공학석사 학위논문

조선 *M&S* 환경구축을 위한 그래픽 사용자 인터페이스 시스템 구현 및 비교분석

Implementation and Analysis of Graphic User Interface Systems for Shipbuilding Modeling & Simulation Environment

지도교수 남 종 호

*2008*년 *2*월

한국해양대학교 대학원

해양시스템공학과

김 대 희

本 論文을 김 대 희의 工學碩士 學位論文으로 認准함.

위원장 : 공학박사 박 주 용 (인)

위 원 : 공학박사 김 원 돈 (인)

위 원 : 공학박사 남 종 호 (인)

2008년 2월

한 국 해 양 대 학 교 대 학 원

해양시스템공학과 김 대 희

Abstract

This paper describes the development of graphic user interface (GUI) systems for the visualization of graphical results of engineering analyses or performance.

Three representative computer graphic libraries, OpenGL, OpenCASCADE, and OpenGL Performer, are selected based on their performance and functionalities. Those libraries are integrated to Microsoft Foundation Class (MFC) to form a graphic window frame. Advantages and disadvantages of each GUI system are discussed to suggest a way to choose an appropriate graphic library suitable to a specific visualization purpose.

The performance of the constructed GUI systems are analyzed with respect to their rendering capability. A complicated geometric model and its sub models are adopted for rendering with the standard data formats such as IGES, STEP, as well as other formats used in popular CAD systems. The results of rendering performance verifies the known capability of the graphic libraries.

List of Figures

List of Tables

- 목 차 -

1. 서론

일반적으로 컴퓨터 프로그래밍에서 라이브러리(Library)는 어떤 특정한 기능들을 모아 놓은 함수 들을 참조하여 사용할 수 있는 형식을 말한다. 나아가 컴퓨터 그래픽스 라이브러리라 함은 컴퓨터 그래픽에 관련된 그래픽스의 구현을 위한 함수들의 집합을 의미한다. 현재 CAD/CAM/CAE 을 포함한 모든 분야에서 3차원 그래픽을 지원하는 라이브러리들이 각광받고 있다. 물론 3차원 그래픽 라이브러리는 2차원 환경을 기본적으로 지원한다.

그래픽 라이브러리는 각각의 라이브러리에 따라서 제작사가 다르고 그 사용법 및 특성이 다르기 때문에 어떠한 응용분야에 적용할지에 따라서 해당 라이브러리의 선택여부가 중요하다. 하지만 기본적으로 라이브러리가 공통적으로 지원하는 함수기능이 있다. 그 기능들은 다음과 같다.

1. 그래픽스 라이브러리의 초기화 또는 그래픽 카드의 초기 설정 등의 초기화 함수
2. 점, 선, 원, 타원 등의 기하학적 원형요소(geometric primitives)를 다루는 함수들
3. 이미지표현, 텍스쳐 맵핑, 조명등 특수 기능 함수들

그래픽 응용프로그램을 구현하기 위해서 선이나 점과 같은 기하학적인 요소들을 그리는 함수들을 직접 구현 하려면 상당한 시간과 노력이 필요하다. 그리고 라이브러리를 작성하기 위해서는 프로그래밍 지식뿐만 아니라 하드웨어 지식 또한 필요하게 된다. 요즘에 운영체제 즉, 마이크로 소프트 윈도우즈, 레드햇 리눅스 등은 자체 그래픽 API(Application Program Interface)함수들을 지원하고 있다. 하지만 이 API함수들은 일반적으로 처리속도가 느리고 성능이 떨어진다. 그리고 다양한 그래픽 기능들을 지원하지 못하며 특히 3차원을 지원하지 못한다는 것이 최대 단점이다. 따라서 구현하고자 하는 그래픽 응용 프로그램이 좀 더 복잡하고, 처리속도가 빨라야 하며, 성능이 좋아야 하고, 3차원을 완벽히 지원해야 한다면 목적에 맞는 그래픽 라이브러리를 선택하여 응용 프로그램을 구현해야만 한다.

그래픽 라이브어리들은 방대한 함수들로 구성되어 있고, 내부의 알고리즘 또한 대단히 복잡하다. 이 라이브러리들을 제대로 이해하기 위해서는 특정 라이브러리들의 매뉴얼을 참고하여 사용법을 익혀야만 한다. 하지만 이런 방대한 수의 함수들을 다 이해할 필요는 없다. 특정 그래픽 라이브러리를 사용할 때 중요한 점은 먼저 그 라이브러리에 대한 기본적인 사용방법, 사용하는 운영체제에 대한 환경설정, 그리고 기본적인 라이브러리의 구성 및 구조 등을 숙지한 다음, 사용하려는 함수를 라이브러리 매뉴얼을 참조하여 응용프로그램 상에 구현하면 된다.

이러한 그래픽 라이브러리를 이용하여 운영체제에서 제공하는 그래픽 API함수들을

이용하지 않고도 원하는 그래픽 응용프로그램을 비교적 손쉽게 구현할 수 있다. 컴퓨터 라이브러리를 위한 그래픽 라이브러리는 그 수가 많으며 다양한 목적에 맞게 각각 모듈화 되어 있다. 물론 그중에서는 거의 업계표준화처럼 자리 잡은 OpenGL과 같은 라이브러리도 있다. 그리고 라이브러리 중에서는 공개버전(open source)과 상용화 버전이 있으며 그 종류와 기능 또한 라이브러리에 따라 다양하다. 하지만 모든 그래픽 라이브러리는 컴퓨터 그래픽을 위한 기본적인 함수들을 제공한다는 점에서 공통점을 가지고 있다.

1.1 연구 배경

선박 가시화 프로그램의 중요성을 인지할 때, 현재 가장 많이 사용하는 그래픽 라이브러리와 MFC를 기반으로 GUI 시스템의 환경구축 필요성이 매우 높다. 본 논문에서는 GUI 시스템 환경 구축방법을 연구하고 구체적인 GUI 시스템을 구현하여 실용적인 시스템기반기술을 확보하고자 한다.

1.2 연구 목적

선박 가시화 프로그램의 환경구축을 위하여 각 그래픽 라이브러리와 MFC를 기반으로 하여 GUI 시스템 구현하고 렌더링 성능 테스트를 통하여 어느 GUI 시스템이 렌더링에 최적화되는지 연구하는 것을 목적으로 한다.

1.3 연구 내용

현재 가장 많이 사용하는 3가지의 그래픽 라이브러리 즉 OpenGL, OpenGL Performer, OpenCASCADE와 GUI 시스템을 구현하기 편리한 MFC를 사용하여 선박 가시화 프로그램 환경을 구축하며, 각 모델 데이터를 이용하여 OpenGL 기반 GUI 시스템과 OpenGL Performer 기반 GUI 시스템의 렌더링 성능 테스트 한다. 그리고 OpenGL 기반 GUI 시스템과 OpenGL Performer 기반 GUI 시스템간의 직접적인 렌더링 성능 테스트를 통하여 어느 GUI 시스템이 렌더링에 최적이 되었는지를 연구한다.

2. Graphics Library 및 API

2.1 OpenGL

2.1.1 OpenGL의 구성

OpenGL은 그래픽스 하드웨어에 대한 소프트웨어 인터페이스로서, 대화형 3차원 애플리케이션 제작에 필요한 오브젝트나 연산을 구성하는데 사용할 수 있는 250여개의 커맨드로 구성되어 있다.

다양한 하드웨어 플랫폼에서 구현될 수 있도록 간결하고 하드웨어 독립적인 형태로 설계되었다. 이와 같은 하드웨어 독립성을 유지하기 위해, 윈도우즈 관련 작업을 수행하거나 사용자 입력을 받아들이기 위한 커맨드는 애초부터 OpenGL에 포함되지 않았다. OpenGL에서는 3차원 오브젝트 모델을 쉽게 표현하기 위한 높은 레벨의 커맨드도 제공하지 않는다. 그래서 OpenGL로 복잡한 객체를 모델링할 때는 오직 선, 점, 폴리곤 등과 같은 기하학적 원형요소(geometric primitives)만 사용해야 한다. 이러한 작업을 보다 쉽게 처리하기 위해서는 GLU(OpenGL Utility Library)와 같은 별도의 라이브러리가 필요하다. GLU는 곡선 및 곡면을 그릴 수 있는 기능을 제공하고 있다 [1].

Fig2.1은 OpenGL의 렌더링 파이프라인의 구성도이다. 대부분의 OpenGL 구현은 이와 같은 일련의 처리단계를 거친다. 렌더링 파이프라인은 OpenGL이 어떻게 처리되는지의 과정을 보여주는 역할을 한다. 정점데이터는 평가자와 정점연산을 거치는데 반해 픽셀데이터 는 다른 경로를 거치게 된다. 그러나 데이터들은 결국 마지막 단계인 레스터화 및 프라그먼트 연산에서 만나게 된다. 그리고 최종적으로 생성된 픽셀데이터는 프레임버퍼에 저장되고 화면에서 가시화된다.

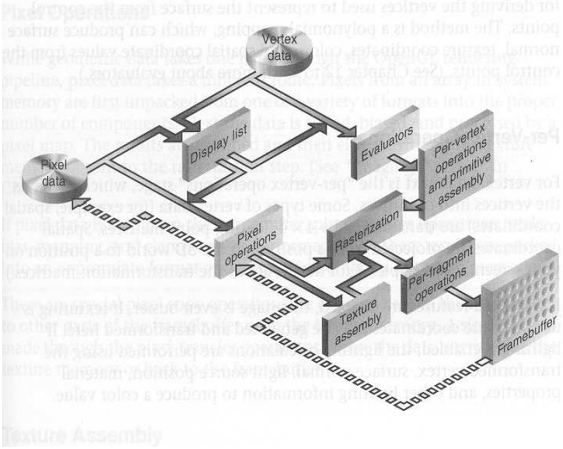


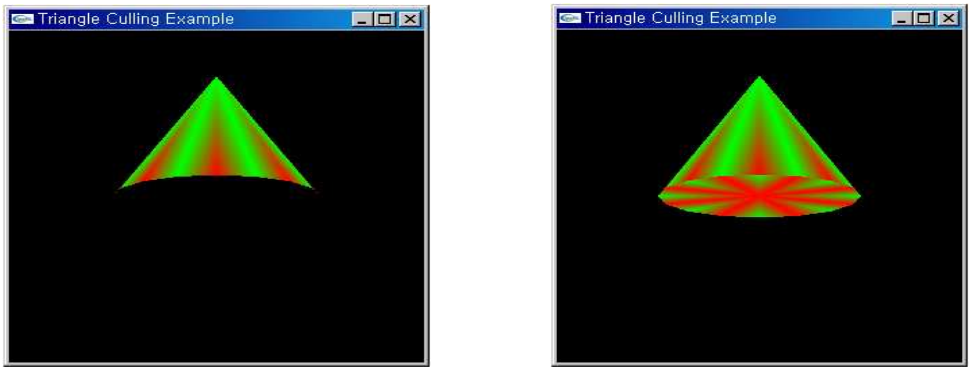
Fig.2.1 OpenGL rendering pipeline [2]

2.1.2 OpenGL API

OpenGL API는 숨은면제거와, 블렌딩효과, 안티-에일리어싱, 텍스쳐 맵핑, 조명효과, 그리고 대기효과를 주는 안개(fog) 등 다양한 기능들을 제공한다.

- 숨은 면 제거 (Hidden-Surface Removal)

다른 객체에 의해 가려진 솔리드 객체의 일부분을 제거하는 것을 말한다. Fig.2.2(a)는 은면이 제거되지 않아 앞에 있는 면이 뒤에 있는 면을 가리지만, Fig.2.2(b)는 은면이 제거되어 앞에 있는 면이 가리지 않는다.



(a) Non visible surface (b) Visible Surface

Fig.2.2 Effect of hidden-surface removal

- 블렌딩(Blending)효과

블렌딩은 화면상의 색상과 물체를 혼합하여 새로운 이미지를 만드는 것이다. 두 이미지가 겹쳐있는 듯한 효과를 얻고자 할 때 사용되는 기법이다. 그리고 블렌딩을 사용하면 그림자 효과도 얻어 낼 수 있다. Fig.2.3은 원형의 모델에 블렌딩효과를 사용하여 그림자 효과를 표현한 것이다.

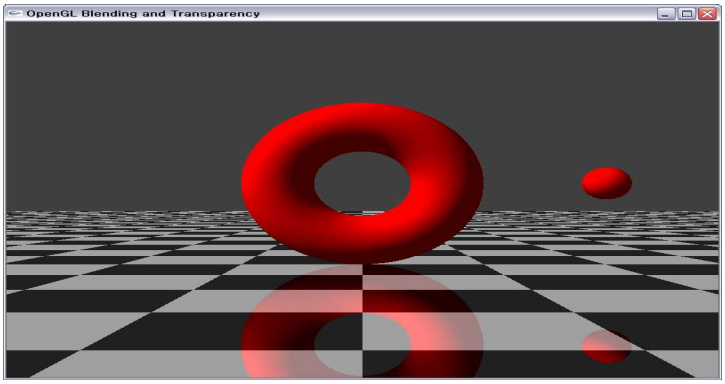


Fig.2.3 Reflection effect with blending

- 텍스쳐 맵핑

단순화 3차 물체의 각 면에 다양한 이미지를 입력 3차원 물체를 더욱더 현실감 있게 만드는 방법이다. Fig.2.4는 원형의 모델에 텍스쳐 맵핑을 하여 현실감 있게 모델을 표현한 것이다.

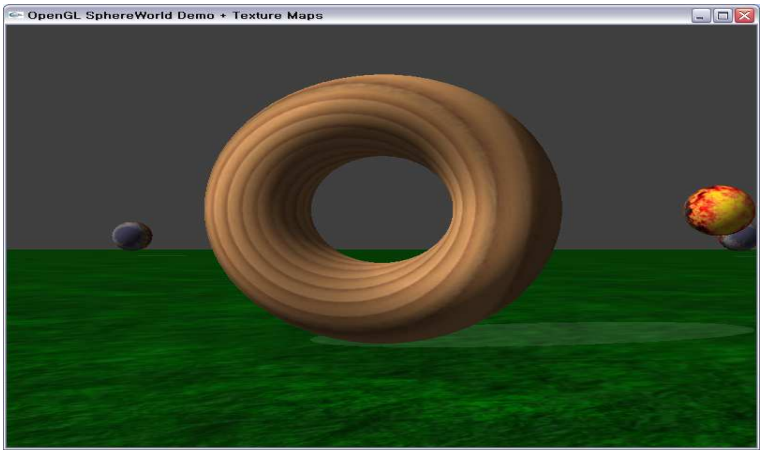


Fig.2.4 Texture mapping

- 조명(lighting)효과

3차원 물체의 각 면의 색상을 조절하고, 조명의 위치와 각도를 적절하게 넣으면 3차원 물체를 더욱더 사실적인 물체를 표현할 수 있다. Fig.2.5는 원형의 모델에 조명효과 주어 사실감 있게 현실감 있게 표현한 것이다.

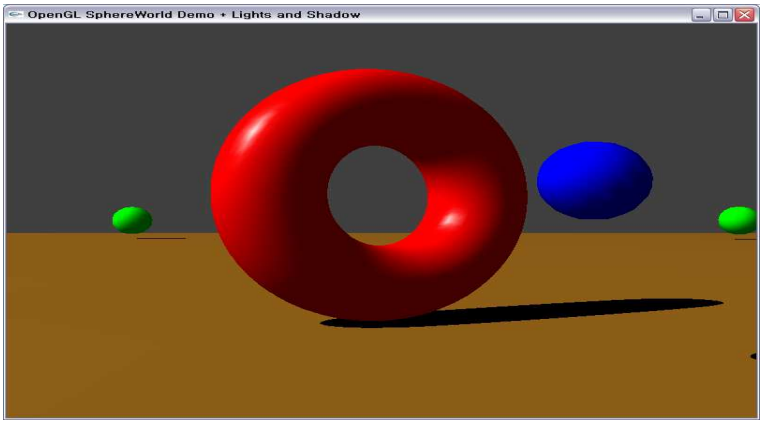
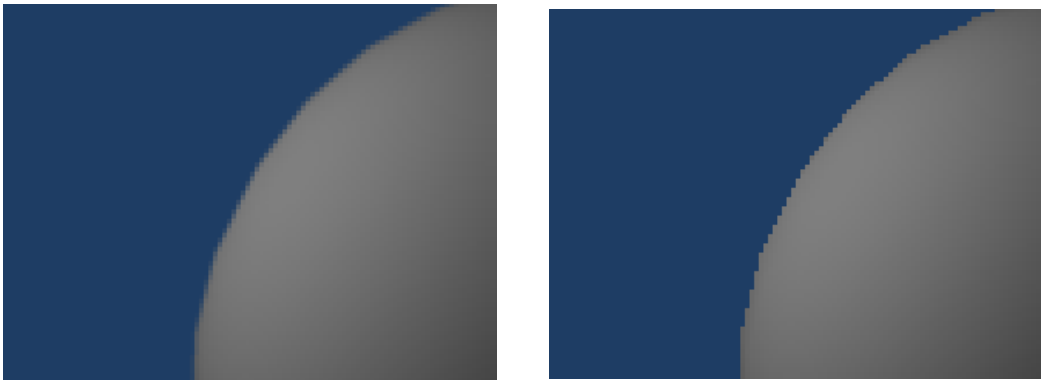


Fig.2.5 Lighting effect

- 안티 에일리어싱

스크린 위에 대각선을 비스듬하게 그으면 화면 이미지를 나타내는 픽셀 때문에 울퉁

불퉁하게 나타난다. 즉 계단 현상이 생기는 것이다. 이런 울퉁불퉁한 선을 에일리어싱이라고 한다. 안티 에일리어싱은 주변 색깔과 배경 색을 혼합해 울퉁불퉁한 효과를 부드럽게 하고 자연스럽게 보이게 하는 것이다. Fig.2.6(a)은 안티에일어싱 처리 전이기 때문에 계단 현상이 발생하여 매끄럽지 못하다. 그러나 (b)는 안티에일어싱 효과를 주어 계단 현상이 없어져 부드러운 이미지를 표현한 것이다.



(a) Jagged line before anti-aliasing (b) Smoothed line after anti-aliasing

Fig.2.6 Anti-aliasing

- 안개(Fog) 효과

안개는 장면 내의 물체를 흐릿하게 표현하는 방식의 특수 효과이다. 안개의 농도를 적절히 조절하면 현실감을 있는 효과를 제공하고 있다. Fig.2.7은 사각형인 모델에 안개효과를 주어 현실감 있게 표현한 것이다.

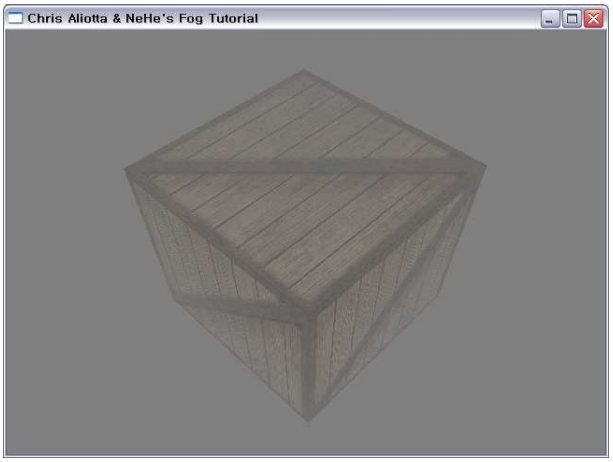


Fig.2.7 Fog effect

2.2 OpenCASCADE

2.2.1 OpenCASCADE의 구성

OpenCASCADE는 강력한 CAD/CAM/CAE 커널(Kernel)이며 3차원 모델링을 위한 개발 플랫폼이다. 재사용이 가능한 C++ 라이브러리들로 구성되어 있으며 공개 버전이므로 자유롭게 개발할 수 있는 소프트웨어이다. OpenCASCADE는 3차원 표면, 솔리드 모델링, 가시화(Visualization), 데이터 교환과 같은 빠른 응용 개발이 가능한 컴포넌트를 포함하고 있다.

Fig.2.8는 OpenCASCADE의 6개 모듈구성을 나타내고 있다. 6개의 모듈은 크게 Foundation Class, 모델링 데이터(Modeling Data), 모델링 알고리즘(Modeling Algorithms), 가시화(Visualization), 데이터교환(Data Exchange), 애플리케이션 프레임워크(Application Framework)로 구성되어 있다 [3].

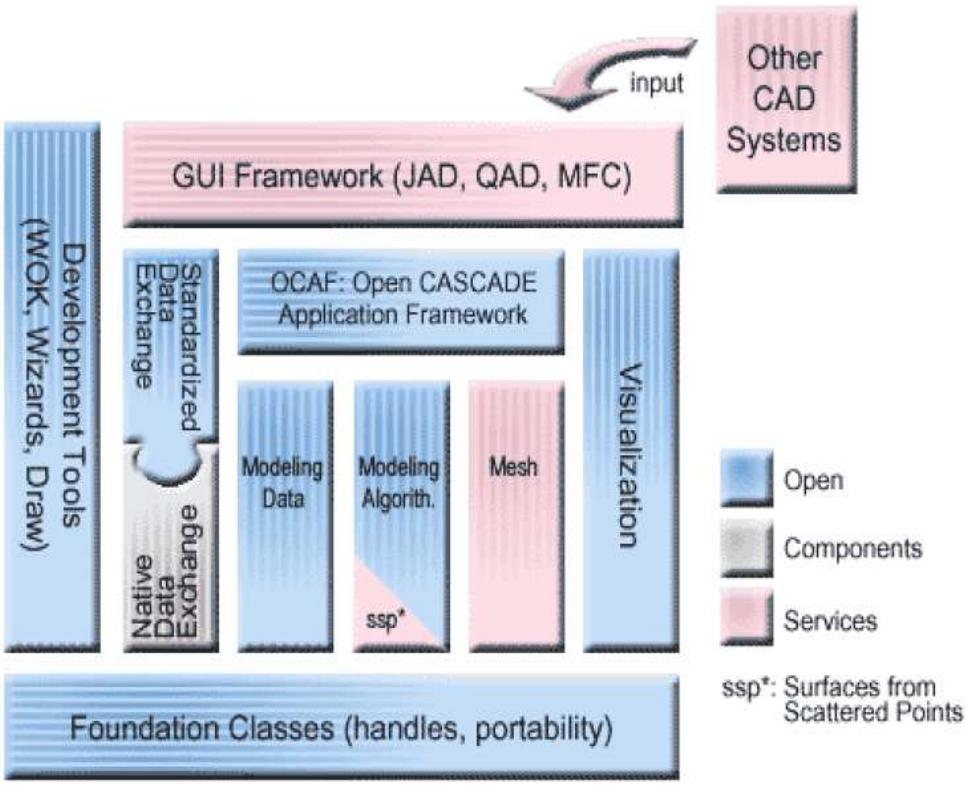


Fig.2.8 Modular structure of OpenCASCADE

2.2.2 OpenCASCADE API

Fig.2.9는 각 모듈들이 포함하고 있는 기능들이다. Foundation Class는 OpenCASCADE에서 커널부분이 되는 클래스이다. 여기에는 수학도구와 각종 CASCADE의 기본데이터 타입들이 포함되어 있다. 모델링 데이터는 2차원과 3차원 기하학적 모델에 대한 자료구조를 포함하고 있다. 모델링 알고리즘은 이미 만들어진 원형 모델, 부울 연산, 모따기에 대한 알고리즘이 포함하고 있다. 가시화는 모델링한 객체를 컴퓨터 모니터에 디스플레이해주는 함수들을 포함하고 있다. 데이터 교환은 IGES, STEP등 CAD의 표준파일 포맷을 임포트(import) 또는 익스포트(export)해주는 함수를 포함하고 있다. 애플리케이션 프레임워크는 CAD 응용프로그램을 작성할 때 빠른 구현을 가능하게 하는 프레임형식이다 [3].

Foundation Classes	Modeling Data	Modeling Algorithms	Visualization	Data Exchange	Application Framework
Kernel Classes Math Utilities	2D Geometry 3D Geometry Geometry Utilities Topology	Construction of Primitives Boolean Operations Fillets and Chamfers Offsets, Drafts Sewing and Sweeps Features Hidden Line Removal Geometric Tools Topological Tools	Services Common to 2D and 3D 2D Visualization 3D Visualization	IGES STEP AP203 AP214 Extended data exchange (XDE)	Data Framework Data Storage Application Desktop

Fig.2.9 Object library modules and their contents

- Foundation Class

  OpenCASCADE의 가장 핵심적인 클래스로써 기본적인 데이터 타입이 정의되어 있고 메모리 관리, 예외 처리, 문자 타입 등 OpenCASCADE의 핵심적인 요소들이 정의 되어 있다. 그리고 모델링에 필요한 수학도두와 기본적인 기하학적 형태들도 정의되어 있다. 다음은 Foundation Class가 제공하는 기능들이다.

  - 기하학적 원형요소, 문자
  - 힙 메모리 자동 관린
  - 예외처리

- 벡터, 매트리스 등 수학 관련 유틸리티

- 모델링 데이터(Modeling Data)

모델링 데이터는 2차원과 3차원 기하학적 모델을 표현하기 위한 자료구조를 정의한다. Fig.2.10은 기본적인 기하학요소인 점, 선, 곡선들을 이용하여 모델을 표현한 것이다.

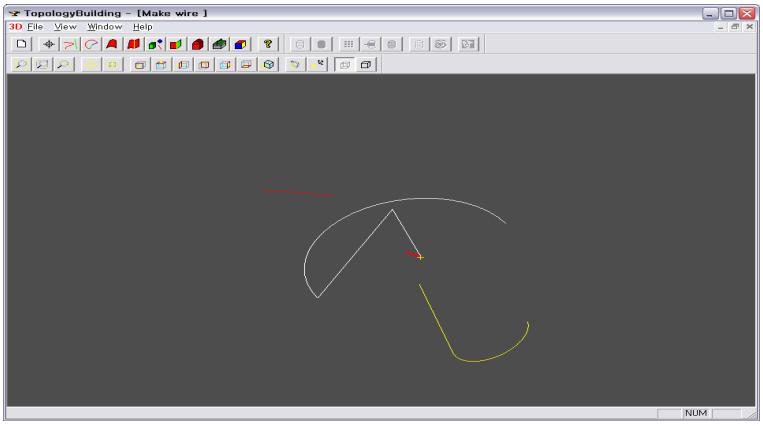
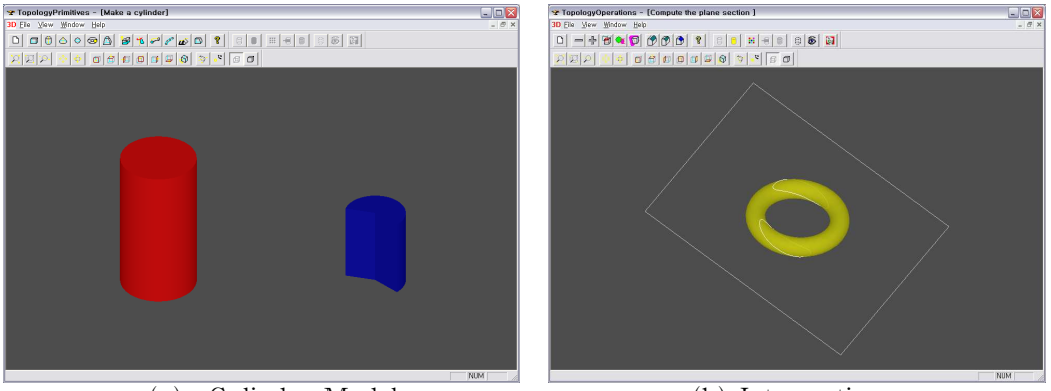


Fig.2.10 Basic geometry of Modeling Data

- 모델링 알고리즘(Modeling Algorithms)

모델링 알고리즘은 프리즘(Prisms), 실리던(Cylinders), 원통(Cones)등과 같은 이미 만들어진 원형 모델을 표현하기 위한 알고리즘, 자르기(Cut), 분할(Section)등의 부울 연산, 모따기 등과 같은 알고리즘들을 제공하고 있다. Fig.2.11(a)은 모델링 알고리즘을 이용하여 실리던 모델을 표현한 것이고 Fig.2.11(b)은 평면과 원형모델간의 교차를 시킨 것이다.



(a) Cylinder Model (b) Intersection

Fig.2.11 Modeling Algorithms

- 가시화(Visualization)

모델을 디스플레이를 해주는 기능이다. 이 가시화를 이용하면 모델에 조명효과, 색상 등의 속성들을 변화시켜 모델을 좀 더 현실감 있게 표현할 수 있다. Fig.2.12는 원통에 조명효과를 주어 현실감 있게 표현한 것이다.

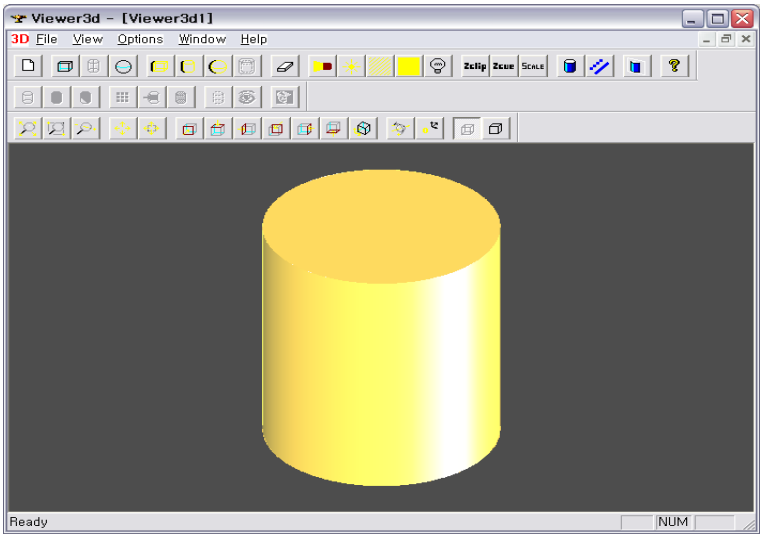
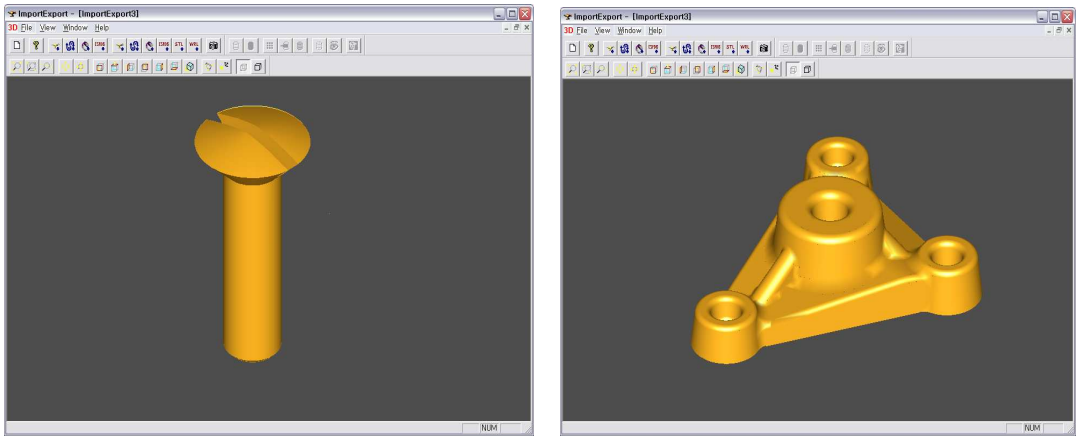


Fig.2.12 Lighting effect

- 데이터 교환(Data Exchange)

데이터 교환은 OpenCASCADE에서 제공하는 IGES와 STEP포맷을 지원한다. Fig.2.13 (a)은 IGES 파일을 로딩한 것이고, Fig.2.13(b)은 STEP 파일을 로딩한 것이다.



(a) IGES Loading (b) STEP Loading

Fig.2.13 Data exchange

2.3 OpenGL Performer

OpenGL Performer는 실시간 3D Graphics, 실시간 비주얼 시뮬레이션을 위한 그래픽스 라이브러리이다. OpenGL Performer는 다음과 같은 응용 분야에 활용되고 있다.

- 비주얼 시뮬레이션과 가상현실
- 고사 양을 요구하는 컴퓨터 그래픽 분야
- 빠른 렌더링속도를 필요로 하는 실시간 시뮬레이션

렌더링을 요구하는 응용프로그램들은 OpenGL Performer를 사용함으로써 이러한 요구들을 충족시킬 수 있다.

Table 2.1은 OpenGL Performer의 주요 라이브러리들을 나타내고. Dynamic Shared Object(DSO)는 IRIX(Silicon Graphics사의 UNIX 버전)와 리눅스 시스템의 주요 라이브러리이며 그 확장자는 .so다. Dynamic link Library(DLL)는 마이크로소프트 윈도우즈의 주요 라이브러리이며 그 확장자는 .dll이다 [4].

Table 2.1 OpenGL Performer libraries

DSO/DLL Name	Header File	Description
`libpf.so` `libpf.dll`	`pf.h`	Main OpenGL Performer library. Contains *libpf*, which handles multiprocessed database traversal and rendering, and *libpr*, which performs the optimized rendering, state control, and other functions fundamental to real-time graphics.
`libpfdu.so`	`pfdu.h`	Library of scene and geometry building tools that greatly facilitate the construction of database loaders and converters. Tools include a sophisticated triangle mesher and state sharing for high-performance databases.
`libpfutil.so` `libpfdu-util.dll`	`pfutil.h`	Utility functions library. Note that `libpfdu-util.dll` is a combination of `libpfdu` and `libpfutil`.
`libpfui.so` `libpfui.dll`	`pfui.h`	User interface library.
`libpfv.so` `libpfv.dll`	`pfv.h`	A graphical viewer library that provides for the easy construction of applications.
`libpfmpk.so` `libpfmpk.dll`	`pfmpk.h`	Library for importing display-configuration information from files using the OpenGL Multipipe SDK configuration file format.
`libpfdb`	`pfdb.h`	Collection of libraries containing the load, convert, and store routines for numerous file formats.

Fig.2.14는 OpenGL Performer 라이브러리와 운영체제간의 관계이다. 모든 OpenGL Performer는 다음과 같이 운영체제와 그래픽 라이브러리의 계층으로 구성되어 있다 [4].

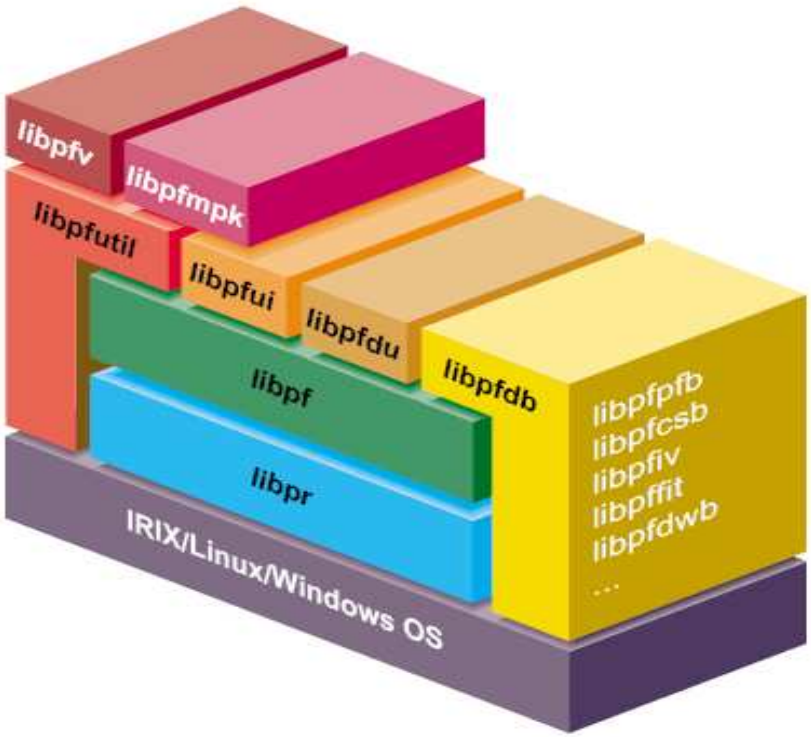


Fig.2.14 Hierarchy of OpenGL Performer library

3. 그래픽 라이브러리 기반 *GUI* 시스템 구현

본 장에서는 MFC와 3가지 주요 그래픽 라이브러리 (OpenGL, OpenCASCADE, OpenGL Performer)를 기반으로 GUI 시스템을 구현하고 각 GUI 시스템을 비교, 분석한다.

3.1 Microsoft Foundation Class(MFC)

MFC란 윈도우즈 프로그래밍에 필요한 약 300개의 클래스를 계층적으로 구현해 놓은 클래스 라이브러리이다. MFC는 GUI 프로그램을 작성하기에 매우 막강한 기능들을 포함하고 있다. MFC를 사용하면 프로그램 개발 기간을 단축시킬 수 있으며 이것을 이용하지 않은 경우보다 프로그램 개발 효율을 수십 배나 높일 수 있기 때문에 본 연구에서는 MFC를 활용하였다 [5].

MFC에는 다음과 같은 클래스가 제공된다.

- MFC의 기반 클래스
  - CObject
    - 클래스 자신의 정보를 디스크에 저장하는 기능
    - 자신이 어떤 클래스인지에 대한 정보를 넘겨주는 기능
- 애플리케이션 프로그램의 뼈대를 이루는 클래스
  - CFramewnd
    - 윈도우의 이동, 크기조절, 최소화, 최대화 등 윈도우 자체를 제어하는데 필요한 모든 기능
  - CDocument
    - 데이터를 저장, 읽기, 처리하는 모든 기능
  - CView
    - 프로그램에서 작업하는 데이터를 화면에 보여주는 역할을 하는 모든 기능
  - CWinApp
    - FramWnd, Document, View를 묶어주고 프로그램을 구동시켜주는 역할
- 그래픽 관련 클래스
  - 다양한 그래픽 기능이 있는 클래스 제공
- 자료구조 클래스
  - 데이터를 저장하기 위한 다양한 자료구조 클래스들을 제공
- 파일 및 데이터베이스 관련 클래스
  - 데이터를 파일로 저장하는 기능과 파일을 데이터베이스에 저장하는 클래스 제공
- 인터넷 관련 클래스

- 인터넷을 이용한 애플리케이션을 쉽게 만들기 위해 클래스 제공

3.2 OpenGL 기반 GUI 시스템

3.2.1 OpepGL 기반 GUI 시스템의 구성

Fig.3.1은 OpenGL 기반 GUI 시스템의 전체적인 구성도를 나타낸 것이다. 이 GUI 시스템은 MFC에 OpenGL과 VRML 파서(parser), 3ds 로더가 포함되어 있다.

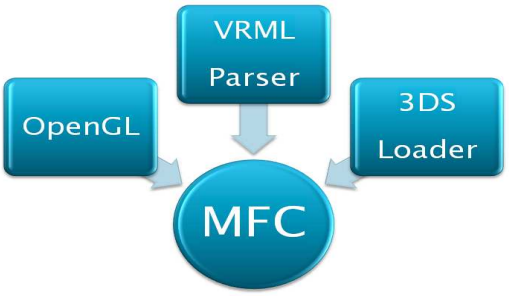


Fig.3.1 OpenGL and MFC based GUI System

OpenGL은 GLUT를 사용하여 윈도우 기반으로 렌더링 할 수 있다. 그리고 GLUT를 사용함으로써 키 이벤트(key event), 마우스 이벤트(mouse event)등의 콜백함수도 간단히 구현 할 수 있다. 그렇지만 GLUT만을 이용하여 GUI 시스템을 구현하기는 매우 어렵다. 따라서 본 논문에서는 GUI 시스템을 구현하기 위해서 MFC를 이용하였다.

3.2.2 WGL(windows GL)

OpenGL과 MFC를 이용하여 GUI 시스템을 구현하기 위해서는 몇 가지 초기 값 설정을 해주어야 한다. 먼저 OpenGL과 MFC를 연결하기 위해서는 마이크로소프트사에서 제공하는 wgl(Window GL)이라는 별도의 함수를 이용한다. 즉, wgl은 OpenGL과 윈도우즈 API를 연결시켜 MFC에서 OpenGL을 사용하게 하는 함수이다. 다음은 연결에 중요한 wgl 함수들에 대한 설명이다 [6].

- wglCreateContext();
  - GDI 장치문맥에 대한 OpenGL 렌더링 문맥을 만들고 그에 대한 핸들을 돌려준다.
- wglDeleteContext();
  - 렌더링 후에 렌더링을 제거하는 함수다.
- wglMakeCurrent();
  - 주어진 렌더링 문맥을 사용하여 렌더링을 할 수 있게 만든다.

3.2.3 PIXELFORMATEDSCRIPTOR

픽셀 포맷은 직접 그릴 객체의 대상, 즉 윈도우나 비트맵에 대한 컬러 비트의 구조에 속성을 지정한다. 픽셀코드를 지정하는 구조체는 PIXELFORMATDESCRIPTOR이다. 이 구조체는 마이크로소프트사에서 제공하며 크기, 버전 넘버, 속성 플래그, 컬러 비트 수, 어큐뮬레이터(accumulator), 깊이버퍼, 스텐실(stencil), 보조버퍼 등의 지정, 메인 층(layer)타입 등의 정보를 지정할 수 있다. Fig.3.2는 PIXELFORATEDSCRIPTOR의 구조를 보인다 [6].

```
typedef struct tag PIXELFORMATDESCRIPTOR {
  WORD  nSize;
  WORD  nVersion;
  DWORD dwFlags;
  BYTE  iPixelType;
  BYTE  cColorBits;
  BYTE  cRedBits;
  BYTE  cRedShift;
  BYTE  cGreenBits;
  BYTE  cGreenShift;
  BYTE  cBlueBits;
  BYTE  cBlueShift;
  BYTE  cAlphaBits;
  BYTE  cAlphaShift;
  BYTE  cAccumBits;
  BYTE  cAccumRedBits;
  BYTE  cAccumGreenBits;
  BYTE  cAccumBlueBits;
  BYTE  cAccumAlphaBits;
  BYTE  cDepthBits;
  BYTE  cStencilBits;
  BYTE  cAuxBuffers;
  BYTE  iLayerType;
  BYTE  bReserved;
  DWORD dwLayerMask;
  DWORD dwVisibleMask;
  DWORD dwDamageMask;
} PIXELFORMATDESCRIPTOR;
```

Fig.3.2 PIXELFORMATDESCRITOR

Fig.3.3는 COpenGLView의 클래스 다이어그램이다. 여기에는 OpenGL과 MFC의 초기 설정 함수들이 정의되어 있다. CreateViewGLContext()함수에는 OpenGL과 윈도우를 연결시키는 API인 wgl이 정의되어 있다. SetWindowPixelFormat() 함수에는 픽셀 코드를 설정하는 PIXELFORMATDESCRIPTOR 구조체가 정의되어 있으며 OnCreate()

함수에는 OpenGL의 조명, 텍스쳐, 배경색등 모든 값이 초기화되어 있다.

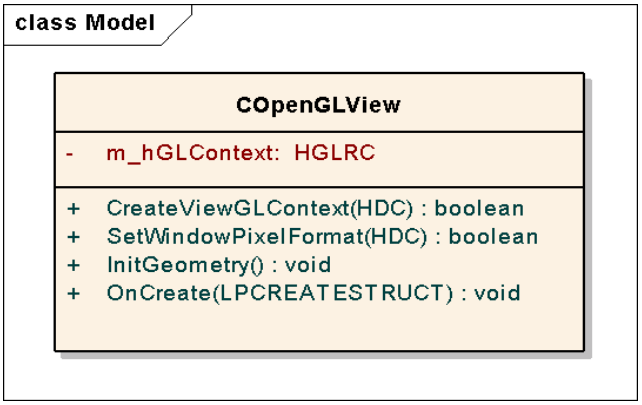


Fig.3.3 COpenGLView class diagram

3.2.4 파서 (Parser)

VRML 파일을 임포트(import)하기 위해서는 VRML 파서가 필요하다. 이 파서는 공개 버전이지만 구현하려는 GUI 시스템에 맞게 직접 수정할 필요가 있다. 본 연구에서는 OpenGL 기반 시스템에 맞추어 파서를 구현하였다. Fig.3.4는 VRML 파서의 주요 클래스 다이어그램이다.

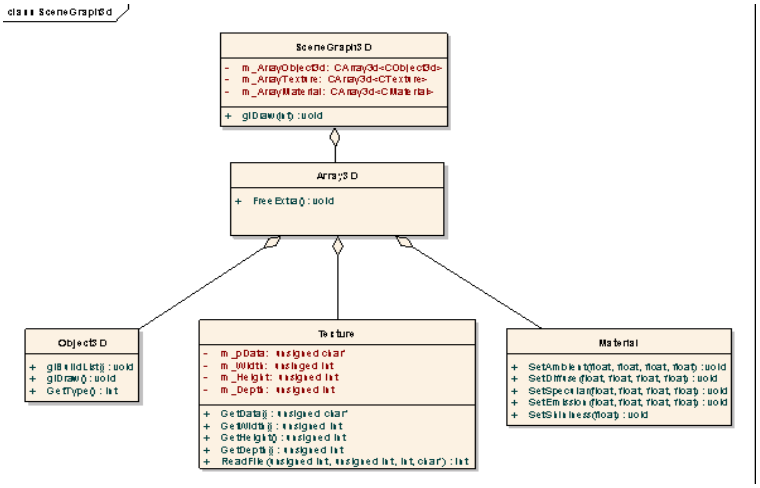


Fig.3.4 VRML parser class diagram

Fig.3.5는 OpenGL 기반 GUI 시스템과 VRML파서를 이용하여 VRML 모델을 로딩한 것이다.

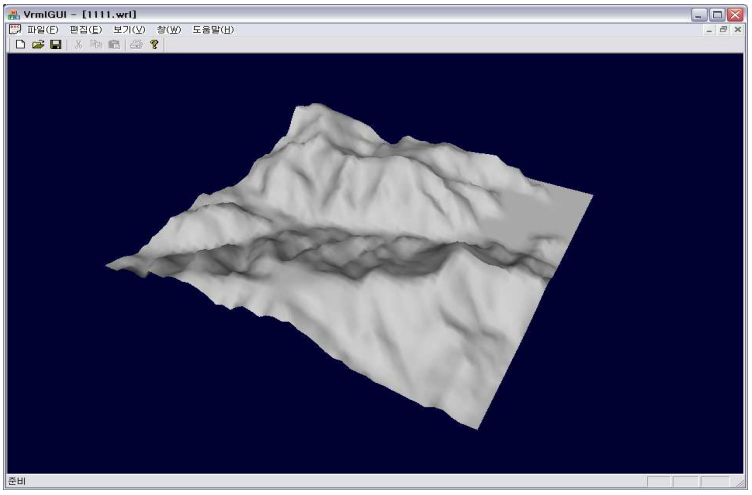


Fig.3.5 An image loaded by VRML parser

3ds파일는 Autodesk사의 공개된 파일 포맷이다. 3ds파일의 구조는 청크(chuck)의 집합으로 구성되어 있다. 하나의 청크 ID, 다음 청크의 번지 ID에 해당하는 데이터 순으로 파일이 구성되며, 청크 ID는 2바이트 워드 형태이다. 3ds파일을 읽는 순서는 “청크 ID -> 다음 청크의 주소 -> 청크 데이터”의 순서로 정보를 읽는다.

3ds파일 로딩을 하기 위해서 공개버전을 사용하였다. 구현하려는 GUI 시스템에 맞게 수정하여 3ds로더를 다시 구현하였다. Fig.3.6은 3ds의 클래스 다이어그램이다.

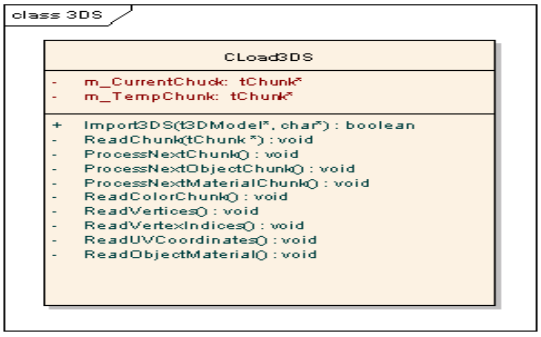


Fig.3.6 3ds class diagram

Fig.3.7은 OpenGL 기반 GUI 시스템과 3ds로더를 이용하여 3ds 모델을 로딩한 것이다.

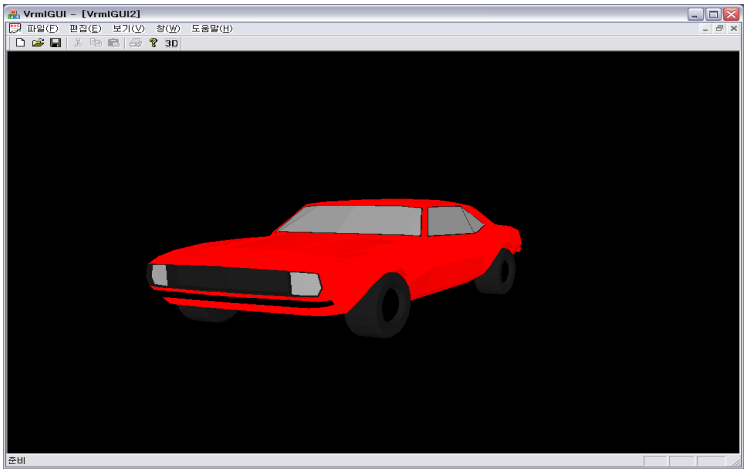


Fig.3.7 An image loaded by 3ds parser

구현된 OpenGL 기반 GUI 시스템에서는 VRML, 3ds파일 로더를 라이브러리화한 후 MFC에 포함시켰다.

3.3 OpenCASCADE 기반 GUI 시스템

3.3.1 OpenCASCADE 기반 GUI 시스템의 구성

Fig.3.7은 OpenCASCADE 기반 GUI 시스템의 전체적인 구성을 나타낸 것이다. 이 GUI 시스템은 MFC에 OpenCASCADE과 IGES와 STEM 파일, 커스텀 위저드(Custom Wizard)로 구현하였다.

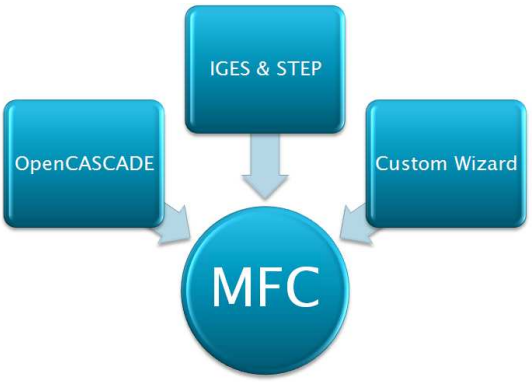


Fig.3.8 OpenCASCADE and MFC based on GUI System

CAD 시스템에서 모델링된 데이터를 임포트하기 위해서 중립파일 형식인 IGES와 STEP을 이용하였다. IGES와 STEP 파일포맷은 OpenCASCADE에서 기본적으로 지원

하고 있다. OpenCASCADE에서 응용 프로그램 개발을 돕기 위해 뷰어 기반 애플리케이션, OCAF(OpenCASCADE Application Framework) 기반 애플리케이션의 MFC Wizard를 제공한다 [7]. 본 연구에서는 뷰어 기반의 커스텀 위저드를 사용자정의에 맞게 직접 구현하여 기본적인 프레임을 구현하였다. Fig.3.9는 커스텀 위저드를 사용하여 기본 프레임을 구현한 것이다.

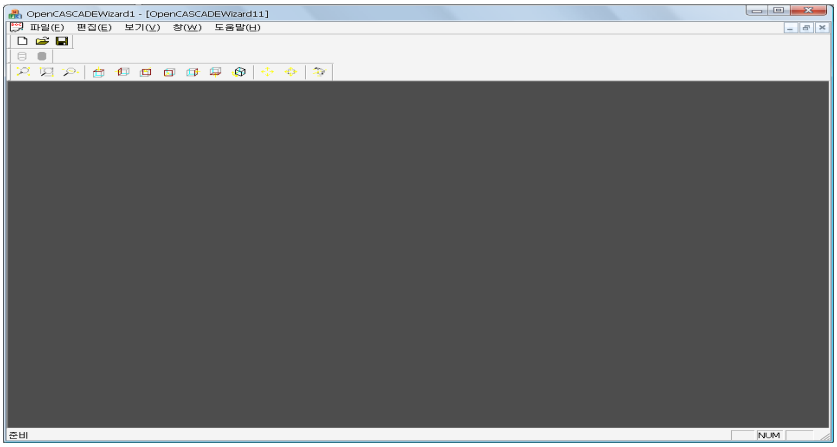


Fig.3.9 Base frame of GUI system

3.3.2 OpenCASCADE 기본적인 프레임 구성

OpenCASCADE는 MFC와 연결하여 사용할 수 있도록 라이브러리를 제공하고 있다. Fig.3.10은 MFC와 CASCADE를 연결하는 주요 클래스 다이어그램이며 초기설정 메서드들이 포함되어 있다. 이 초기 설정 메서드들은 CASCADE에서 기본적으로 제공하고 있다. IGES와 STEP 파일을 임포트하기 위해서 기본 프레임에 메뉴바와 툴바에 각각 아이콘을 추가하였다. 그리고 사용자정의에 맞게 필요한 툴바와 메뉴바를 구현하여 최종 프레임을 구현하였다.

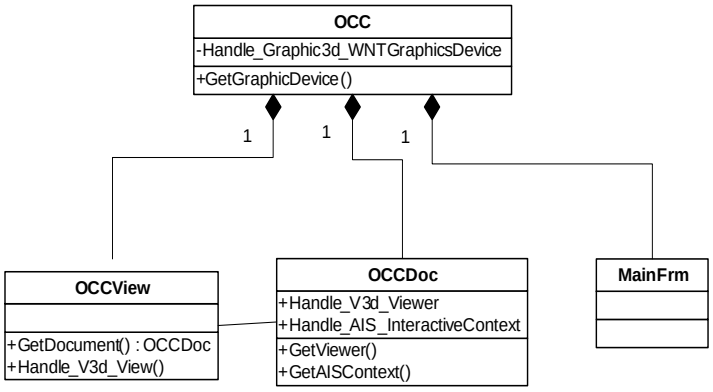


Fig.3.10 Class diagram of OpenCASCADE and MFC

Fig.3.11은 사용자가 필요로 하는 툴바와 메뉴바를 구현하여 최종적으로 GUI System을 보인다. 이 툴바의 기능으로 기본적인 Curve와 Surface 데이터를 가시화할 수 있으며 모델을 다양한 시각에서 볼 수 있는 뷰잉 기능을 표시할 수 있다.

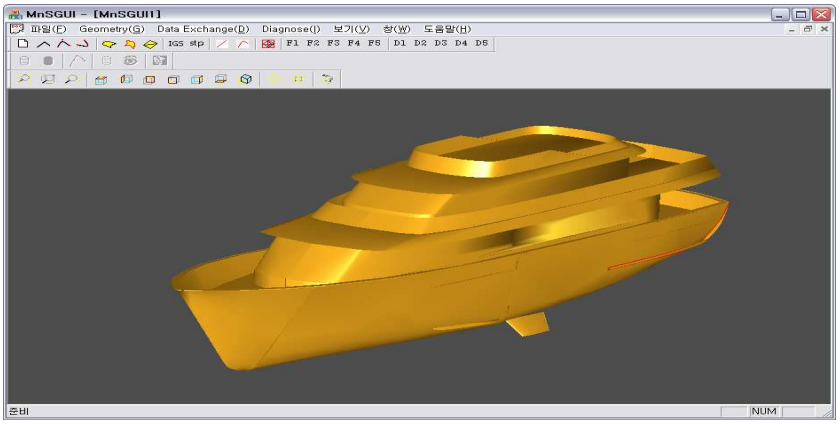


Fig.3.11 OpenCADCADE and MFC based GUI System

3.4 OpenGL Performer 기반 *GUI* 시스템

3.4.1 OpenGL Performer 기반 *GUI* 시스템 구성

Fig.3.12는 OpenGL Performer 기반 GUI 시스템의 전체적인 구성을 나타낸 것이다. 이 GUI 시스템은 MFC를 기반으로 OpenGL Performer, 3ds 로더, OpenFlight 포맷을 연계하여 구현되었다.

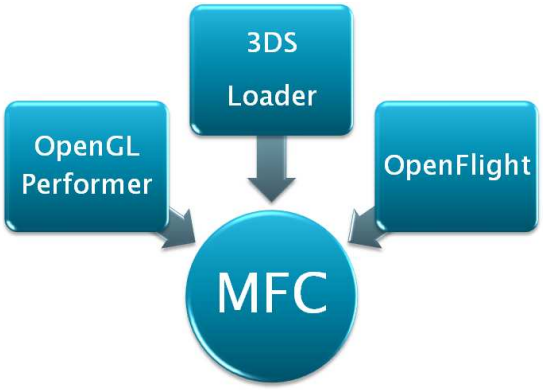


Fig.3.12 OpenGL Performer and MFC based on GUI System

*3.4.2 OpenGL Performer*의 렌더링 구성

OpenGL Performer는 모델인 씬그래프(scene graph), 모델을 보여주는 역할을 하는 채널(channel), 모델을 렌더링 하는 파이프(pipe), 최종적으로 모델을 디스플레이하는

윈도우로 구성되어 있다.

Fig.3.13은 OpenGL Performer가 모델을 렌더링 하여 최종적으로 디스플레이를 하는 과정을 나타내는 그림이다. 씬그래프는 모델의 모든 비주얼 데이터가 포함되어 있으며 모델에 조명효과, 텍스쳐 맵핑 등 다양한 효과 기능을 포함하고 있다. pfChannel은 카메라와 같은 역할을 한다. 이것은 단지 비주얼 데이터를 뷰 할 수 있으며, 카메라 위치와 뷰잉프러스텀(viewing frustum)등의 기능을 가진다. pfPipe는 뷰잉프러스텀 안에서 비주얼 데이터를 pfPipeWindow에 렌더링을 한다. 그리고 렌더링된 장면을 최종적으로 pfPipeWindow에서 디스플레이를 한다 [4].

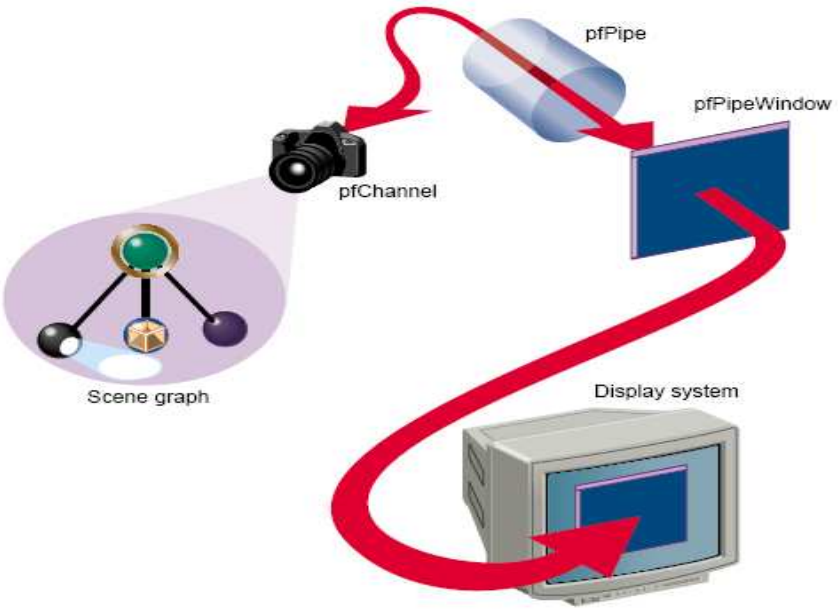


Fig. 3.13 Path of rendering

다음은 OpenGL Performer를 이용하여 애플리케이션을 실행하는 기본적인 순서이다.

1. OpenGL Performer 초기화
2. 파이프 생성
3. 윈도우와 채널 생성
4. 채널을 윈도우에 연결
5. 관련된 채널에 모델 데이터 할당
6. 채널에 모델 위치 설정
7. 모델을 드로잉하기 위해서 pfFrame()함수 호출
8. 시뮬레이션하기 위해 6번 과정을 반복

Fig.3.14 Performer 애플리케이션 의 액티브 다이어그램으로 애플리케이션을 구현할 때 Performer가 기본적으로 동작하는 과정을 나타낸 것이다.

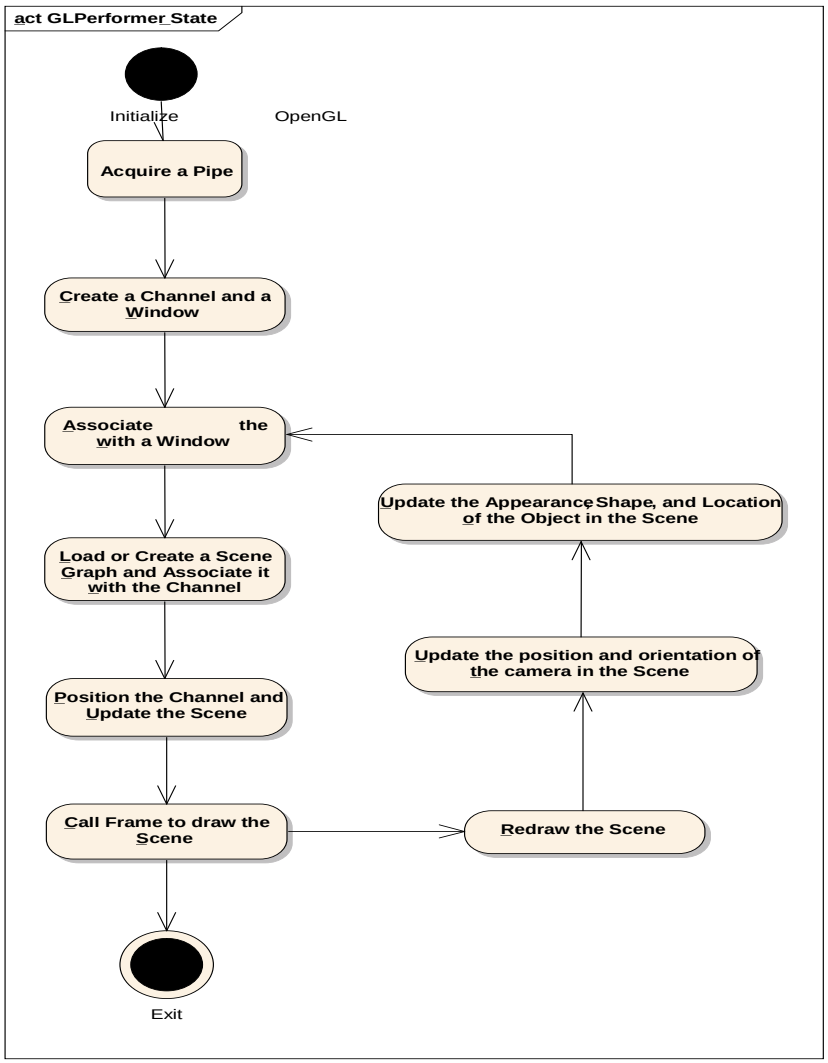


Fig.3.14 Activity diagram of OpenGL Performer application

3.4.3 OpenGL Performer 기반 GUI 시스템 프레임 구성

MFC는 기본 GUI 프레임을 생성하기 위해서 사용되었고, OpenGL Performer API를 이용하여 모델을 가시화한다. OpenGL Performer 모델의 확장자는 Performer 그래픽 라이브러리가 기본적으로 제공하는 OpenFlight 확장자인 .flt를 사용하였다.

Fig.3.15는 MFC의 기본 프레임을 형성하는 클래스들의 관계이다. CWinApp 하부에 CDocument, CView, CFrame이 포함되어 있으며, CWinAPP가 최종적으로 프레임을

구동한다.

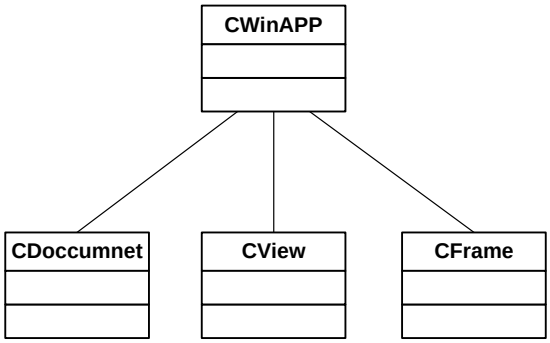


Fig.3.15 Class Diagram of MFC

Fig.3.16는 MFC의 CView에 OpenGL Performer 주요 API를 포함한 관계를 보인다.

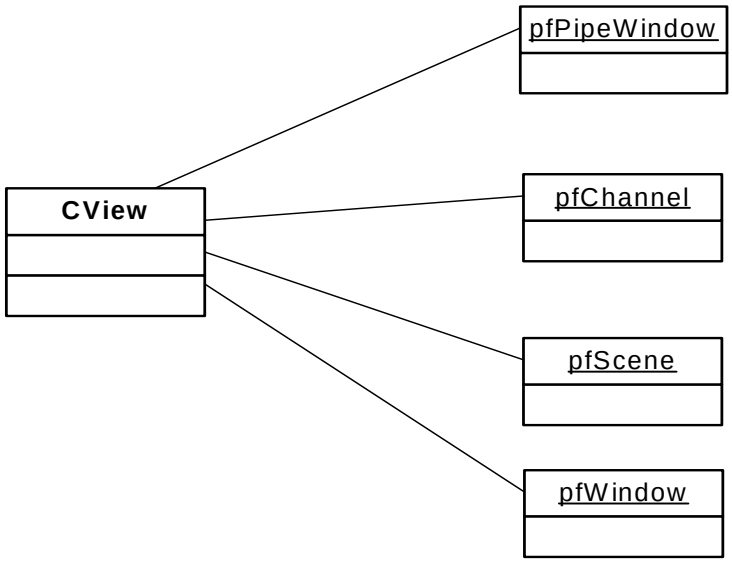


Fig.3.16 MFC and OpenGL Performer API

Fig.3.17은 MFC와 OpenGL Performer를 이용하여 GUI 프레임을 구현한 것으로 모델을 임포트하기 위한 모든 기본적인 세팅이 완료된 프레임이다.

Fig.3.17 MFC and OpenGL Performer based on GUI system

3.5 GUI 시스템의 비교 및 분석

다음 Fig.3.18은 구현한 세 가지 GUI 시스템을 비교분석하여 장단점을 나타낸 것이다.

Fig.3.18 Comparison of 3 GUI systems

OpenGL 기반 GUI 시스템에서는 MFC와 연결을 할 때 wgl이라는 별도의 함수가 필요하였다. 윈도우즈에서 제공하는 환경설정 구조체를 사용하여 구현하였다. 시스템 구현 후에 시스템이 안정성을 보였다는 장점이 있지만, 사용자가 필요로 하는 기능들이

있으면, 직접 코딩을 하여야 하는 단점이 있다.

OpenGL Performer 기반 GUI에서는 OpenGL Performer의 구성을 파악하여야 MFC와의 연결을 할 수 있기 때문에 많은 어려움이 있었다. 장점으로는 다양한 데이터 파일 포맷을 편리하게 로딩할 수 있다는 것이었다.

OpenCASCADE는 MFC와의 연결을 할 때 최적화가 되어있다. OpenCASCADE 기반 GUI 시스템 자체에 MFC의 기본프레임을 구현시켜주는 애플리케이션 위저드가 있어 GUI 시스템 구현이 편리하였다. 그리고 사용자 필요로 하는 기능 또한 제공되어 사용하기 편리성이 있지만, 현재 개발되는 중이라는 점 때문에 시스템자체가 불안정하여 여러 에러 현상을 유발하는 것이 가장 큰 단점으로 파악되었다.

결과적으로 구현된 GUI 시스템들을 비교분석한 결과, MFC를 사용하여 GUI 시스템을 구현하기 편리한 그래픽 라이브러리는 OpenCASCADE이고, 가장 구현하기 힘든 그래픽 라이브러리는 OpenGL Performer다.

4. GUI 시스템 성능 테스트

각 그래픽 라이브러리를 기반으로 구현된 GUI 시스템의 성능을 분석하기 위해서 윈도우즈에서 제공하는 시간측정함수를 사용하였다. 윈도우즈 상에서 시간을 측정하기 위한 방법 중에는 몇 가지가 있는데 그중에서도 WM_TIMER 메시지나 멀티미디어 타이머를 많이 사용한다. WM_TIMER의 경우는 초당 클록 수가 18.3이므로 1/18.3 즉 55ms정도의 정확도를 가진다. 멀티미디어 타이머의 경우는 1ms 정도의 정확도를 가지지만, 그 이하의 시간을 측정하는 데는 적합하지 않다.

본 논문에서는 보다 높은 정확도를 확보하기 위하여 Win32 API에서 제공하는 QueryPerformanceFrequency와 QueryPerformanceCounter를 채택하였다. Eq.4.1은 본 연구에서 사용한 CPU 시간 측정식이다.

CPU Clock(ms) = (마지막 시간 - 처음 시간) / 주파수 (Eq.4.1)

본 연구에서는 구현된 각 GUI 시스템에 특정 모델 데이터를 입력하여 그 로딩시간을 QueryPerformanceFrequency와 QueryPerformanceCounter함수들을 이용해 측정하고 그 결과를 토대로 각 GUI 시스템의 파일포맷 처리 성능을 비교하였다.

Table 4.1은 성능 테스트를 실행 할 모델 데이터의 파일 포맷 형식을 나타낸다.

Table 4.1 File formats available in each GUI system

OpenGL	OpenGL Performer
VRML(*.wrl)	OpenFlight(*.flt)
3DS(*.3ds)	3DS(*.3ds)

OpenGL 기반 GUI 시스템의 경우 파일 포맷의 임포트와 익스포트를 제어하는 라이브러리가 별도로 포함되지 않기 때문에 공개버전인 VRML 파서 및 3ds 로더를 사용하여 wrl & 3ds 포맷을 채택하였다. OpenGL Performer 기반 GUI 시스템은 라이브러리에서 자체적으로 다양한 파일 포맷을 제공하지만 그 중 OpenGL Performer의 표준 파일 포맷으로 정의되어 있는 OpenFlight(.flt) 포맷과, 동등한 비교를 위하여 OpenGL에서 채택된 3ds 포맷을 채택하여 렌더링 성능을 비교 및 분석하였다. OpenCASCADE 기반 GUI 시스템에서는 데이터 상호간에 변환이 되지 않아 렌더링 성능 테스트를 실행하지 않았다.

4.1 테스팅 플랫폼 세팅

본 연구에서는 각 GUI 시스템의 성능 분석을 위해 nVidia 사의 Intel 775A 칩셋에 기반을 둔 ASUS마더보드를 기준으로 인텔 펜티엄 D 프로세서, 2GB RAM, nVidia사의 QuadroFX 4500 그래픽카드를 장착하였다(Table 4.2). 테스트를 위한 운영 체제로는 마이크로소프트사의 윈도우 XP 프로페셔널 버전 2002 서비스 팩 2를 채택하였고, 그래픽 서브-시스템의 드라이버는 nVidia의 쿼드로 그래픽 드라이버 110.38을 채택하였다.

Table 4.2 Specifications for test platform

Mother Board	ASUS Intel 775A Chipset
CPU	Intel Pentium D Processor
RAM	DDR 1024MB * 2
Operation System	Microsoft Windows XP Professional Version2002 Service Pack 2
Graphic Card	nVidia QuadroFX 4500 Chipset
Device Driver	nVidia Quadro Graphic Driver Version 110.38

4.2 GUI 시스템 렌더링 성능 테스트

OpenGL과 OpenGL Performer 기반 GUI system에서 렌더링 성능 테스트를 실시하였다. 테스트할 모델은 점과 면, 그리고 메시로 모델링된 13,000톤 탱커선박이다. 탱커의 초기 모델링은 Rhino3D에서 [8] 실시하였고 모델링 후 각 서브 모델별 .3dm 포맷으로 익스포트 한 후 Autodesk사의 3D MAX 8.0과 MultiGen Creator 2.5를 [9] 이용하여 채택한 파일포맷으로 각각 변환하였다. 모델의 원형요소정보에 대한 초기조건을 일치시키기 위해 기본 모드(standard mode)에서 파일포맷을 변환, 변환 후 각 포맷의 원형요소정보가 일치함을 확인하였다. 본 연구에서는 모델링에 있어서 가장 중요한 정점, 면 정보들을 원형요소로 지정한다.

각 GUI 시스템성능 테스트는 탱커의 7개 서브 모델을 대상으로 실시하였다. 각각의 각 서브모델은 고유한 정점과 면 개수를 가지고 있고 있다. Table 4.3은 탱커의 각 서브 모델별 정점 및 면의 수를 리스트하고 있다.

Table 4.3 Complexity of data model

	Number of vertices	*Number of faces*
Deck shell	47,865	99,054
Deck house	144,182	289,278
선체내부 I	198,427	305,186
선체내부 II	218,497	338,168
선체내부 III	223,458	347,162
선 체	6,279	5,596
탱커 전체	463,591	823,038

Fig.4.1부터 4.6까지 13K 탱커의 각 서브모델에 대한 와이어프레임을 도시하고 있고, Fig.4.7~8은 통합된 탱커의 와이어프레임과 음영 처리된 화면을 보여준다.

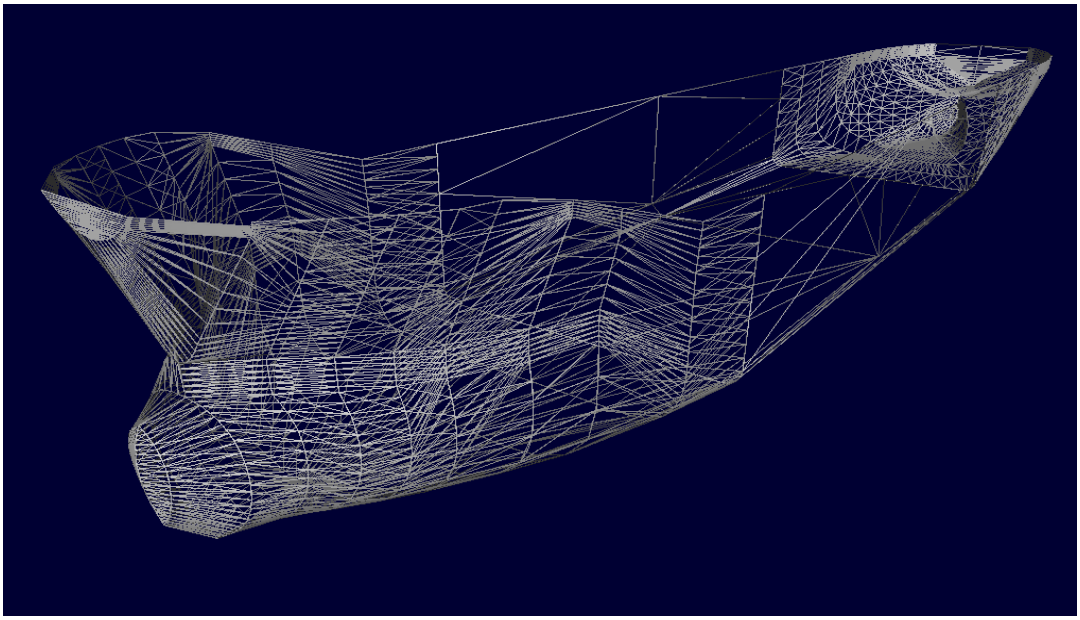


Fig.4.1 Wireframe model of hull

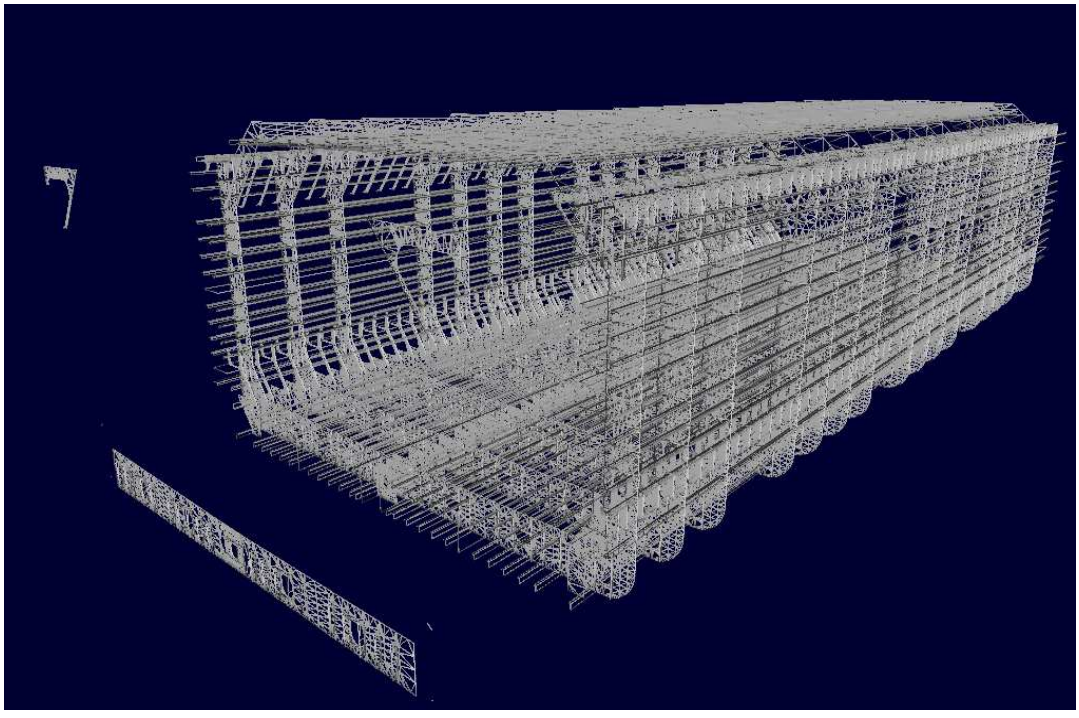


Fig.4.2 Wireframe model of internal structure I

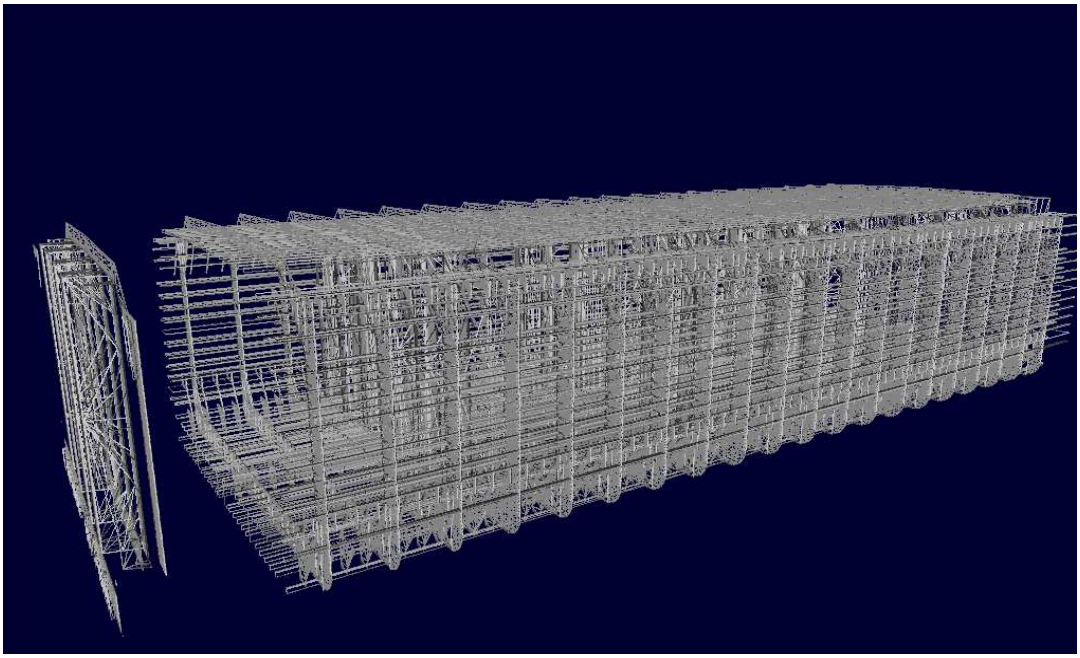


Fig.4.3 Wireframe model of internal structure II

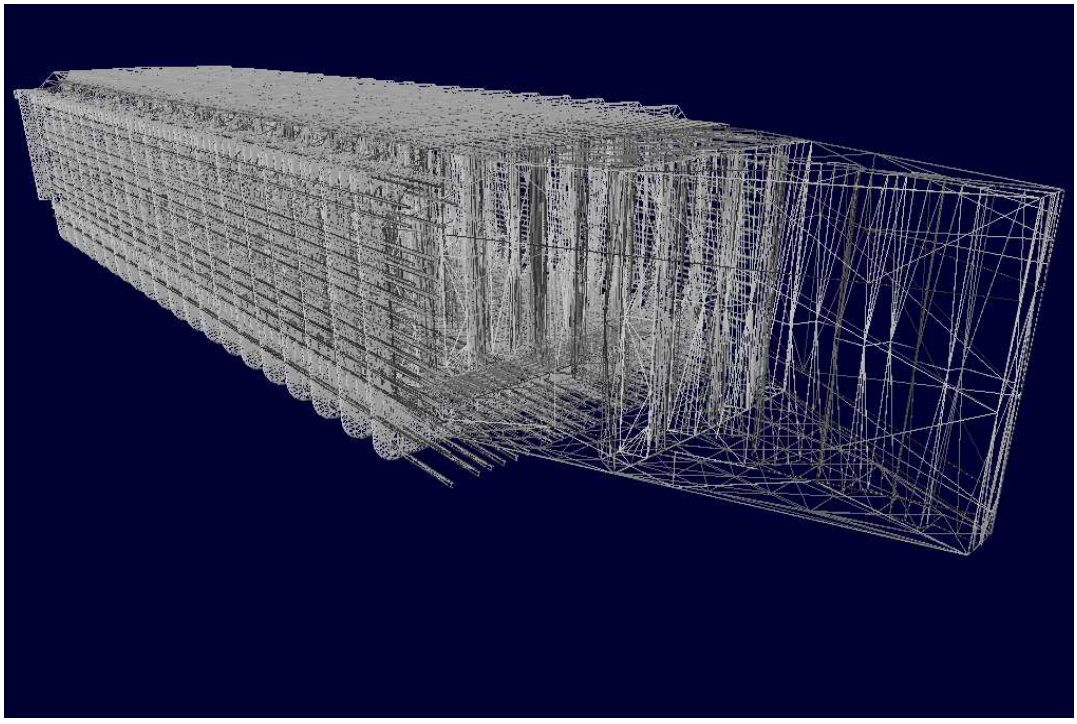


Fig.4.4 Wireframe model of internal structure III

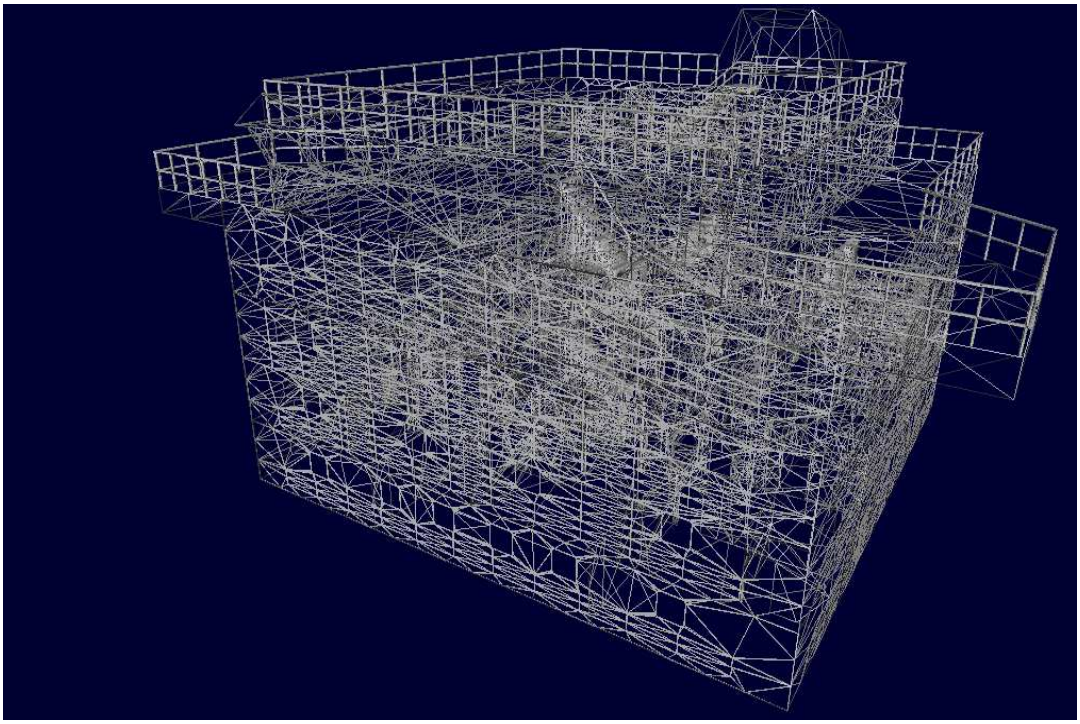


Fig.4.5 Wireframe model of deck shell

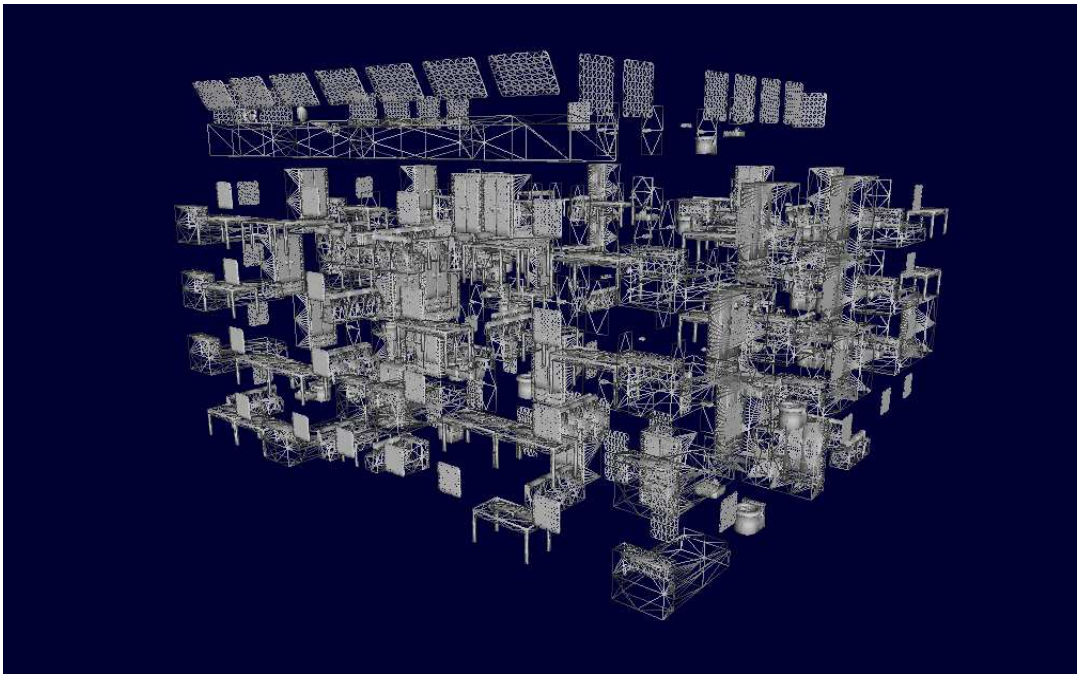


Fig.4.6 Wireframe model of deck house

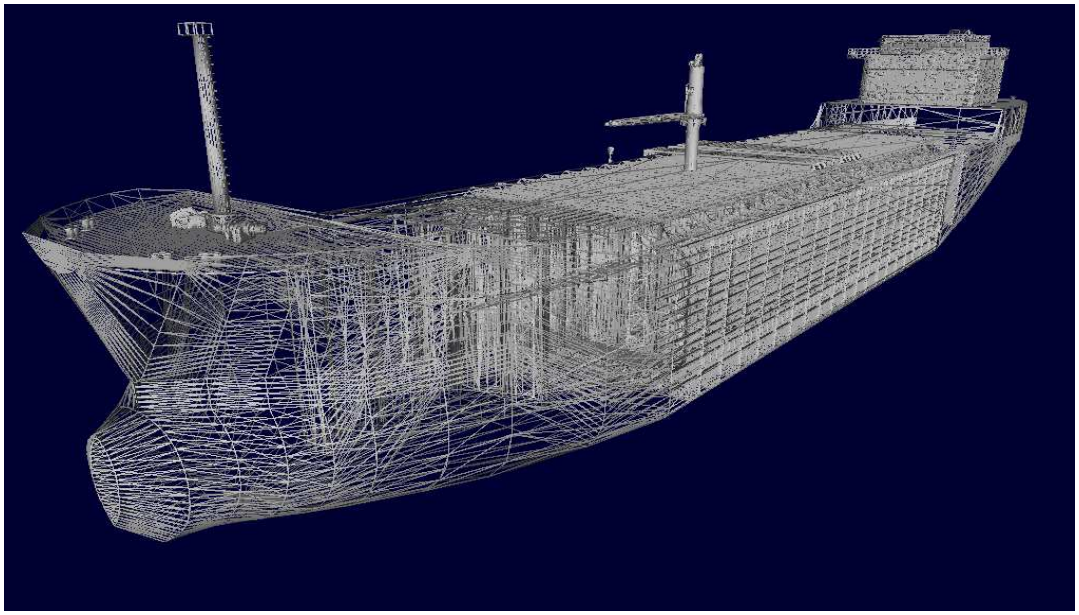


Fig.4.7 Wireframe model of integrated 13K tanker

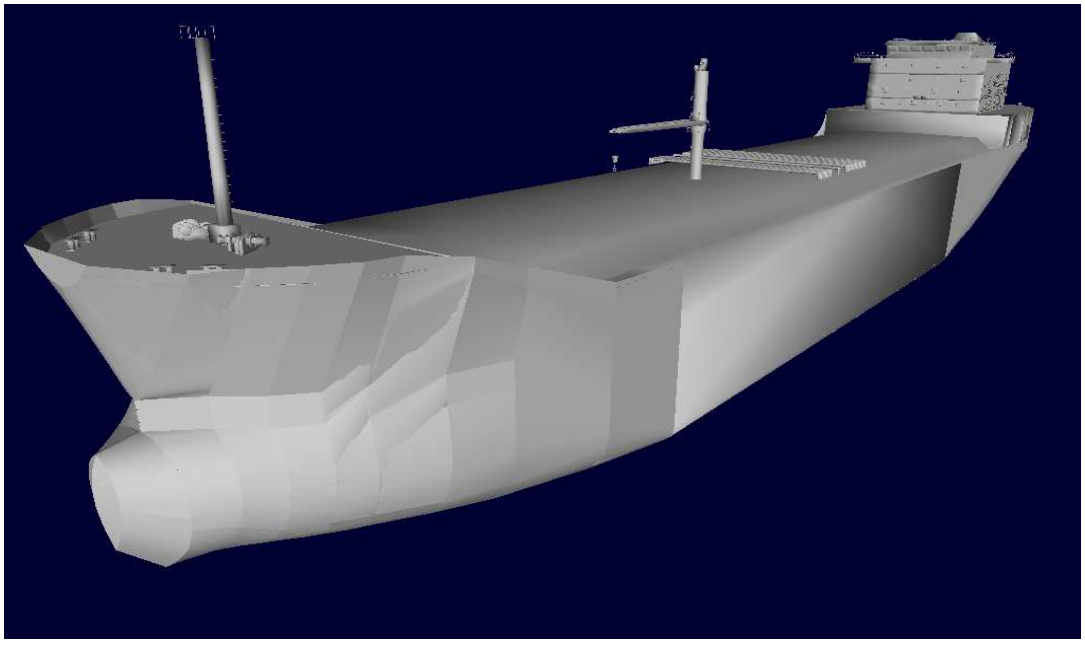
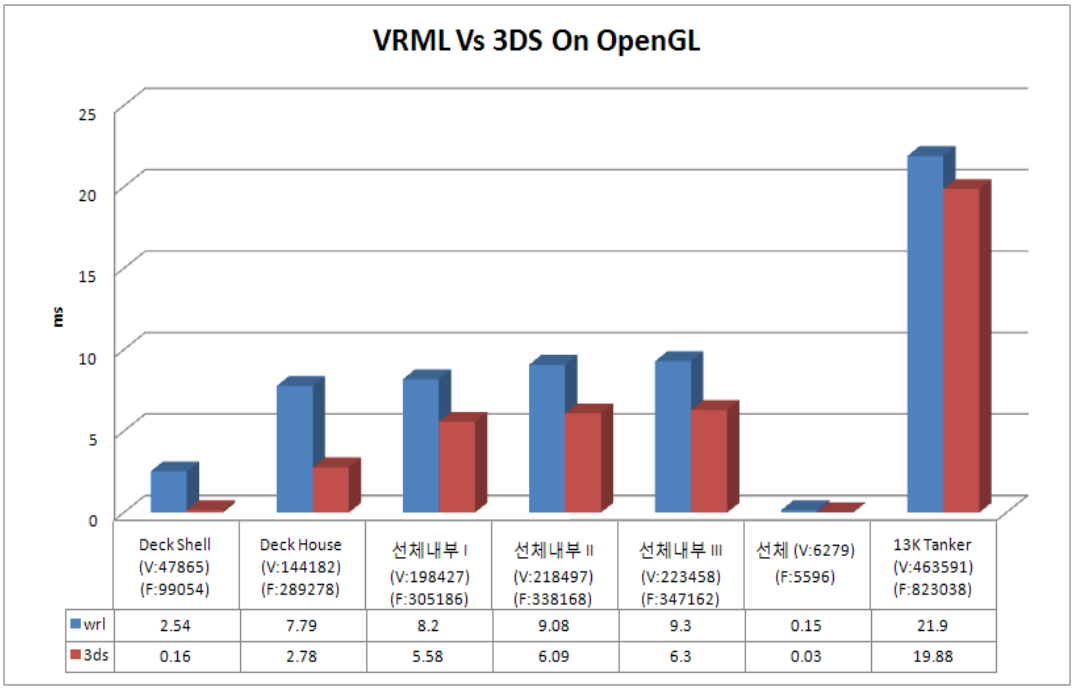


Fig.4.8 Shaded model of 13K tanker

4.2.1 OpenGL 기반 GUI 시스템의 렌더링 성능 테스트

OpenGL 기반 GUI 시스템의 렌더링 성능 테스트에서는 VRML 포맷과 3ds 포맷을 비교분석하였다.

Fig.4.9은 각 포맷에 대하여 각 서브모델의 렌더링 결과를 그래프로 나타내고 있다. 가로축은 각각의 파트별 모델을 나타내고 첫 번째 칸의 괄호 안은 V는 정점, F는 면을 나타내고 세로축은 각각의 모델들에 대한 렌더링 처리 시간(frames/ms)을 나타내고 있다. 그래프에서 보이는 바와 같이 정점과 면의 수가 많을수록 처리시간이 더 오래 걸린다는 것을 알 수 있다. 또한 OpenGL 기반 GUI에서는 VRML보다 3ds 포맷이 빨리 렌더링 되지만 이 두 파일포맷의 차이는 그리 크지 않다는 점을 알 수 있다.



	Deck Shell (V:47865) (F:99054)	Deck House (V:144182) (F:289278)	선체내부 I (V:198427) (F:305186)	선체내부 II (V:218497) (F:338168)	선체내부 III (V:223458) (F:347162)	선체 (V:6279) (F:5596)	13K Tanker (V:463591) (F:823038)
wrl	2.54	7.79	8.2	9.08	9.3	0.15	21.9
3ds	0.16	2.78	5.58	6.09	6.3	0.03	19.88

Fig.4.9 Comparison between VRML vs 3ds on OpenGL based GUI system

4.2.2 OpenGL Performer 기반 GUI 시스템의 렌더링 성능 테스트

OpenGL Performer 기반 GUI 시스템의 렌더링 성능 테스트에서는 flt 포맷과 3ds 포맷을 비교분석하였다.

Fig.4.10은 flt 포맷과 3ds 포맷의 테스트 결과를 그래프로 나타내고 있다. OpenGL Performer 기반 GUI 시스템에서도 정점과 면의 수가 많을수록 렌더링 처리시간이 오래 걸리지만 OpenGL 기반 GUI 시스템에 비하여 그 의존도가 덜 하다는 것을 알 수 있다. 또한 flt 포맷보다도 3ds 포맷에서 월등히 빠르게 렌더링 되는 것을 알 수 있다.

결과적으로 OpenGL Performer 기반 GUI 시스템에서 정점과 면수에 따라 렌더링 처리 시간이 달라지며 3ds 파일 포맷을 이용하면 렌더링 속도가 훨씬 개선된다는 것을 알 수 있었다.

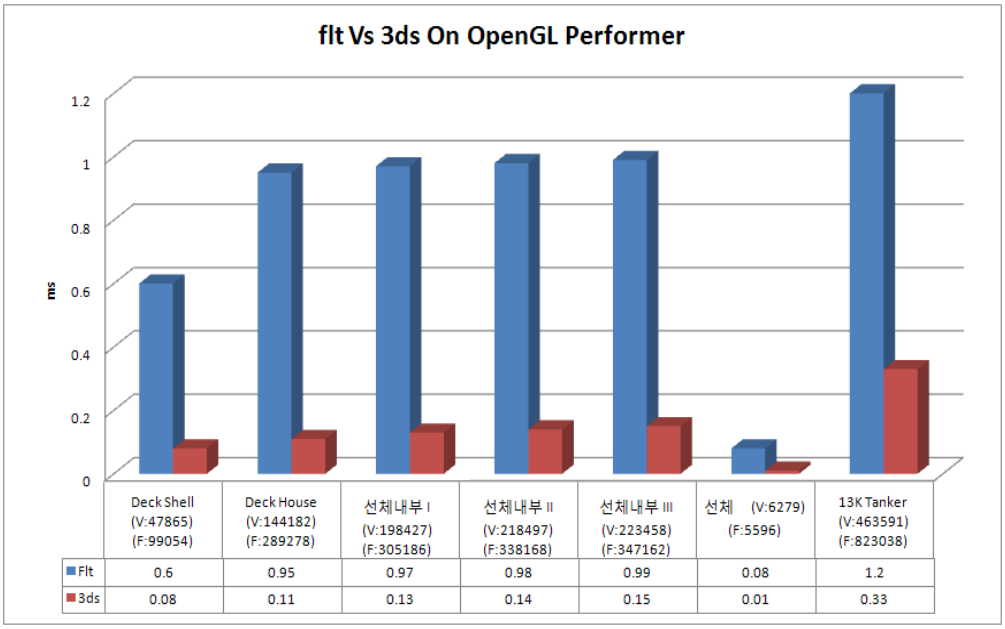
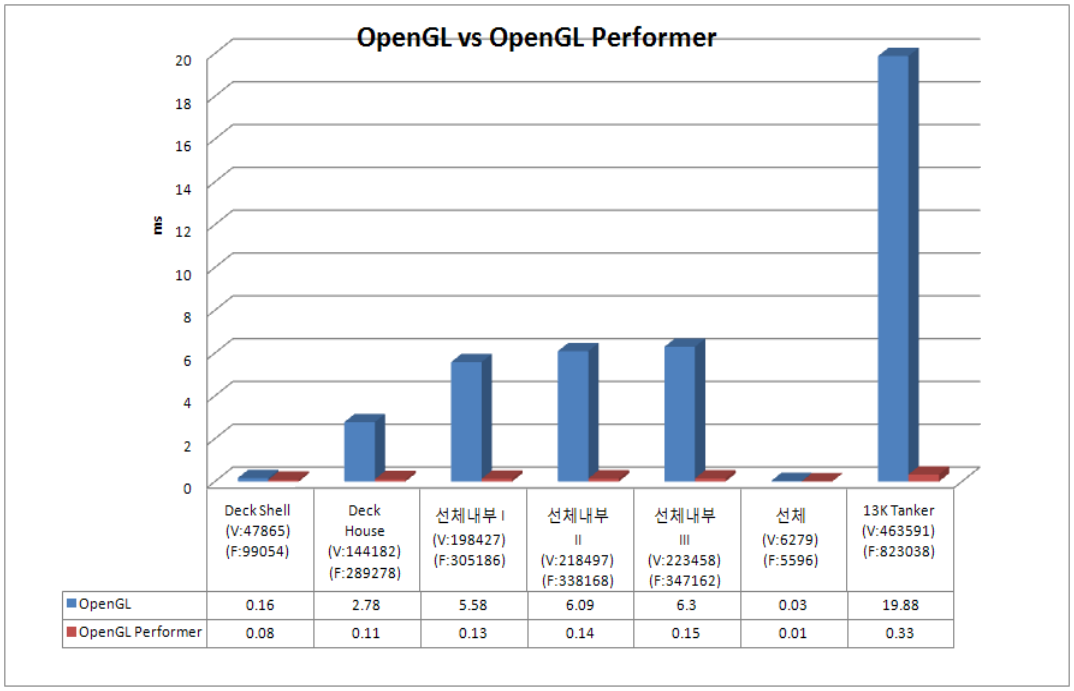


Fig.4.10 Comparison between flt vs 3ds on OpenGL Performer based GUI system

4.2.3 OpenGL 기반 *GUI* 시스템과 *OpenGL Performer* 기반 *GUI* 시스템의 렌더링 성능 비교

OpenGL 기반 GUI 시스템과 OpenGL Performer 기반 GUI 시스템간의 렌더링 성능 테스트는 직접적인 성능비교를 위해 동일한 파일 포맷인 3ds 포맷을 이용하였다. Fig.4.11은 비교 테스트 결과를 도시하고 있다. 그래프에서 보이는 바와 같이 OpenGL Performer 기반 GUI 시스템이 OpenGL 기반 GUI 시스템보다 렌더링 속도가 월등히 빠르다는 것을 알 수 있다. OpenGL 라이브러리는 OpenGL Performer 보다 낮은 레벨의 라이브러리이다. 그런데도 OpenGL Performer가 OpenGL보다 렌더링 시간이 빠르다는 것은, 그만큼 OpenGL Performer 라이브러리가 렌더링 속도에 있어 최적화 되어있다는 것을 의미한다는 것이고 본 테스트를 통하여 그 사실이 검증되었다.



	Deck Shell (V:47865) (F:99054)	Deck House (V:144182) (F:289278)	선체내부 I (V:198427) (F:305186)	선체내부 II (V:218497) (F:338168)	선체내부 III (V:223458) (F:347162)	선체 (V:6279) (F:5596)	13K Tanker (V:463591) (F:823038)
OpenGL	0.16	2.78	5.58	6.09	6.3	0.03	19.88
OpenGL Performer	0.08	0.11	0.13	0.14	0.15	0.01	0.33

Fig.4.11 Comparison between OpenGL vs OpenGL Performer based GUI systems

본 연구에서는 OpenGL 기반 GUI 시스템과 OpenGL Performer 기반 GUI 시스템간의 모든 성능을 알 수 없었지만 렌더링 성능 테스트를 통해 정점과 면의 개수가 렌더링에 끼치는 영향 및 의존도를 짐작할 수 있었고, 렌더링에 보다 최적화된 라이브러리를 확인할 수 있었다.

5. 결론

3차원 그래픽 API를 구현할 때 어떤 그래픽 라이브러리와 파일 포맷을 선택할지에 대한 결정은 GUI 관련 프로젝트를 수행함에 있어서 매우 중요한 요소로 작용한다. 본 논문에서는 선박 가시화 등과 같이 전산응용 개발환경에 적합한 GUI 시스템개발을 위한 그래픽 라이브러리들의 구성과 설정, GUI 시스템과 MFC 연결, 시스템의 성능 테스트 등에 대한 연구를 수행하였다. 연구결과를 활용하여 사용자가 모델링과 시뮬레이션을 위한 3차원 그래픽 API를 구현할 때 어떤 그래픽 라이브러리와 데이터 파일 포맷을 선택을 위한 지침서를 제공하고자 한다.

그래픽 라이브러리를 활용한 GUI 시스템 개발을 위하여 OpenGL, OpenCASCADE, 그리고 OpenGL Performer 라이브러리를 채택하고 MFC를 결합하여 유용한 GUI 시스템들을 구현하였다. 각 GUI 시스템의 활용도를 분석한 결과, CAD시스템의 기능을 요구하는 종합적인 가시화 시스템을 구현하고자 한다면 OpenCASCADE 라이브러리가 가장 적절하고, 3차원 그래픽을 더욱 세부적으로 표현하고 다양한 효과를 구현하려면 OpenGL 라이브러리가 최적이며, 모델을 가시화하고 가상현실 및 시뮬레이션 응용과 같이 빠른 렌더링 속도가 요구 될 때에는 OpenGL Performer 라이브러리가 가장 최적이라고 결론을 얻었다.

각 라이브러리에 따른 렌더링 효율과 데이터 파일포맷과의 호환성을 보다 실질적으로 검증해보기 위하여, 구현된 OpenGL과 OpenGL Performer GUI 시스템에 대한 테스트를 수행하였다. 선정된 특정 모델의 원형 인자인 정점과 면만을 가진 7가지 서브모델들을 여러 포맷형식으로 변환한 후, 각 시스템에서 렌더링 처리 시간을 측정하였다. 측정결과를 비교 및 분석해본 결과 단연 OpenGL Performer 기반 GUI 시스템이 렌더링 속도 부문에서 뛰어남을 알 수 있었다. 이 테스트를 통해 OpenGL Performer 기반 GUI 시스템이 렌더링과 비주얼 관련 처리속도 부문에 있어서 가장 최적화가 잘 되어 있다는 사실을 검증하였고, 나아가 모델링과 시뮬레이션 분야에서 OpenGL Performer의 활용도가 높다는 사실을 다시 한 번 확인할 수 있었다. 그리고 다양한 모델링 파일 포맷 중에서도 3DMax기반의 3ds 포맷이 모든 라이브러리들과 최적의 호환성을 가진다는 사실 또한 알 수 있었던 계기가 되었다.

참고문헌

[1] Dave Shreiner외 3명 공저 / 남기혁 역, OpenGL 프로그래밍 가이드 제4판, 정보문화사, 2005

[2] Richard S외 2명 공저 / 최현호 역, OpenGL Super Bible 제3판, 정보문화사, 2005

[3] “www.OpenCASCADE.org”, OpenCASCADE Technology Main Page, 2001

[4] http://www.sgi.com/products/software/performer/manuals.html

[5] 김용성 저, VISUAL C++ 6 완벽 가이드, (주)영진출판사, 2003

[6] "nehe.gamedev.com", Nehe Productions Main Page, 1997

[7] 최상수, 신동목, “OpenCASCADE를 이용한 블록조립 계획용 CAD 인터페이스”, 한국 해양공학학회지 제18권 제3호, pp 26 ~ 31, 2004

[8] Robert McNeel & Associates, Rhinoceros NURBS Modeling For Windows 사용자 가이드, 1993

[9] "http://www.Multigen-Paradigm.com", Presagis Main Page, 2007