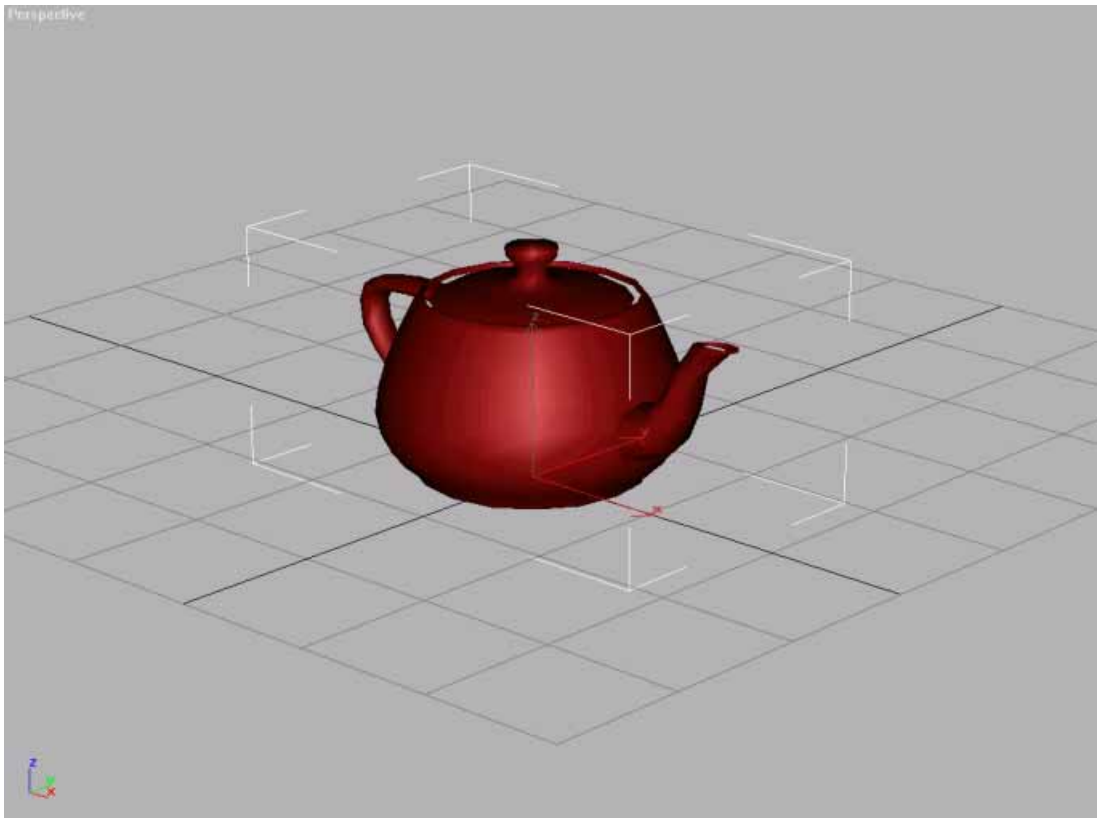

그림 6 3D Studio MAX를 이용한 주전자 모델링

```
*3DSMAX_ASCIIEXPORT 200
*COMMENT "AsciiExport Version 2.00 - The Aug 12 :16:45 2002"
*SCENE{
        *SCENE_FILENAME "AseTest.max"
        *SCENE_FIRSTFRAME 0
          .....
        *SCENE_BACKGROUND_STATIC 0.0000 0.0000 0.0000
        *SCENE_AMBIENT_STATIC 0.0431 0.0431 0.0431
}
```

```
*MATERIAL_LIST{
        *MATERIAL_COUNT 0              ------------------> ①
}
*GEOMOBJECT{
*NODE_NAME "Teapot01"
*NODE_TM{                              ------------------> ②
*NODE_NAME "Teapot01"
  ........
*TM_SCALE 1.0000 1.0000 1.0000
*TM_SCALEAXIS 0.0000 0.0000 0.0000
*TM_SCALEAXISANG 0.0000
}
*MESH{
*TIMEVALUE 0
*MESH_NUMVERTEX 530                    ------------------> ③
*MESH_NUMFACES 1024                    ------------------> ④
*MESH_VERTEX_LIST{
*MESH_VERTEX 0 3.1776 -5.5249 44.5949 -----------------> ⑤
*MESH_VERTEX 4 5.0357 -5.5249 44.5949
*MESH_FACELIST{
*MESH_FACE 1: A: B: 6B: 1C: 0AB: 1BC: 1CA: 0
*MESH_SMOOTHING 1 *MESH_MTLID 0
*MESH_FACE 2: A: 1 B: 6 C: 7 AB: 1 BC: 1 CA: 0  --------> ⑥
*MESH_NUMVERTEX 0                      -------------------> ⑦
}
*PROP_MOTIONBLUR 0
*PROP_CASTSHADOW 1
*PROP_PRECVSHADOW 1
*WIREFRAME_COLOR 0.3451 0.7804 0.8824          ---------> ⑧
```

표 3 ASE 파일의 포맷

2.4.2 ASE 파일분석하기

표 3에 보는 것과 같이 모든 리스트는 '*'라는 표시로 시작한다.

①에서 MATERIAL_LIST는 MAX에서 물체에 입히는 재질정보가 나열되는 노드다. 재질정보가 없기 때문에 0이다.

②에서 NODE_NAME은 그 물체의 이름을 담고 있는 노드이다. 물체의 이름은 MAX에서 지정하며, 차후에 메모리에 있는 모델을 이름으로 찾기 위해 필요한 정보이므로 저장할 필요가 있다. 그 밑에 NODE_TM이라는 노드가 보이는데, 물체의 회전과 이동상태를 저장하고 있는 매트릭스 정보다. 그러나, 본 연구에서는 MAX의 정적인 에니메이션을 지원하지 않고 차후에 이벤트로 지정하는 동적인 움

직임을 지원할 것이므로 이 노드는 읽지 않기로 한다.

③과 ④에는 총 정점들의 개수와 그 정점을 이용한 면들의 개수가 저장되어있다. 이 주전자 모델은 1024개의 면과 530개의 정점들로 구성돼 있다는 것을 알 수 있다.

⑤은 정점의 목록이다. 총 530개라서 중간에 생략했다. 정점은 "MESH_VERTEX" 라는 노드로 대변되며 이 노드들은 ⑤ 바로 위의 *MESH_VERTEX_LIST 노드의 자식 노드들이다. "MESH_VERTEX" 노드에는 4개의 인자가 존재하는데, 첫 번째 인자는 정점 목록에서의 순서 번호이다. 그 뒤를 따르는 3개의 소수점이 바로 정점의 X, Y, Z 좌표이다.

⑥노드는 *MESH_FACE_LIST 노드의 자식 노드들로서 물체를 이루는 각 면이 어떤 정점들로 연결돼 있는지를 나타내고 있다. MAX에서 모든 면은 최종적으로 삼각형의 세 꼭지점으로 분리돼 출력된다. MESH_FACE에서 그 삼각형의 세 꼭지점에 해당하는 정보가 A; n1 B; n2 C; n3의 각 n에 해당한다. MESH_FACE 바로 다음에 나오는 숫자는 그 면의 순서정보이며, 항상 1씩 늘어나는 순서이기 때문에 별 필요가 없는 정보다. ⑥의 맨 뒤에 있는 *MESH_MTLID는 서브매터리얼을 지정하는 번호이므로 읽어야할 정보다 *MESH_MTLID는 각 면마다 다른 매터리얼을 적용할 수 있다.

⑦은 정점이나, 면 정보의 목록처럼 텍스처 매핑 정보를 담는 부분이지만 본 모델에는 텍스처 매핑을 하지 않아 *MESH_NUMVERTEX의 수가 0으로 설정돼있다.

마지막으로 ⑧의 *WRITEFRAME_COLOR 노드는 MAX 화면에서 보이는 모델의 기본 색상정보를 의미하며, 매터리얼이 없는 모델에 생성되는 정보이다. 매터리얼이 있으면 이 노드는 존재하지 않는다.

2.4.3 노드정보 읽기

ASE의 파일을 열어 메터리얼의 개수, 정점 개수, 면 개수, 정점좌표, 면 목록, 와이어 프레임 컬러를 화면에 출력한다. 파라미터로 파일의 이름을 넘겨받게 돼 있다.

파싱을 하는 주요 처리 방법은 aseParse()함수에서 fscanf()함수를 이용해 하나의 문자열을 읽고 그 문자열과 같은 노드의 이름이 있는지 비교해, 같으면 그 노드에 맞춰 나머지 정보를 읽는 것이다. ASE 파일의 경우 문법구조가 간단해서 이와 같은 방법으로도 가능하지만, 제대로 된 분석기를 위해서는 fscanf()함수를 이용하지 않고 각 노드를 미리 분리한 뒤에 파싱하는 것이 기본이다.

```
void aseParse(FILE *s)
{
    char data[256];

    fseek(s,0,SEEK_END);                    ----------> ①
    LONG fs=ftell(s);
    printf("read file(%d bytes)...₩n₩n", fs);
    while(!feof(s)){
    fscanf(s,"%s",data);                    ----------> ②
        if(!strcmp(data, MAT_MATERIAL)){    ----------> ③
        mat_count++;
        }
        else if(!strcmp(data, GEOMOBJECT)){
        object_count++;
        }
        .....
        else if(!strcmp(data,VERTEX)){
        short v;
        float f[3];

        fscanf(s,"%d",&v);
        fscanf(s,"%f %f %f",&f[0],&f[1],&f[2]);
        printf("* vtx %d: %f, %f ,%f₩n",v,f[0],f[1],f[2]);
        }
        else if(!strcmp(data. FACE)){
        short v;
        short vv[3];
        face_count++;

        fscanf(s,"%d",&v);
        fscanf(s,"A:₩t %d B:₩t%d C:₩t%d",&vv[0],&vv[1],&vv[2]);
        printf("* face %d: %d, %d, %d₩n",v,vv[0],vv[1],vv[2]);
        }
    }
}
```

표 4 노드 정보를 읽기위한 aseParse()함수

①은 파일의 크기를 알아내는 부분으로 큰 의미는 없다. ②은 토큰이라고 부를 수 있는 하나하나의 문자열을 뽑아내는 부분이다. 문자열을 하나씩 뽑아내 그 밑

에 나열 돼 있는 비교문에서 해당 노드와 같은지 비교하는 것이다. ③이 첫 번째 비교문으로 매터리얼 노드가 나올 때 마다 매터리얼 카운터 변수를 1씩 증가시킨다.

2.4.4 바이너리 변환

모델링된 파일의 경우 필요 없는 노드들을 제외시켜 파일의 크기를 줄여주어야 한다. 이를 위하여 바이너리로의 변환 과정이 필요하다. 바이너리로의 변환은 위의 경우와 개념적으로 다른 것은 없고 각 노드들의 이름이 숫자로 대변된다는 점만 다르다. 사람이 바이너리 정보를 직관적으로 이해할 수 없다는 점과 플랫폼마다 다를 수 있는 변수 타입의 메모리 크기, CPU마다 다를 수 있는 Big-Endian, Little_Endian 이라는 메모리 저장 방식 때문에 텍스트 정보에 비해 일반적으로 다루기가 어렵다. 그래서, 바이너리를 따로 제어하는 라이브러리를 만들어 사용하였다.

2.4.5 메모리 로드

컨버트된 파일을 생성한 후에 화면에 시각화 하기위한 절차가 필요하다. 표 5는 컨버트된 파일을 해석하여 메모리에 원래의 자료구조로 생성시키는 과정이다.

```
typedef struct
{
        BFMINT8    object_count;
        BFMINT8    mat_count;
        BFMINT16 face_count;
} FILE_HEADER;                                  --------------> ①
```

```
typedef struct
{
        float v[3];
        float n[3];             // vertex normal
```

```
} STRUCT_VERTEX;                                  ----------------> ②
```

```
typedef struct
{
        int a;
        int b;
        int c;
        float n[3];             // face normal
        int mtlid;
} STRUCT_FACE;                                    ------------------> ③
```

```
typedef struct
{
        float v[2];
} STRUCT_TVERTEX;                                 -------------------> ④
```

```
typedef struct
{
        int a;
        int b;
        int c;
} STRUCT_TFACE;                                   --------------------> ⑤
```

```
typedef struct _STRUCT_MATERIAL
{
        float ambient[4];
        float diffuse[4];
        float specular[4];

        struct _STRUCT_MATERIAL* submtl;
        int nSubmtl;
        BOOL bTexture;

        int tid;
} STRUCT_MATERIAL;                                ------------------------> ⑥
```

```
typedef struct
{
        char name[32];

        int nVtx;
        int nFace;
        int nTVtx;
        int nTFace;

        float wfc[3]; // wire frame color

        STRUCT_VERTEX* vtx;
        STRUCT_FACE* face;
        STRUCT_TVERTEX* tvtx;
        STRUCT_TFACE* tface;
```

```
        int mtlref;
        w3DVector3min, max;
} STRUCT_OBJECT;                    ----------------------> ⑦
```

표 5 컨버트된 파일을 메모리에 생성하기위한 과정

①은 ASE 안의 물체의 개수, 메터리얼의 개수 ,총 면의 개수를 지정한다.

②은 물체의 정점에 대한 자료구조이며, n은 정점마다의 수직벡터를 저장한다. 정점마다 수직벡터가 필요한 이유는 OpenGL의 Smooth-Shading을 쓰기 위함이다.

③은 물체의 면에 대한 자료구조이다. 면마다 다른 매터리얼을 가질 수 있기 때문에 mtlid를 추가로 갖고 있으며, n에는 면마다 필요한 수직 벡터를 저장한다.

④은 텍스처 정점을 무시한다. 텍스처는 항상 2차원이라고 간주하므로, Z값은 무시하고 배열의 크기도 2(x,y)로 돼있다.

⑤은 텍스처 면의 정점 인덱스 순서를 저장한다. 반 시계방향으로 간주한다.

⑥은 매터리얼 노드를 설명하는 자료구조이며, 그 가운데 마지막 tid 메터리얼 노드에서 텍스처 파일을 지정할 경우 그 매터리얼에 해당하는 텍스처 ID를 말한다. 텍스처 ID는 ASE파일에서 텍스처 파일이 나타나는 순서와 같다.

⑦은 물체하나를 대변하는 자료구조이다. 만일 여러 그룹과 물체로 ASE파일이 구성됐다면, 이 자료구조로 이뤄진 배열을 준비하면 될 것이다. name은 물체의 이름을 나타낸다.

2.4.6 바이너리 파일 디코더

디코딩 함수인 binReader()는 바이너리 파일을 읽어들이는 중추적인 기능을 수행한다. 표 6에서 디코딩 함수를 분석하면 ①은 정점을 읽어들이는 부분으로 y와 z 값을 교환하고 z에 -를 붙여야 화면에 정확히 출력된다. ②는 물체의 면을 읽어들여 그 면에 해당하는 수직벡터를 계산하는 과정이다. ③은 텍스처 좌표계를 읽어들이는 부분인데, 텍스처를 바로 출력하면 텍스처가 뒤집혀서 입혀지기 때문에

y에 -값을 붙여 텍스처가 원래방향으로 읽혀지도록 한다. ④는 저장된 텍스처 파일을 뽑아내는 과정 가운데 첫 번째로 저장된 파일의 크기를 얻어낸다. ⑤는 서브 메터리얼이 존재하는 지 판단해 존재하면 이 정보를 서브 메터리얼에 저장한다. ⑥이 텍스처 파일을 OpenGL메모리에 적재하는 부분으로 IJL을 사용한다.

```
void w3DModelRenderer::binReader()
{
        while (!bfm_bin.Err())
        {
                bfm_bin.ReadSig(&sig);

                switch(sig)
                {
                case SIG_VERTEX:
                        {
                                float f[3];
                                bfm_bin.ReadFloat32((PBFMFLOAT32)f, 3);
                                obj[cur_obj].vtx[vtx_counter].v[0] = f[0];
                                obj[cur_obj].vtx[vtx_counter].v[1] = f[2];
                                obj[cur_obj].vtx[vtx_counter].v[2] = -f[1];
                                vtx_counter++;                          ----------> ①
                        }
                        break;
                case SIG_FACE:
                        {
                                // face normal calculation.
                                vtx[0][0] = obj[cur_obj].vtx[v[0]].v[0];
                                vtx[0][1] = obj[cur_obj].vtx[v[0]].v[1];
                                vtx[0][2] = obj[cur_obj].vtx[v[0]].v[2];

                                vtx[1][0] = obj[cur_obj].vtx[v[1]].v[0];
                                vtx[1][1] = obj[cur_obj].vtx[v[1]].v[1];
                                vtx[1][2] = obj[cur_obj].vtx[v[1]].v[2];

                                vtx[2][0] = obj[cur_obj].vtx[v[2]].v[0];
                                vtx[2][1] = obj[cur_obj].vtx[v[2]].v[1];
                                vtx[2][2] = obj[cur_obj].vtx[v[2]].v[2];

                                calcNormalf(vtx, obj[cur_obj].face[face_counter].n);
                                ReduceToUnitf(obj[cur_obj].face[face_counter].n);
                                                ----------> ②

                                face_counter++;
                        }
                        break;
```

```
                    case SIG_TVERTEX:
                            {

                                    obj[cur_obj].tvtx[tvtx_counter].v[0] = f[0];
                                    obj[cur_obj].tvtx[tvtx_counter].v[1] = -f[1]; ------------------> ③
                                    tvtx_counter++;
                            }
                            break;
```

```
case SIG_TEXTURE_NAME:
                            {
                                    bfm_bin.ReadInt32((BFMINT32*)&fs, 1);  ----------------------> ④

                                    if (fs > 0)
                                    {
                                            buf = (char*)malloc(fs);
                                            bfm_bin.ReadInt8((BFMINT8*)buf, fs);

                                            if (-1 != cur_submtl)   -------------------------> ⑤
                                            {

                                                    mtl[cur_mtl].submtl[cur_submtl].bTexture = TRUE;
                                                    mtl[cur_mtl].submtl[cur_submtl].tid = tid;
                                            }
                                            else
                                            {
                                                    // the material has a texture
                                                    mtl[cur_mtl].bTexture = TRUE;
                                                    mtl[cur_mtl].tid = tid;
                                            }
                                                    tex[tid].LoadJPEG(buf,fs,GL_LINEAR,
                                        GL_REPEAT);  ----------------------------------> ⑥
                                    }
                            }
                            break;
```

표 6 바이너리 파일 디코더인 binReader()함수

2.4.7 뷰어로의 렌더링

이 부분은 바이너리로 출력된 파일을 디코더에 메모리 자료구조로 저장한 것을 출력하게 된다.

①은 참조하는 메터리얼이 없을 경우 WIREFRAMECOLOR로 색상을 설정하는 부

분이다.

②에는 메터리얼이 서브 메터리얼을 갖고 있는지 판단하며, 다음의 ③에서 서브 메터리얼 번호가 실제 서브 메터리얼의 범위를 벗어나지 못하도록 조정한다.

④와 ⑤는 텍스처 ID에 해당하는 텍스처를 활성화 시켜 면을 렌더링 할 때 적용한다.

⑥과 ⑧은 면과 텍스처 면의 정점 인덱스를 읽어 들이는 부분이며 이 순서대로 정점을 렌더링하면 된다.

⑦에서는 텍스처 정보가 없을 때 텍스처 기능이 비활성화 되도록 한다.

```
for (k=0; k<fh.object_count; k++)
{
        mtlref = obj[k].mtlref;
        for (i=0; i<obj[k].nFace; i++)
        {
                if (-1 == mtlref)    ----------------------------------------------------------> ①
                {
                        glColor3f(obj[k].wfc[0], obj[k].wfc[1], obj[k].wfc[2]);
                }
                else
                {
                        if (mtl[mtlref].nSubmtl > 0)    ---------------------------------------> ②
                        {
// this material has sub-material
                                iSubmtl = obj[k].face[i].mtlid;
                                if (iSubmtl > mtl[mtlref].nSubmtl)
                                iSubmtl = mtl[mtlref].nSubmtl - 1; ----------------------> ③

                                if (mtl[mtlref].submtl[iSubmtl].bTexture)
                                {
                                        bTexture = TRUE;
                                        glEnable(GL_TEXTURE_2D);
                                        tex[mtl[mtlref].submtl[iSubmtl].tid].Apply();
                }    ------------------------------------------------------------------> ④
                                else
                                        bTexture = FALSE;

                                glMaterialfv(GL_FRONT, GL_AMBIENT,
                                mtl[mtlref].submtl[iSubmtl].ambient);

                                glMaterialfv(GL_FRONT,                    GL_DIFFUSE,
                                mtl[mtlref].submtl[iSubmtl].diffuse);
```

```
                                        glMaterialfv(GL_FRONT,GL_SPECULAR,
                                        mtl[mtlref].submtl[iSubmtl].specular);
                                }
                                else
                                {       // this material has no sub-material
                                        if (mtl[mtlref].bTexture)
                                        {
                                                bTexture = TRUE;
                                                glEnable(GL_TEXTURE_2D);
                                                tex[mtl[mtlref].tid].Apply(); ---------->⑤
                                        }
                                        else
                                                bTexture = FALSE;

                                        glMaterialfv(GL_FRONT, GL_AMBIENT, mtl[mtlref].ambient);

                                        glMaterialfv(GL_FRONT, GL_DIFFUSE, mtl[mtlref].diffuse);

                                        glMaterialfv(GL_FRONT, GL_SPECULAR, mtl[mtlref].specular);
                                }
                        }

                        va = obj[k].face[i].a;
                        vb = obj[k].face[i].b;
                        vc = obj[k].face[i].c;      ----------------------------------------> ⑥

                        if (!bTexture)
                                glDisable(GL_TEXTURE_2D);   ----------------------------> ⑦

                                glBegin(GL_TRIANGLES);

                        if (bTexture)
                        {
                                ta = obj[k].tface[i].a;
                                tb = obj[k].tface[i].b;
                                tc = obj[k].tface[i].c;       ----------------------------> ⑧
                                }
                        }

                        glEnd();

                }
        }
}
```

표 7 뷰어로 렌더링 하기위한 모듈

2.4.8 JNI를 이용한 Java 3D로의 호출

Java 에서는 C나 C++ 혹은 어셈블리 언어와 같이, Java 이외의 언어로 작성된 프로그램을 자바가상머신(Java Virtual Machine) 위에서 실행할 수 있도록 인터페이스를 제공해 준다. 즉, Java가 지원하지 못하는 플랫폼 종속적인 기능들을 사용할 수 있으며, 이미 다른 언어로 만들어진 어플리케이션이나 라이브러리를 사용할 수 있다. 이것을 Java Native Interface라고 하며, 이 방법을 사용하여 C/C++과 OpenGL에서 작성된 함수를 Java 3D로 불러 들여 실행 할 수 있다. 이러한 방법은 C/C++ 같이 플랫폼 종속적인 언어와 상호 작용하도록 작성하면, 이 어플리케이션은 플랫폼 독립성을 잃어버릴 것이며, 이식성, 코드의 재사용 등의 강력한 Java 언어의 특징을 이용할 수 없게 되는 단점을 가지며, 각 데이터 타입의 차이에서 오는 변환에도 신경을 써야하며, 예외처리 등의 오류가 발생할 수도 있다.

2.5 사용자 작동 모듈

사용자 작동 모듈에서 제공하는 주요 기능에는 시점의 제어, 부품의 선택과 이동과 같은 조작이 있다. 본 절에서는 이 기능에 대하여 기술한다.

물체의 움직임은 일반적으로 회전(Rotation), 이동(Panning) 그리고 확대/축소(Zoom)로 나뉜다. 패닝과 줌은 물체의 위치를 조절함으로써, 회전은 물체의 피치(Pitch)와 요(Yaw)를 조절함으로써 제어된다. 그림 7은 사용자 작동 모듈의 구성을 그림으로 도시한 것이다.

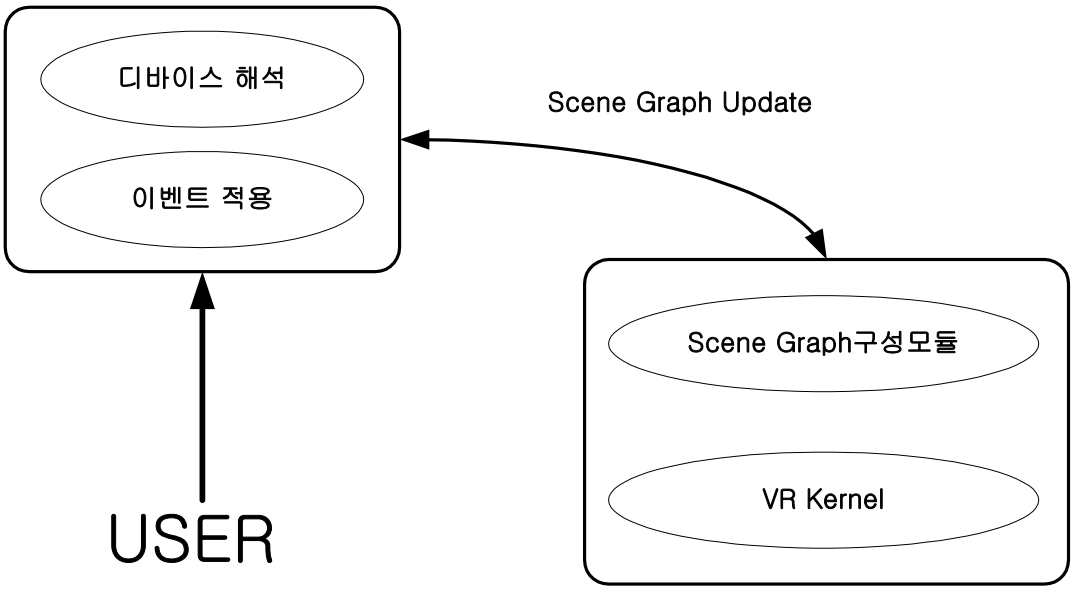

그림 7 사용자 작동 모듈 구성도

시점 제어를 위하여 다양한 방식을 제공하고 있는데, 시점제어는 삼차원 시각화를 위한 시점, 즉 가상 카메라의 위치를 조절하는 것으로 시점이 위치할 삼차원 공간을 정하기 위한 방법이다. 본 연구에 의해 제작된 뷰어는 기본적으로 Java 3D에서 제공하는 Behavior클래스를 이용하여 제작되었다.[5]

- Java 3D의 오브젝트의 변환을 위한 절차

1. 변환을 수행하기 위한 TransformGroup클래스를 생성한다.
2. 변환의 대상이 되는 기하도형을 생성한다.
3. 대상이 되는 기하도형을 변형시키기 위하여 Transform3D 클래스를 생성한다.
4. 생성한 기하도형은 TransformGroup클래스에 추가한다.

그림 8 모델링 데이터 뷰어 실행화면

그림 8은 Java 3D를 이용한 데이터 모델링 뷰어의 실행화면이다. 보고 있는 그림은 포르쉐자동차의 모델링 파일이며, OBJ파일과 ASE 파일의 두가지 형태를 가질 수 있다. 마우스의 조작에 의한 네비게이션은 다음과 같이 작동한다.

마우스 왼쪽 버튼을 클릭한 상태에서 커서의 위치에 따라 상하좌우의 회전을 하며 마우스 중간 버튼은 클릭한 상태에서 모델링 객체의 확대 축소의 기능을 가지며, 마우스 오른쪽 버튼을 클릭한 상태에서 화면을 좌우로 움직여 모델링된 객체를 이동 시킬 수 있다.

다음 그림 9에서 마우스 왼쪽 버튼을 이용하여 그림 8에서 오른쪽으로 회전 시킨 것임을 알 수 있다 그림 10은 역시 마우스 왼쪽 버튼을 이용하여 그림 9의 상태에서 아래쪽으로 회전 시킨 것이다. 그림 11의 경우 마우스 중간 버튼을 이용하여 그림 8에서 보이는 차체의 모습을 확대하여 본 모습이다. 그림 12는 그림 11의 상태에서 마우스 오른쪽 버튼을 이용하여 화면을 이동시킨 것이다.

그림 9 마우스 왼쪽 버튼을 이용한 좌우 회전

그림 10 마우스 왼쪽 버튼을 이용한 상하 회전

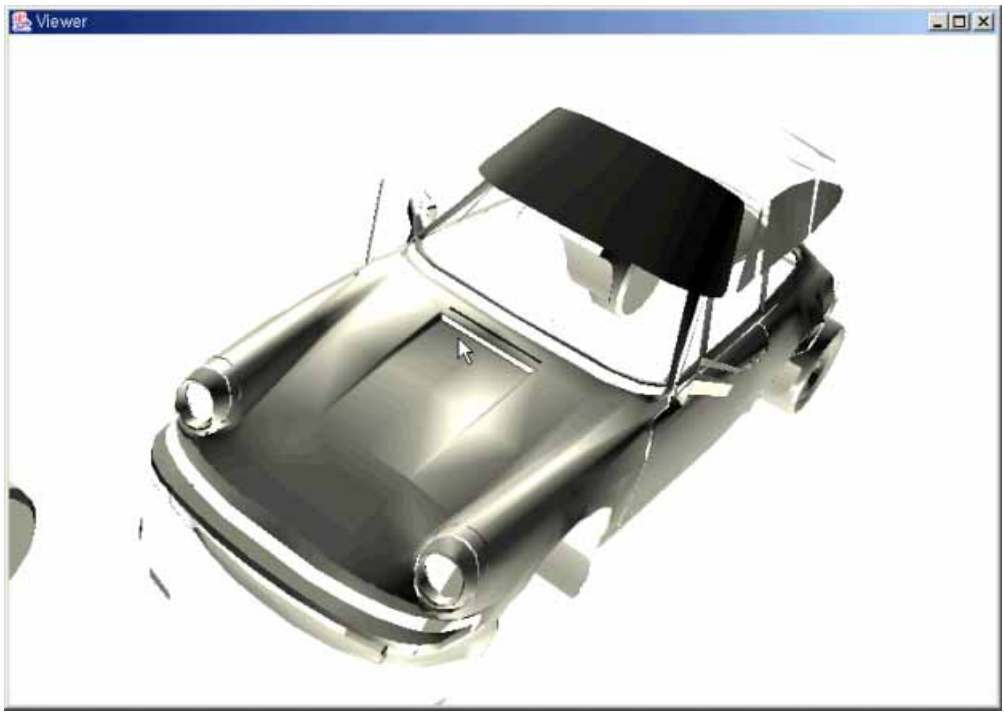

그림 11 마우스 중간 버튼을 이용한 확대/축소

그림 12 마우스 오른쪽 버튼을 이용한 이동

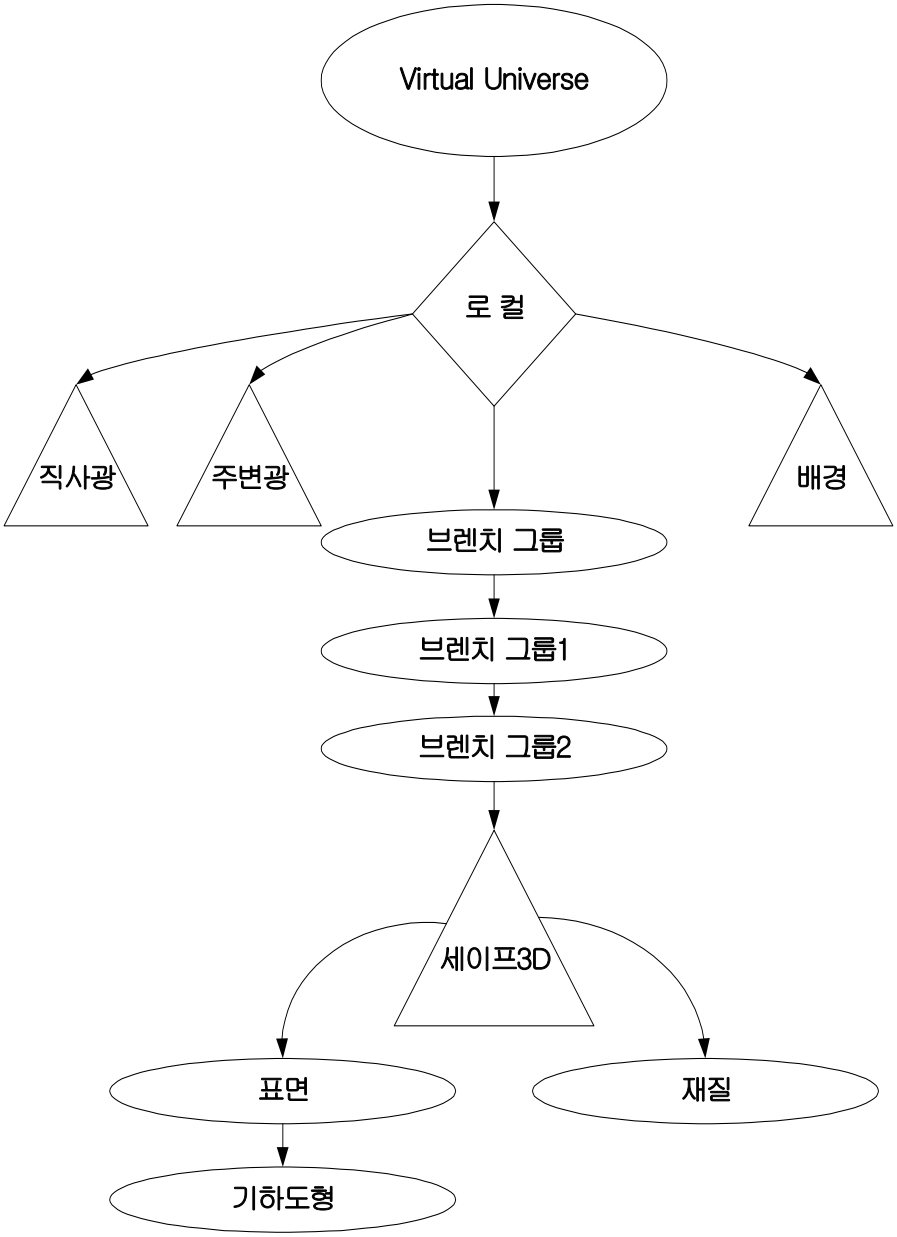

그림 13 제작한 프로그램의 장면 그래프

그림 13은 앞에서 구현된 모델링파일의 장면 그래프를 나타낸다. Java 3D에서는 VRML이나 Open Inventor와 유사한 브렌치 구조의 장면 그래프를 가진다. 먼저 가상 우주에 좌표계 설정을 위해 로컬을 붙인다. 그 다음 로컬과 3차원 객체

들은 연결하는 브랜치 그룹을 붙인다. 이 브렌치 그룹에 배경과 조명 그리고 3차원 객체를 붙인다. 그 브랜치 그룹에 먼저 변환된 객체를 화면에 알맞게 보여주기 위해 크기를 조절하는 트랜스폼 그룹을 붙인다. 그리고 그 밑에 자유로운 회전, 이동 등 갖가지 변환을 수행하는 트랜스폼 그룹을 붙인다. 이제 그 밑에 3차원 객체인 세이프 3D를 재질과 표면 속성을 설정하여 붙이면 장면 그래프는 완성된다.

2.6 엔지니어링 데이터베이스 구현

엔지니어링 데이터베이스는 제조 데이터 관리의 하위단계의 개념으로 자료저장과 조회 등의 정적인 기능을 수행하는 엔지니어링 데이터에 관련된 도면, 매뉴얼, 실험결과 등을 저장하고 관리하는 시스템이다. 현재까지의 엔지니어링 데이터베이스 시스템은 데이터를 중심으로 한 관리나 프로세스 관리 같은 것들을 프로그램으로 하드 코딩하였던 관계로 인하여 단위업무의 성력화 또는 순수 데이터 관리 측면에서는 효율성을 가졌으나, 기업의 무한 경쟁 시대에 생존 수단으로 행해지고 있는 BPR(Business Process Restructure) 또는 컨커런트 엔지니어링에 대응할 수 있는 유연성이 결여되어 있었다.

이러한 결함을 보충하여 발전된 제조 데이터 관리는 제품의 개념정의 단계부터 설계, 개발, 제조, 출하 그리고 고객 서비스에 이르기까지, 제품의 전 라이프사이클에 걸쳐 발생하는 각종 데이터와 정보의 흐름을 효율적으로 제어하고 관리하는 시스템으로, 그 주요기능은 데이터 저장 및 문서 관리, 제품구조관리, 워크플로우 관리와 분류 및 검색 기능 등으로 구분할 수 있다.[7],[8],[9]

그러나, 본 연구의 범위는 모델링 데이터의 저장, 수정 그리고 관리 등의 범위에 한정되므로, 기초적인 형태만 지닌 엔지니어링 데이터베이스를 구축하며 이를 위하여 상용화되어있는 MySQL을 이용하여 구현한다.

또한, 데이터 처리를 위한 기술은 데이터를 제어하는 기술과 데이터를 처리하는 기술로 나눌 수 있다. 데이터 제어는 자료의 처리를 위해 어떠한 경로로 데이터를 전달하여 가공하는지에 대한 비즈니스 로직을 의미하며, 데이터의 처리는 이렇게

전달된 데이터를 이용해 실제로 데이터베이스에 조회, 수정, 삭제, 입력 등의 트랜잭션을 처리하는 기능을 말한다. 이러한 기능을 위해서는 네트워크 기반에서의 Java의 RMI와 JDBC를 이용하여 이러한 데이터 처리(데이터의 조회, 수정, 삭제, 입력)등을 행한다.

2.6.1 엔지니어링 데이터베이스 시스템의 종류와 특징

엔지니어링 데이터베이스를 위해 다양한 솔루션들이 상용화되어 있지만, 각 제품은 공급업체나 그 제품이 기반으로 하고 있는 근본적인 접근 방법에 따라 매우 다른 특성을 가지고 있다.[10]

1) CAD에 기초한 엔지니어링 데이터베이스

도면 파일의 관리를 위해 CAD시스템의 공급업체들이 제공하는 CAD시스템에 추가로 공급하는 시스템으로 통상 응답 성능 면에서 회사 내의 광범위한 통합이나 CAD시스템과 중립적인 연결 등을 수행하는 능력이 부족하며, 한가지의 CAD시스템을 사용하는 회사에만 적합하다.

2) 기본적인 도구에 기초한 엔지니어링 데이터베이스

시스템 프로그래머들을 위해 만들어진 것으로 즉시 사용가능한 기본적인 구조(객체, 조건) 등을 가지고 있는 엔지니어링 데이터베이스와 통합을 개발하는 데 쓸 수 있는 도구(데이타베이스와 사용자 인터페이스)들로 구성되어있다. 이런 엔지니어링 데이터 베이스는 공급업체가 계획한 솔루션을 스스로 개발할 수 있는 기회를 주는 반면에, 작은 규모의 회사들에게는 큰 부담이 되고 고유한 제품을 만들 수 있는 잠재력을 가지고 있는 회사에게는 개별적으로 개발된 프로그램에 지나치게 의존하게 하는 위험성을 가지고 있다.

3) 통합된 연결을 가지고 있는 실사용자용 엔지니어링 데이터베이스

내부적인 연결(CAD 관리)과 외부(CAD/MRP 마스타 데이타 연결)에 필요한 중립적인 특성을 가지고 여러 가지 CAD시스템을 사용하는 환경을 관리 통합하기 위

한 실사용자 환경에 따라 변경이 가능한 모듈(예를 들어, 표준품 관리 시스템, 문서관리 시스템, MRP 시스템 연결)로 구성되어있으며, 응용과 구성에 있어서 융통성이 있으며 동시에 사용자 자신에게 개발을 위해서 특별한 프로그램 도구로 최소의 노력으로 개발하는 것이 가능하다.

2.6.2 엔지니어링 데이터베이스개발의 기법과 도구

1) 사용자 인터페이스와 사용자 기능

일반적으로 사용자 인터페이스와 기능에 대한 추세는 다음과 같다. 그림과 통합된 윈도우형의 사용자 인터페이스, 그림이 없는 윈도우형 사용자 인터페이스, 일반 문자 단말기용 등이 있다.

일반적으로 리눅스 상에서는 X 윈도우 화면만이 데이타베이스의 조회뿐만이 아니라 그림(픽셀 이미지)을 함께 볼 수 있는 유일한 방법이다.

2) 데이타 저장과 조회 기능

본 연구에서 구현된 엔지니어링 데이터베이스는 테이블에 데이타를 보관하고 여러 가지 조회 기능을 가지고 있으며, 데이타 관리의 기초를 형성하는 연결 기능을 제공하는 정도 수준이다.

2.6.3 엔지니어링 데이터베이스 계획의 단계

본 연구에서는 상용화되어 있는 엔지니어링 데이터베이스 제품에서 제품 정보 관리 데이터베이스 계획 방법을 근간으로 하여 이를 단순화 체계화하는 작업을 하였다.

본 연구에 의해 구현된 엔지니어링 데이터베이스 계획의 단계는 그림 14와 같이 5단계로 구분해 볼 수 있다.

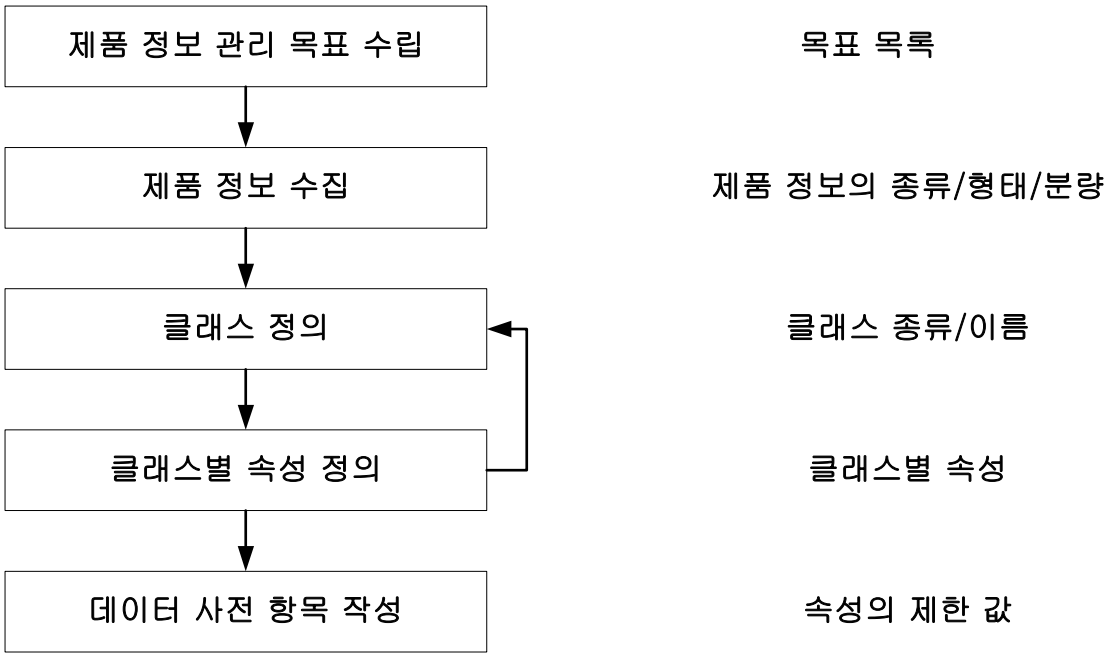

그림 14 데이터 베이스 계획의 단계

1) 제품 정보 관리 목표 수립

엔지니어링 데이터베이스 구축의 첫번째 단계는 엔지니어링 데이터베이스의 목표를 설정하는 것이다. 엔지니어링 데이터베이스의 목표는 기업의 사업 목표를 달성하는 데에 기여할 수 있는 엔지니어링 데이터베이스의 역할을 정의하고, 이러한 역할을 만족시킬 수 있는 엔지니어링 데이터베이스의 목표를 결정하여야 한다.

목표 설정 단계에서는 관리할 데이터와 이를 이용하는 엔지니어링 프로세스 관점에서 측정 가능한 목표를 설정하여야 한다. 엔지니어링 데이터베이스 구축의 측정 가능한 목표로는 문서 조회 시간, 총 문서 개수, 문서 증가율, 설계 변경 횟수, 설계 변경에 소요되는 시간 등이 있다.

2) 제품 정보 수집

제품 정보의 수집은 엔지니어링 데이터베이스을 통해서 관리될 필요가 있는 정보를 파악하여, 엔지니어링 데이터베이스의 기초를 정의하는 중요한 절차이다. 수

집해야 할 내용의 초점은 제품 정보의 종류와 그 정보의 양이다. 일반적으로 제품 정보의 종류를 파악하기 위해서는 업무에 사용되고 있는 문서를 수집함으로써 파악이 가능하다.

정보 수집의 대상이 되는 문서의 종류는 대체로 문자 파일, 2차원 도면, 3차원 모델, 유한요소해석 모델, NC 프로그램, 부품 시방서, 부품 시험 성적서, 작업 지시서, 기술검토서, 제품 개발 일정표, 기술보고서, 품질관리 지침서, BOM, 소요 자재 목록, 설계변경 요청서, 설계변경 지시서 등이다. 정보를 수집할 때는 문서 작성 도구도 함께 파악하여 사용하고 있는 응용프로그램의 종류를 알아보고, 생성되는 문서의 형태를 조사한다. 일반적으로 CAD 파일, 워드프로세서 파일, 이미지, 종이 등의 형태로 분류할 수 있다.

3) 클래스 정의

수집된 제품 정보는 일차적으로 다음과 같은 4개의 종류로 구분할 수 있다. 정보를 다음의 4가지 대분류로 분류한 뒤에 각 클래스 내에서 또 다시 서브클래스로 세분화하여야 한다.[11]

- **부품 클래스** : 제품을 구성하고 있는 실제 부품을 기술하는 정보.

예) 모델, 조립품, 표준 구매 부품, 사내 제작부품, 기계 부품, 전자 부품, 전기 부품 등

- **일반 클래스** : 제품이 아닌 다른 항목을 기술하는 정보 객체.

예) 설계 인력, 조직표, 제품 개발 프로젝트 정보 등

- **전자 양식 클래스** : 데이터베이스 속성만으로 표현할 수 있는 정보 객체.

예) 설계변경 요청서, 설계변경 지시서 등

- **문서 클래스** : 데이터베이스 속성만으로 표현될 수 없으며 첨부된 파일을 가지고 있는 정보 객체.

예) 도면, 3차원 모델, 기술검토서 등

하나의 서브클래스로 정의하기 위해서는 다음과 같은 클래스 통합과 분리의 원칙을 적용할 수 있다.

4) 클래스 통합의 원칙

1. 대부분의 속성이 같고 일부만 다르면 통합
2. 일부 속성이 다르다고 별개의 서브클래스로 분류하다 보면 클래스의 개수가 너무 많아져서 조회 시 불편해 질 수 있다.
3. 여러 개의 클래스가 같은 속성을 가지고 있고 요소의 개수가 많지 않으면 통합
4. 클래스 내의 문서들에 대한 속성이 비슷하고 하나의 클래스 내의 요소 개수가 많지 않으면, 서브클래스 종류 같은 속성을 만들고 통합하여 처리한다.

5) 클래스 분리의 원칙

1. 상당수의 요소가 속성을 채울 수 없으면 분리
2. 하나의 서브클래스 내에 저장될 모든 요소들이 채울 수 없는 속성이 있다면, 별도의 서브클래스로 분리한다.
3. 한 클래스에 10,000개 이상의 요소가 들어가면 분리
4. 문서에 대한 검색속도와, 열람 시 성능을 보장받기 위해서는 하나의 서브클래스에 10,000개 이상의 요소가 포함될 때에는 별도의 클래스로 나눈다.

2.6.4 클래스별 속성 정의

클래스의 속성이란 제품 정보 관리 시스템 내에 데이터베이스 테이블의 속성으로 정의되어, 클래스의 요소를 표현할 수 있는 각각의 항목들을 말한다.

클래스 속성을 추출하는 데에는 클래스의 요소를 표현하는 데 사용되는 속성과 클래스의 요소를 조회하는 데 사용되는 속성, 이 2가지 측면을 고려하여야 한다.

즉, 수집된 각종 정보를 분석하여 클래스에 속해 있는 요소를 기술하고 있는 특징적 관리항목을 속성으로 채택할 수 있으며, 다른 한편으로는 사용자들이 그 요

소를 찾고자 할 때 어떤 특성을 가지고 찾는가를 감안하여야 한다.[12] 따라서 정보 수집 시 사용자들이 어떤 방법으로 기존 제품 정보를 검색하고 있는 지가 파악되어야 한다. 이러한 측면이 작업계획서에 반영되어 있다. 이러한 요소의 속성은 요소 등록 시 입력되며, 검색 시 요소를 용이하게 검색할 수 있도록 지원할 수 있도록 속성을 정의해야 한다.

예를 들어 부품클래스 중에 모델이라는 서브클래스의 경우 요소를 표현하기 위한 속성으로 모델번호, 모델명 등이 속성으로 정의될 수 있으며, 사용자들이 과거에 작업한 모델을 조회할 때 외형치수를 주로 사용한다면 가로, 세로, 높이 등의 치수가 속성으로 정의될 수 있다.

2.6.5 데이터 사전 항목 작성

데이터 사전은 요소의 속성에 입력하는 값을 제어하는 역할을 한다. 데이터 사전에서는 허용되는 데이터 유형을 정의하고 특정 속성에 입력할 수 있는 값을 제한한다.

2.6.6 MySQL를 이용한 시스템 구성

본 연구에서 사용된 MySQL은 3.23.49버전으로 각 모델링 데이터를 클래스 단위로 나누고, 아래 표8에서 보이는 것처럼 같이 모델링 파일의 속성을 테이블로 구성하였다.

Field	Type	Key	Default
Class	char	PRI	
Part No.	int	PRI	
Part Name	char		N
Description	char		N
Commission			N
Material	int		N
Creator	char		N
Create Data	data		N
Tolerance	float		N
Weight	float		N
Surface	boolean		N
Unit	char		N

표 8 MySQL에서 생성된 속성 테이블

2.7 2계층 방식의 네트워크를 이용한 연동

네트워크가 컴퓨팅 환경의 기반을 이루게 되면서 이기종 컴퓨터간 분산 컴퓨팅 환경의 중요성이 부각되고 있으나, 이것은 각 컴퓨팅 환경을 구성하는 플랫폼의 다양성 때문에 매우 복잡하고 어려운 작업이다. Java 엔터프라이즈 플랫폼은 분산 컴퓨팅을 위한 다양한 API를 제공하여 응용프로그램의 생산성 향상 및 플랫폼 독립성을 제공하고 있다. CORBA와 Java의 연동을 위한 JavaIDL, 원격 Java 메서드 호출을 위한 RMI, 관계형 데이터베이스와의 연결을 위한 JDBC 등의 API를 이용하여 개발자는 Java라는 단일 플랫폼 하에서 분산 응용프로그램을 작성할 수 있다.[13] 본 연구에서는 이러한 JDBC와 RMI에 대한 전문적인 기술을 요구하는 수준이 아니며, 간단한 사용 방법 정도의 언급과 소개정도에 관한 설명만 언급하고자 한다.

Java 분산 응용프로그램을 작성하기 위한 모델로는 일반적인 클라이언트/서버 모델로 알려져 있는 2-계층 구조와 서버측을 비즈니스 논리 계층과 데이터베이스 계층으로 분리한 3-계층 구조가 있다.[14] 현재 대부분의 분산 응용프로그램이 채택하고 있는 2-계층 구조의 경우, 비즈니스 논리 계층은 클라이언트측에서 실

행되는 응용프로그램에 포함되어 있으며 특정 업무 수행을 위해 서버 측 데이터베이스에 대한 접근 및 이후의 처리를 수행하게 된다. 그러나 전체 시스템의 크기가 커지고 클라이언트의 수가 많아지면 각 응용 프로그램 간에 동일한 역할을 수행하는 비즈니스 논리 계층을 포함하게 되는 중복 문제가 발생하게 될 것이다. 바꾸어 말해서 비즈니스 논리가 바뀌게 된다면 관련된 모든 응용프로그램의 수정이 불가피하게 된다. 잘 설계된 3-계층 구조 시스템의 경우 이와 같은 중복이 발생하지 않으며 다른 계층에 영향을 주지 않고 비즈니스 논리 계층을 변경할 수 있다. Java 분산 컴퓨팅 API를 이용하여 응용프로그램을 작성할 때는 구성하고자 하는 시스템에 적절한 분산 컴퓨팅 모델과 API를 선택하는 것이 효과적인 결과를 얻을 수 있는 기본이 된다. 그림 15 과 그림 16은 각각 Java 분산 컴퓨팅 API를 이용하여 구성 가능한 2-계층 및 3-계층 시스템을 보여준다.

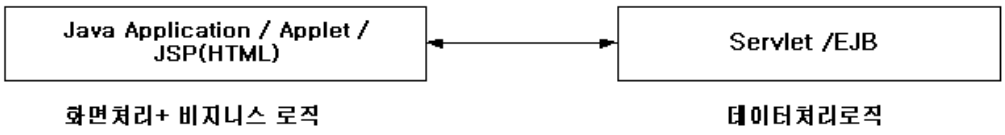

그림 15. 2-계층구조

그림 16. 3-계층구조

2.7.1 JDBC

데이터베이스는 대용량의 데이터를 체계적으로 저장하고 가공하는데 필수적이며 따라서 많은 자료를 처리해야 하는 기업용 프로그램의 경우 특히 많이 사용된다. 또한 관계형 데이터베이스의 경우, 매우 오랜 역사를 가지고 있으며 현재는 거의

대부분의 데이터베이스 프로그램은 어떤 식으로든 이 형식의 데이터베이스를 SQL문으로 제어해서 데이터를 다루고 있다.

JDBC는 SQL문을 통해서 데이터베이스를 제어할 수 있게 지원하는 라이브러리이다. JDBC는 데이터베이스를 직접 제어하지 않으며 JDBC 드라이버를 거치게 된다. 보통은 JDBC드라이버가 데이터베이스시스템에 직접 연결되어 있는 것이 일반적이다. 또는 중간에 미들웨어를 두어서 프로그램과 데이터베이스가 일대일이 아닌 다대다의 대응이 될 수 있다. ODBC의 경우와는 다르게 Java에서는 연결 전에 JDBC 드라이버를 로딩해 주어야 한다는 점이 다르다.

MySQL과의 연동을 위해 JDBC 2.0.8 드라이버를 설치하고 다음과 같은 방식으로 구현하였다.

1) MySQL 관련 JDBC 클래스를 찾기 위해

```
Class.forName("org.gjt.mm.mysql.Driver").newInstance();
```

2) 데이터베이스와 연결하기 위한 JDBC URL은 다음과 같은 형식으로 사용한다.

```
jdbc:mysql:///mysql?user=lans&password=mysqllans
```

2.7.2 RMI

RMI는 네트워크로 연결된 원격지에 존재하는 다른 Java 객체의 메소드를 호출하는 것이다. 이는 Java의 Java.rmi 패키지에 의해서 지원되는 새로운 통신 방식이라고 할 수 있다. RMI의 이점은 네트워크상의 어느 위치에 있는 Java 메소드도 마치 자신의 컴퓨터에 있는 것과 같이 호출 할 수 있다는 것이다. 즉, 자신의 컴퓨터, Local LAN 또는 WAN상의 다른 시스템에 존재하는 Java 객체를 구별하지 않고 똑같은 방식으로 다룬다. 또한 객체의 메소드를 직접 호출하는 효과이기 때문에 메시지 전달이나 제어가 손쉬워 소켓 프로그래밍에서처럼 복잡한 프로토콜을 구현할 필요가 없다. 소켓 프로그래밍에서는 프로토콜에 따른 메시지 교환이나 확장의 문제 등 객체지향 프로그래밍의 장점을 살릴 수 없는 요소가 어느 정도

존재했었다. 그러나 RMI는 마치 네트워크가 아닌 프로그램 내의 다른 객체의 메서드를 호출하는 듯한 기능을 제공하므로 이전의 객체 지향적설계를 그대로 적용할 수 있다. 따라서 프로토콜을 사용할 경우 복잡해 질 수 있는 메시지 해석과 오류 처리 과정이 없어져 프로그래밍은 쉬워지고 기능은 견고해진다. 또한 RMI서버와 클라이언트 개발을 어느 정도는 분리시킬 수 있으며 확장에 있어서도 객체지향의 장점을 활용할 수 있다.

2.7.3 네트워크 기반 엔지니어링 데이터베이스 시스템의 구현

그림 17은 구현된 엔지니어링 데이터베이스 시스템의 화면이다. 이것은 초기 실행 시에 나타나는 화면으로 각각의 창에 원하는 검색 조건을 입력하고 검색 버튼을 누르면 아래와 같이 모델링 데이터파일에 관한 정보가 나타나게 된다. 포함된 클래스 그룹과 파일 이름, 도면 번호, 제작자, 생성된 날짜 등의 정보가 나타나고 있는 것을 볼 수 있다. 그림 18은 그림 17에서 검색된 파일의 상세정보를 보여주는 창이다. 그림 19는 검색된 파일을 새로운 클래스 그룹을 만들고 그 클래스로 이동하기 위하여 클래스 그룹을 수정하는 화면이다.

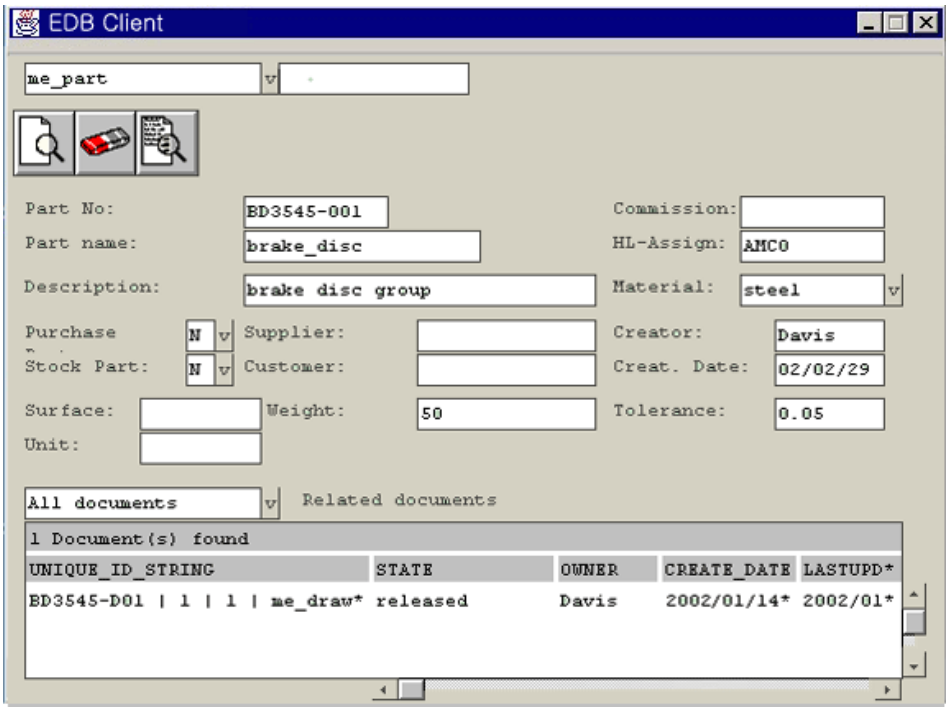

그림 17 모델링 데이터의 검색과 속성 보기

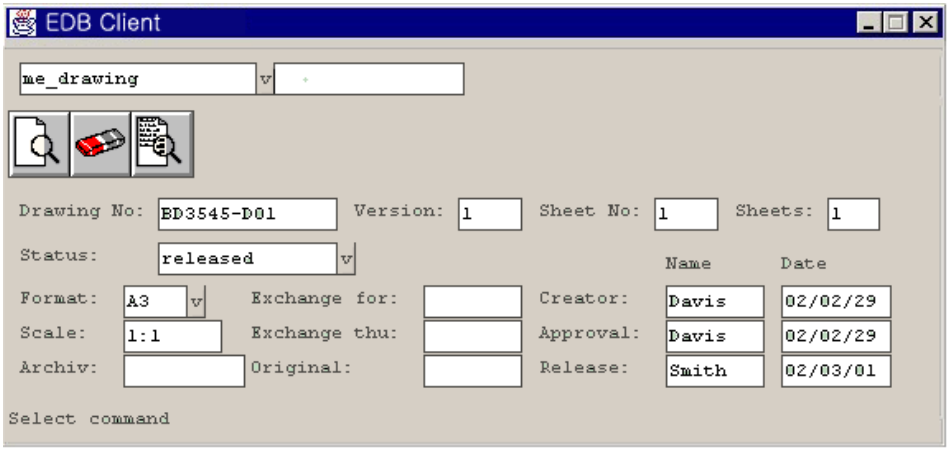

그림 18 검색된 모델링 데이터 파일의 상세 속성보기

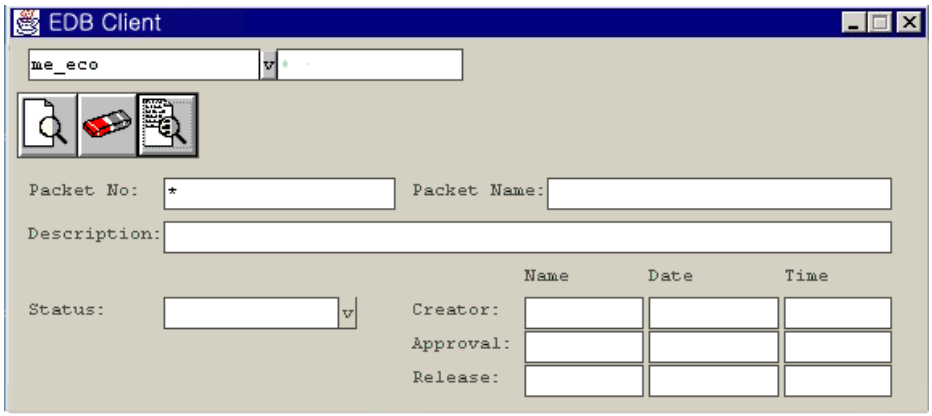

그림 19 모델링 데이터 파일의 상세 보기 2

이것의 동작 원리는 다음과 같다.

1. 클라이언트인 윈도우 시스템에서 서버인 linux로의 접속을 시도한다. 이때 Java RMI 프로토콜을 참조하는 연결을 가지고 있다. 이 연결은 리눅스 서버의 객체를 호출한다. Java RMI는 클라이언트 스터브를 포함할 수 있는 데, 리눅스 서버 상의 객체를 호출한다. 서버와 클라이언트 사이에 연결된 세션은 연결 종료를 결정할 때까지 지속된다.
2. 서버로부터 호출된 객체 사이의 참조가 이루어져 MySQL에 저장된 테이블의 정보를 다운로드 받는다.
3. 클라이언트가 화면상에 테이블의 정보를 로드한다.

제 3장 결론

본 연구의 목적은 상용화된 어플리케이션에서 사용 중인 협업설계시스템의 기본이 되는 뷰어를 중소기업에서도 사용할 수 있는 저가의 시스템으로 구현해 봄으로써 기술의 연구와 보완을 위한 과제로 삼고자 한다. 특히, 웹 기반으로의 확장을 위한 엔지니어링 데이터베이스와 표준화된 모델링 데이터인 STEP, IGES 등의 파일을 로더하기 위한 로더 개발은 연구와 보완을 필요로 한다.

1. 모델링 데이터를 시각화하는 모듈은 Java 3D에서 지원하는 OBJ파일과 Java 3D에서 지원하지 않는 포맷인 ASE를 로더하는 두 가지의 데이터 포맷을 이용하였다. 본 연구에서 적용하고자 하는 모델링 뷰어 기술은 모델링 데이터 시각화를 포함한 CAD/CAM분야에 한정되는 기술이 아니라, 의학, 시뮬레이션 등의 분야에 적용할 수 있다. 그러나, 고급화 되고 전문화되기 위한 어플리케이션이 되기 위해서는 표준화된 모델링 데이터인 STEP (Standard for Exchange of Product Data), IGES (Initial Graphics Exchange Specification) 등의 파일을 로더하기 위한 로더 개발은 연구와 보완을 필요로 한다. 또한, 사용자 작동에 의한 실시간 설계 변경과 모델링 데이터의 수치적인 해석을 위하여 로더 자체의 추가적인 확장은 연구와 보완을 필요로 한다.

2. 실시간 사용자 작동 모듈은 Java 3D에서 기본적으로 지원하는 Behavior 클래스를 이용하였으며, 마우스와 키보드의 입력 장치를 사용하여 시각화된 데이터를 관찰하기 위한 다양한 시점 제어 방식과 객체 이동 제어 방식을 적용하였다. 그러나, 다른 여러 가지 입력 장비와의 연동을 위한 제어방식의 연구와 보완이 필요하다.

3. 엔지니어링 데이터베이스는 단순히 데이터의 관리, 수정 등의 기능으로 제한하였으며, 모델링 데이터 관리를 위해 Linux 기반의 서버로 구축하여 데이터 관리를

위해 MySQL을 이용하였다. 그러나, 중소기업형 PDM시스템으로의 확장을 위해서는 사용상의 편리성, 신뢰성 등을 향상하기 위하여, 구체화, 체계화된 부품 클래스, 상용화된 오퍼레이팅 시스템과 데이터베이스 어플리케이션 등을 필요로 한다.

4. 2-계층 네트워크 기반의 구조는 Java에서 지원하는 RMI방식으로 구현되었으며, 서버-클라이언트 간의 데이터 공유와 가상환경에서의 작업을 지원할 수 있다. 이것은 단순화된 네트워크 환경 하에서 연동 가능하도록 한 것이며, 복잡한 네트워크 내에서의 연동을 위해 3-계층 방식으로의 확장과, CORBA와 같은 통신 방식으로 연구 보완하여야 한다.

참고 문헌

[1] John Vince, "Virtual Reality Systems", Addison-Wesley, 1995.

[2] 원광연 외, "Virtual Reality 기술을 이용한 3D Assembler 개발", 한국과학기술원, 1999.

[3] 성운재, “협업 가상 환경에서의 동시성 제어 모델”, 한국과학기술원, 1999.

[4] CoCreate, "CoCreate Installing CoCreate WorkManager, CoCreate, 1997.

[5] Sun MicroSystem, "http://Java3d.co.kr/tutorial/j3dapi/".

[6] Aaron E. Walsh, "Core Web3D", PH PTR.

[7] Ed Miller, "PDM today", CIM data, 1996.

[8] OMG MfgDTF, "PDM Enablers Joint Proposal to th OMG in Response to OMG Manufacturing Damain Task Force RFP1", OMG, 1998.

[9] CIM data, "Product Data Management", A Technology Guide, 1994.

[10] EP, Eigner+Partner GmbH, "Administration HandBook", Eigner+Partner GmbH, 1992.

[11] Harrelson et al, Bill Harrelson and Alan Mendel, "Vendor Selection & Evaluation", CIM data PDM Conference '96 Tutorial Proceedings, 1996.

[12] David A. Taylor, "Object-Oriented Technology", A Manager's Guide. Addison-Wesely Publishing Company, 1990.

[13] Robert Orfali 외, “CORBA, Java and the Object Web", BYTE Vol. 22 No.10, 1997.

[14] David A. Taylor, "Business Engineering with Object Technology", john Wiley & AMP Sons, Inc. 1995.