工學碩士 學位論文

UX를 고려한
3차원 그래픽 프로그램의
GUI 개발 도구 설계 및 구현

2014年 8月

釜慶大學校 大學院

IT 融合應用工學科

全 俊 泳

工學碩士 學位論文

UX를 고려한 3차원 그래픽 프로그램의 GUI 개발 도구 설계 및 구현

指導敎授 金 榮 鳳

이 論文을 工學碩士 學位論文으로 提出함.

2014年 8月

釜慶大學校 大學院

IT 融合應用工學科

全 俊 泳

全俊泳의 工學碩士 學位論文을 認准함

2014年 6月 30日

主 審　　工學博士　辛 奉 烋 (印)

委 員　　工學博士　金 榮 鳳 (印)

委 員　　工學博士　宋 河 周 (印)

목 차

표 목 차

그 림 목 차

Design and Implementation of a UX-based GUI development tool for 3D graphics programming

Jun-Young Jeon

Dept. of IT Convergence and Application Engineering, Graduate School
Pukyong National University

Abstract

Recently, three dimensional graphics technologies are widely required in the fields of 3D games, 3D simulations to test the movement of general mechanical products, training to control the very expensive equipment, and so son. These three dimensional systems have not given the user interface that is in harmony with three dimensional displays. Many researchers have tried to develop the user interface which shows the adequate harmony in three dimensional programming environments. However, since users have directly drawn each graphical user interfaces on the three dimensional rendering region, it is difficult to implement the message processing and the event detection. It makes the big jump in the developing cost and also the processing time.

Therefore, in order to solve this problem, we propose a GUI development tool to design the user interface which can be

intuitive expression with three dimensional. We implement a GUI in the course of development of three dimensional graphics programs. We also enhance the portability and developing speed through the usage of view-handler design patterns. We separate the rendering part and the event data processing part and thus make the multi-processing possible. Non-synchronization employing triple buffers gives the rapid responds time. Through the employment of our developing tools, we can develop the natural user interface that gives a harmony with the application and three dimensional simulators. The user interface is possible to adopt the various graphical effects and also give a natural interaction with users.

1. 서 론

1.1. 연구배경

고사양 하드웨어의 개발과 초고속 인터넷의 유무선 서비스가 가능해지고, 특히 애플의 아이폰이 출시된 이후로 스마트 폰과 같은 소형의 기기들이 각광을 받고 있다. 아이폰이나 안드로이드 폰과 같은 소형기기들의 보급이 일반화 되면서 휴대전화의 기능뿐만 아니라 일정 관리, 네비게이션, 게임 등의 일상생활에 활용 가능한 다양한 어플리케이션들이 일반인들의 주요 관심 사항이 되고 있다[1].

기존의 일반적인 어플리케이션에서는 그림 1.1 과 같은 WinAPI 및 MFC 또는 AWT 및 Swing 같은 윈도우 컨트롤 기반의 어플리케이션을 사용하였다[2]. 하지만 유저들의 눈높이가 높아짐에 따라 다양한 그래픽 및 화려한 퍼포먼스를 요구하게 되어 그림 1.2 와 같은 게임, 시뮬레이션 같은 기존의 3차원 처리 기술이 필요한 어플리케이션뿐만이 아니라 일반 어플리케이션에서도 3차원 처리 기술을 필요로 하고 있다. 이에 따라 다양한 3차원 그래픽 프로그램 환경에서의 사용자 인터페이스(User Interface, UI)의 중요성 또한 커지고 있다. 또 유저의

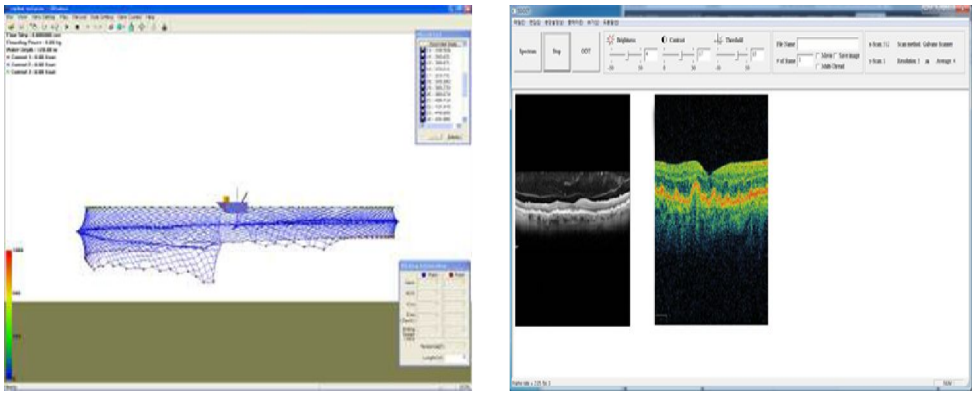

그림 1.1 기존 GUI 형식의 3차원 어로 시뮬레이터와 광학 OCT장비

편이성을 위한 사용자 경험(User Experience, UX)을 고려한 그래픽 유저 인터페이스(Graphic User Interface, GUI), 3차원 GUI 등의 다양한 개발 관련 요구도 커지고 있다.

그림 1.2 3차원 어플리케이션에서의 UI

1.2. 연구 목적

기존에 볼 수 있었던 일반 3차원 어플리케이션은 3차원 렌더링 영역을 따로 두고 3차원 렌더링 바깥에 Window기반의 UI를 사용하는 형식의 프로그램이 일반화 되어있다. 이에 따라 UI부와 3차원 렌더링부의 DC충돌로 인한 격자 현상이 발생하고, 사용자와의 상호작용 화면은 2차원과 3차원의 결합으로 어색한 모습을 보이고 있다. 또한 정형화된 윈도우 컨트롤의 GUI에서는 다양한 그래픽 효과를 기대하기도 어렵다. 따라서 게임처럼 GUI가 렌더링 영역 내에 프로그램의 일부처럼 구현하는 요구가 높아지고 있지만, 3차원 렌더링부에 GUI를 일일이 직접 그려주어야 하고, 이벤트 감지 및 메시지 처리 또한 직접 구현해야 하는 개발 비용 및 시간이 늘어나게 되는 문제점을 가지고 있다[3].

본 논문에서는 위의 문제를 해결하기 위한 GUI 개발도구를 설계 및

구현한다. 먼저 GUI가 렌더링 영역 내에 프로그램의 일부처럼 보이도록 일반적인 3차원 그래픽 프로그램 개발의 구조 안에서 GUI를 구현하고, 이를 통해 쉐이더(Shader) 및 3차원 모델을 사용하는 등의 다양한 그래픽 효과 적용이 가능하도록 한다. 또 GUI를 일일이 그리거나, 이벤트 감지 및 메시지 처리를 매번 구현 하지 않도록 뷰-핸들러(View-Handler) 디자인 패턴 구조를 적용해 개발의 용이성과 이식성을 높이도록 한다. 여기에 유저의 편이성을 높이기 위해, 개발의 다양한 구조가 적용가능 하도록 설계하여, UX 디자인을 고려한 GUI 구현이 가능하도록 한다.

마지막으로 GUI 렌더링 부과 데이터 프로세스 처리부를 각각 독립적 구조로 구성해 멀티 프로세싱하고, 트리플 버퍼를 사용하여 비동기화 하여 빠른 응답속도를 보장하는 GUI 개발 도구를 구현하고자 한다.

1.3. 논문 구성

본 논문의 구성은 다음과 같다. 서론에 이어 2장에서는 본 논문에서 설계 및 구현하려는 UX를 고려한 3차원 그래픽 프로그램의 GUI 개발 도구에 필요한 기초 정의 및 관련 연구에 대해 기술한다. 3장에서는 본 논문에서 제안한 구조적 접근과, 뷰-핸들러 디자인 패턴, 마지막으로 트리플 버퍼에 대해 기술한다. 4장에서는 제안한 개발도구 구현 방법과 각각 컨트롤들의 기능과 개발도구 사용방법을 알아본다. 5장에서는 구현 결과를 살펴보고, 마지막으로 6장에서 결론과 향후 연구에 대해 기술한다.

2. 관련 연구

2장에서는 본 논문에서 설계 및 구현하려는 UX를 고려한 3차원 그래픽 프로그램의 GUI 개발도구에 필요한 기초 정의 및 관련 연구에 대해 기술한다. 먼저 UI의 정의, UX 디자인, 마지막으로 3D 유저 인터페이스에 대해 알아본다.

2.1. 유저 인터페이스(User Interface, UI)

원래 인터페이스(Interface)는 ‘ 이질적인 두 가지 물질이 접촉 한다’ 는 의미의 화학용어에서 유래 되었다. 일반적으로 말해 두 종류의 서로 다른 세계가 서로 접촉해서 발생되는 의사소통의 장소를 의미한다. 예를 들면 손잡이는 사람과 문 사이의 인터페이스이고, 자동차의 핸들, 엑셀레이터, 클러치 등은 운전자와 자동차간의 인터페이스라 할 수 있다.

컴퓨터 관련 용어에서는 인터페이스는 하드웨어나 소프트웨어 같은 컴퓨터와 인간과의 커뮤니케이션으로 이해되고 있다. 즉, 1차적으로 사람과 도구, 기계, 컴퓨터, 시스템 등의 접점을 의미하고, 2차적으로는 사용자와 각각의 시스템 사이의 정보 채널로 이해 할 수 있다.

초보적인 의미의 인터페이스는 화면이나 문서 자체가 중요한 핵심을 가진다. 그러한 개념이 발전하여 두 개의 객체가 상호 작용하는 곳에서 사용자는 인터페이스를 자연스럽게 접하게 된 것을 ‘ 유저 인터페이스(User Interface, UI)’ 라고 한다. 우리는 어떤 새로운 장비나 제품 혹은 프로그램 등을 처음 대하게 될 때 어떻게 사용해야 할지 몰라

당황하는 경우가 종종 있다. 이를 이용해보려고 추측도 하고, 기존의 다른 경우에 작동해 보았던 경험을 살려 시도해 보기도 한다. 다행히 그러한 시도가 맞아 떨어지면 다음 단계로 수월히 넘어 갈 수 있지만, 그렇지 못하면 많은 시간을 낭비하거나, 심한 경우에는 엉뚱한 결과를 초래하기도 한 다. UI는 이러한 이유 때문에 사용자와 사물간의 상호 작용을 돕기 위해 탄생한 도구이다. 이는 또 사용자와의 상호 교감을 의미한다. UI를 하는 목적은 사용자의 시행착오를 줄여주기 위해 사용자가 쉽게 인식하고 사용하기 편리 하게 상호 작용 요소를 제공하려는 데 있다[4].

그래픽 유저 인터페이스(Gaphic User Interface, GUI)는 작업을 달성하는데 필요한 시각정보를 그래픽적으로 사용자에게 제공하는 인터페이스를 말한다. 즉 디스플레이 그래픽을 이용한 컴퓨터 소프트웨어의 인터페이스 사용자 대화방식이다. 함축적으로 말하자면 '그래픽 유저 접속구조'이다. 1990년대 이후 대부분의 소프트웨어 인터페이스는 GUI의 범주에 해당할 만큼 그 중요성이 커지고 있다. 또 최근에는 3차원 GUI 까지 다양한 GUI가 필요로 하고 있다. 일반적으로 GUI는 CRT표시와 같이 제한 되어있는 공간에서의 정보 전달을 목적으로 한다. 예전의 휴먼 컴퓨터 인터페이스(Human Computer Interface)는 기존의 MS-DOS와 같이 대부분 문자로 명령어를 입력하여 컴퓨터에게 명령을 내리는 방법으로 진행됐었다. 그러나 GUI는 화면에 미리 미리 배치하여 사용자로 하여금 복잡한 절차 없이 직접 컴퓨터와 커뮤니케이션을 할 수 있도록 해준다. 사회의 각종 시스템이 컴퓨터화 되고 소형 컴퓨터가 제품에 적용되면서 모바일 폰, 전자수첩, 현금출납기 등의 LCD 패널 디스플레이를 통해 GUI의 다양한 형태가 응용되고 있다[5].

2.2. 사용자 경험(User Experience, UX) 디자인

경험이란 사람이 실지로 보고, 듣고, 겪는 일이나 그 과정을 통해 얻어지는 지식이나 기능, 느낌으로 그 대상에 구애됨 없이 오래전부터 사용된 용어이다. 경험은 여러 학자에 의해 다양한 분야에서 다양한 의미로 연구되어 오고 있으나, 대상과의 접촉을 통해 이루어진다는 기본적인 개념은 동일하게 적용되어 오고 있다. 경험이 그 대상에 구애됨 없이 오래전부터 사용된 용어이듯이, 사용자 경험의 개념도 모든 객체들에 적용되어진다. 다시 말해 사용자가 상품을 만지거나, 혹은 사운드를 듣고, 코로 냄새를 맡는 등의 모든 감각적 경험을 포함하며, 디지털 인터페이스의 조작을 통한 인터랙션이나 사람 사이의 커뮤니케이션까지도 모두 아우른다고 할 수 있다. 여러 학자들의 견해를 살펴보면 사용자 경험은 아직 확정된 개념으로 자리 한 것은 아니나 상호 작용이 중요한 요인이 되고 있으며, 특히 시스템과 관련지어 사용자 경험에 대해 설명하고 있음을 알 수 있다[6].

UX란 단순한 경험에서 나오는 요소를 넘어선 많은 사용자의 감성요소와 제품과 상호작용할 수있는 인터랙션적인 요소가 포함되어 있다. UX의 향상 요소들로는 그림 3과 같이 상호작용 방식, 정보 제공 방식, 효과성, 효율성, 만족성을 포괄하는 사용성, 그리고 인간공학적 요소 등이 포함되어 있다[7].

또 사용자 경험 디자인은 경험이라는 추상적인 인간의 감성을 자극할 수 있는 요소가 시각적이고 감성적 요소인 디자인과 결합된 것으로, 사용자 목적 달성의 과정을 구현하게 되는 것이다. 오늘날 디자이너가 시각적 요소를 디자인하는데 있어서 사용자 경험은 디자인의 승

그림 2.1 사용자 경험 디자인

패를 좌우할 만큼 중요한 요소로 자리 잡혀 있다. 경험 디자인의 사전적 의미는 '사용자가 제품, 서비스 혹은 시스템을 사용하거나 체험하는데 있어 지각하는 것이 가능한 조직적 상호교감적인 모델을 창조하고 개발하는 디자인의 한 분야' 라고 정의될 수 있다. 또 최근에는 사용자 경험 디자인을 그림 2.1 과 같이 기술과, 고객의 요구사항, 비즈니스 목표의 결합 형태로도 분리하기도 한다. 하지만 경험이란 실체가 없기 때문에 경험 자체 객관적으로 디자인이 불가능하며, 경험의 주체가 개인이고, 스스로 간직하고 있는 것이기 때문에 객관적인 잣대로 개개인의 경험한 내용 전부를 있는 그대로 디자인할 수는 없다. 경험 자체를 디자인할 수 없지만 인터페이스, 환경, 시나리오와 같이 경험이 발생하는 상황(서비스)나 상황을 바꿀 수 있는 제품이나 기능 등 사용자 경험과 관련된 디자인 가능한 요소는 디자인할 수 있다. 이와 같이 '디자인 가능한 요소' 로 놓고 볼 때 경험디자인은 사용자의 피드백을 '경험' 이라는 시각으로 이해함으로써 그들의 요구를 보다 실질적으로 접근, 새롭고 개선된 경험을 제공하기 위한 디자인이라 할 수 있다. 디자인 가능한 요소라 함은 제품의 기능, 형태, 사회적 의미와 기

호, 정보, 유용성 그리고 외관을 구성하는 시각적 요소 등을 예로 들 수 있다. 이를 통하여 사용자들은 컨텐츠를 편리하게 사용할 수 있는 사용성이 향상되고, 이에 따라 컨텐츠에 대한 사용자들의 만족도 상승과 컨텐츠 가치 향상으로 이어진다[8].

2.3. 3D 유저 인터페이스 (3D User Interface)

3D 유저 인터페이스는 그림 2.2와 같은 일차원적으로 3D로 구현된 UI를 말하기도 하고, 3D가 도입되면서 각광받기 시작한 가상현실(virtual reality) 및 게임 내에서 캐릭터 및 유저가 직관적으로 활용 가능한 UI를 말한다. 2D 평면인 모니터 상에서 대상 물체의 보이지 않는 부분 및 숨겨진 부분을 사용자의 능동적인 공간 구조 변화에 의하여 화면의 전환 없이 즉시적으로 확인이 가능한 UI를 말한다. 젤저(Zeltzer)는 가상현실의 특징을 자율성(Autonomy), 상호작용성(Interaction), 그리고 현재성(Presence)으로 구분하였다[9]. 여기서 자율성이란 현실과 같은 물리적 속성을 가상현실의 요소가 가지고 실제와 같이 움직일 수 있는 행위의 실제성을 의미한다. 2D와 3D의 인터페이스의 가장 큰 차이점 중 하나는 자율성에서 찾을 수 있는데 그것은 인터페이스의 3차원 회전(rotation)과 Z축의 깊이감(Depth)을 통해 정보의 탐색구조와 체계화를 가능하게 한다는 점이다. 다양한 시점에서 자유로운 3차원 공간 내 상황의 확인권한을 부여하여 기존 2D 전략시뮬레이션의 영역을 확대시켰다. 가상현실에서의 상호작용성(interaction)이란 가상환경을 다루고자 하는 사용자의 모든 행위를 의미하며 이는 탐색(Navigation), 이동(Tracking), 제어(Control), 피드백(Feedback)으로 나눌 수 있다[10]. 2D 인터페이스와 3D 인터페

이스의 상호작용성의 차이는 대상 행동 조작의 차이이다. 2차원 인터페이스의 행동영역, 즉 상하좌우의 메뉴이동에서 3차원의 개념, 즉 Z축을 통한 제 3행동인 깊이의 이동 및 회전 등의 행동이 가능해 진다. 따라서 Z축을 이용한 새로운 데이터 인지력은 활용하기에 따라 무한한 능력을 지닐 수 있을 것이다[11].

그림 2.2 일반 어플리케이션과 게임에서의 3D UI

3. 제안한 접근법

본 장에서는 UX를 고려한 3차원 그래픽 프로그램의 GUI 개발도구를 설계 및 구현에 있어 다음과 같은 접근을 통한 방법을 알아본다. 먼저 구조적 접근과, 두 번째로 뷰-핸들러 디자인 패턴, 마지막으로 트리플 버퍼에 대해 기술한다.

3.1. 구조적 접근

기존에 볼 수 있었던 일반 3차원 어플리케이션은 3차원 렌더링 영역을 따로 두고 3차원 렌더링 바깥에 MFC, WinAPI, AWT, Swing과 같은 Window기반의 GUI를 사용하는 형식의 프로그램이 일반화 되어 있다. 이에 따라 그림 3.1 과 같은 GUI부와 3차원 렌더링부의 DC충돌로 인해 다른 DC가 그린 부분에 대해 화면에 남아있거나, 렌더링되지 않는 격자 현상이 발생하고, 사용자와의 상호작용 화면은 2차원과 3차원의 결합으로 어색한 모습을 보이고 있다. 또한 정형화된 윈도우

그림 3.1 3차원 게임에서의 DC충돌로 인한 격자현상

컨트롤의 GUI에서는 다양한 그래픽 효과를 기대하기도 어렵다.

이를 해결하기 위해 구조적으로, 그림 3.2와 같이 3차원 그래픽 프로그램 개발의 구조 내에서 GUI부를 독립적으로 구성해 따로 구현한다. 이를 통해 GUI가 렌더링 영역 내에 프로그램의 일부처럼 보이고, Shader를 활용한 다양한 애니메이션과 3차원 모델을 사용하는 등의 다양한 그래픽 효과 적용이 가능하다.

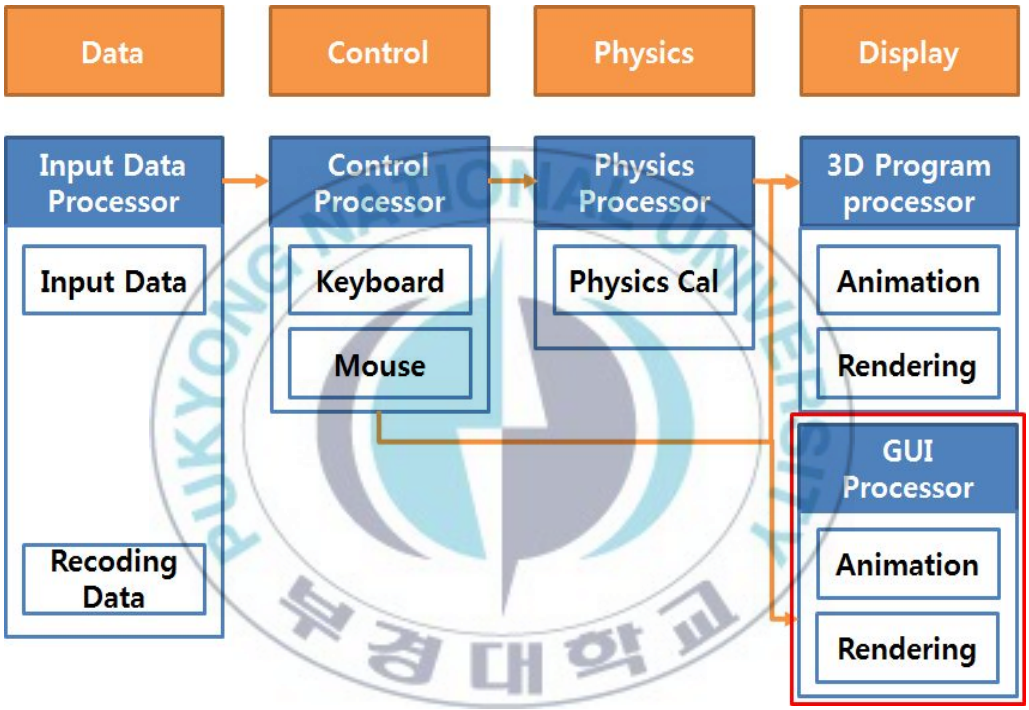

그림 3.2 일반적 3차원 그래픽 프로그램의 구조내의 GUI

3.2. 뷰-핸들러(View-Handler) 디자인 패턴

3.1절의 3차원 그래픽 프로그램 개발의 구조 내에서 GUI를 구현하기 위해선 매번 일일이 각각의 컨트롤들을 그리거나, 이벤트 감지 및 메시지 처리를 매번 구현 해야만 한다. 이런 문제로 인해 개발 비용 및 개발 시간이 발생해 어떤 일정한 구조 및 개발 도구를 필요로 한

다. 이에 본 논문에서는 뷰-핸들러 디자인 패턴 구조를 적용해 개발의 용이성과 이식성을 높이도록 한다. 또 여기에 다양한 구조가 적용가능하도록 설계해 UX디자인을 고려하여 유저의 편이성을 높이고자 한다.

뷰-핸들러 디자인 패턴의 구조는 다음 그림 3.3 와 같이 뷰-핸들러(View-Handler), 추상뷰(Abstract View), 상세뷰(Specific View), 보조(Supplier)로 구성된다. 각각의 기능을 살펴보면 뷰-핸들러는 응용 소프트웨어 시스템의 뷰들을 열고 조작하고 닫는 역할 등의 모든 뷰를 관리하는 일을 한다. 추상뷰는 모든 뷰에 공통적인 인터페이스를 정의한 컴포넌트이며, 상세뷰는 추상 뷰로 부터 파생된 것으로 추상뷰의 인터페이스를 실제로 구현한 컴포넌트이다. 보조는 데이터가 필요하면 데이터를 조작하여 뷰가 표시할 데이터를 제공하는 역할을 한다 [12].

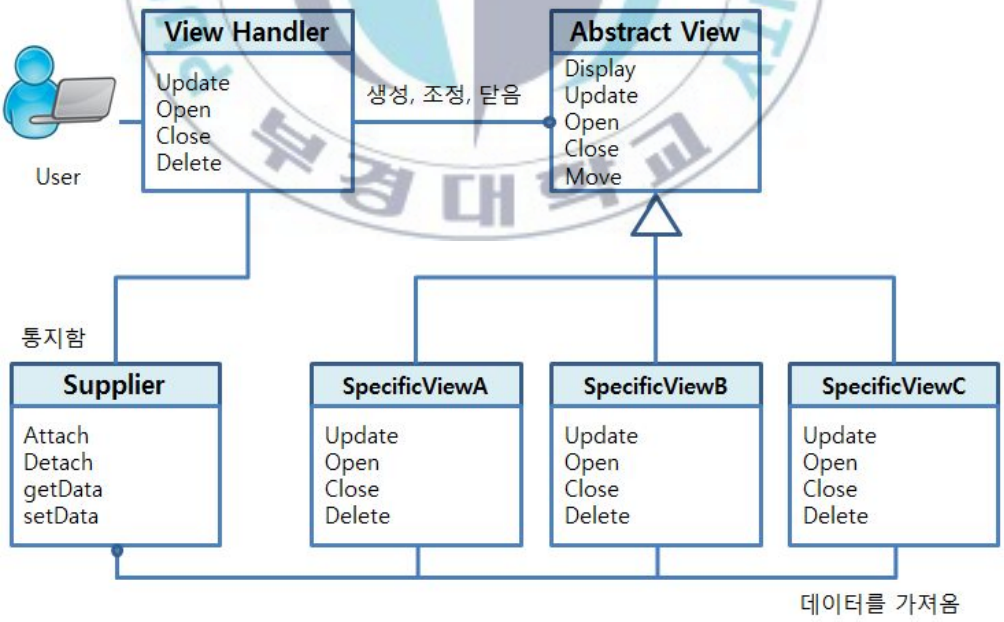

그림 3.3 뷰-핸들러 디자인 패턴 구조

뷰-핸들러 디자인 패턴의 구현은 다음과 같은 4단계로 이루어잔다.

1. 필요한 뷰를 모두 정의한다.
2. 모든 뷰들에 대한 공통 인터페이스를 정의한다.
3. 각각의 뷰를 구현한다.
4. 뷰-핸들러를 구현한다.

아래 그림 3.4는 뷰-핸들러 패턴의 다중 창을 여는 프로그램의 동작의 예이다. 사용자가 새 창을 열기위해 뷰를 호출, 뷰-핸들러는 해당 뷰 객체를 만들고, 초기화 한다. 열린 뷰에 대한 자체목록에 새 뷰를 추가 자신을 디스플레이하기 위해 뷰들을 호출하고 이 때 보조 컴포넌트로 부터 데이터를 가져온다.

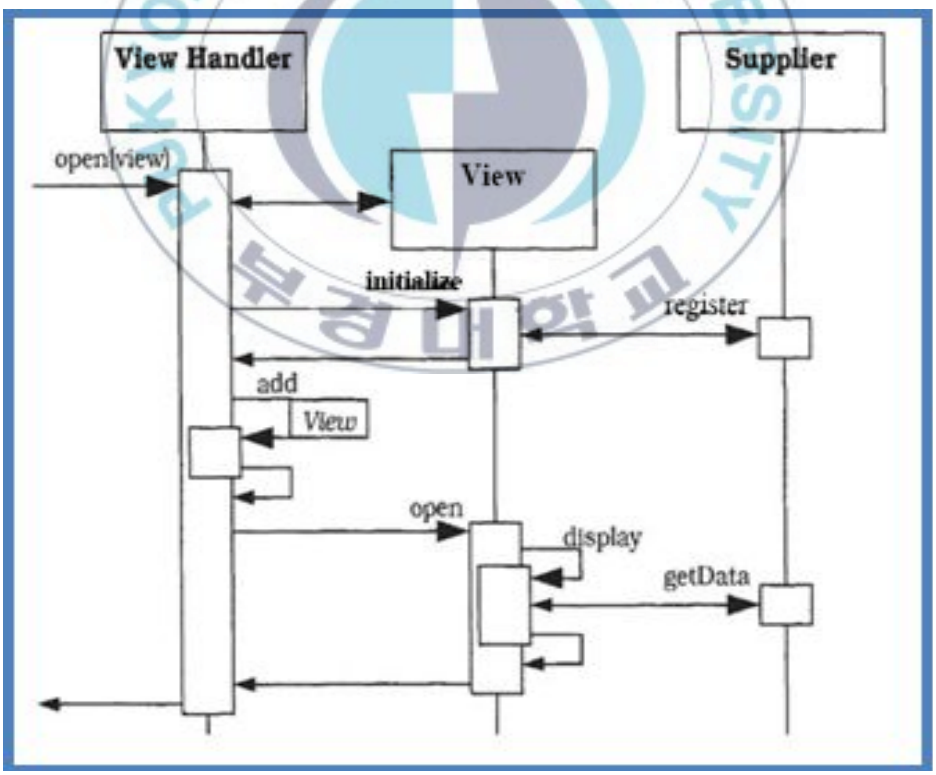

그림 3.4 뷰-핸들러 디자인 패턴의 동작의 예

3.3. 트리플 버퍼 (triple Buffer)

3.1절에서 설명한 구조의 3차원 그래픽 프로그램 개발의 구조 내에서 GUI를 구현하게 되면, 3차원 프로그램 내의 데이터 처리 및 렌더링 처리 시간 이후 GUI부를 렌더링 하게 되는데, 이 경우 이벤트 감지 및 메시지 처리시간도 매 프레임마다 늦어지게 되기 때문에 유저와의 상호작용이 느려지는 큰 단점을 안게 된다. 이를 문제를 해결하기 위해 본 논문에서는 프로그램 및 GUI 렌더링 모듈과 프로그램 프로세스 처리부를 각각 독립적 구조로 구성해 멀티 프로세싱하기로 한다.

듀얼 코어 이상급의 CPU를 탑재한 시스템에서는 일반적인 단일 스레드 구조보다 2개 이상의 물리적 코어를 활용한 멀티 스레드 구조가 더 빠른 처리속도를 보여주는 장점을 가지지만 다중 스레드 구조에서는 같은 리소스를 공유할 때 충돌이 일어날 수 있기에 크리티컬 세션(CriticalSection), 세마포(Semaphore), 뮤텍스(Mutex) 등의 동기화와 다양한 방식의 비동기화 처리 방법을 해야만 한다. 동기화 방식은 결국 가장 느린 스레드를 기다려 속도를 보장하는 반면, 비동기 방식에서는 각각의 최대 스레드의 최고 속도의 성능을 보여주기 때문에, 본 논문에서는 트리플 버퍼를 사용한 비동기 방식을 사용하여 빠른 응답속도를 보장하는 GUI 개발 도구를 구현하고자 한다.

트리플 버퍼의 구조는 두 스레드 간 공유되는 데이터를 3개의 버퍼를 사용하여 서로 충돌이 일어나지 않게끔 스케줄링 하는 구조이다[13]. 위의 그림 3.5는 단일 스레드와, 더블 스레드에서의 더블 버퍼, 트리플 버퍼의 구조와 처리속도의 비교이다. A는 단일 스레드에서는 데이터 처리 모듈에서 렌더링에 필요한 데이터 처리를 한 후 렌더링 모듈을 통해 화면에 디스플레이하는 구조이다. B와C는 데이터 처리

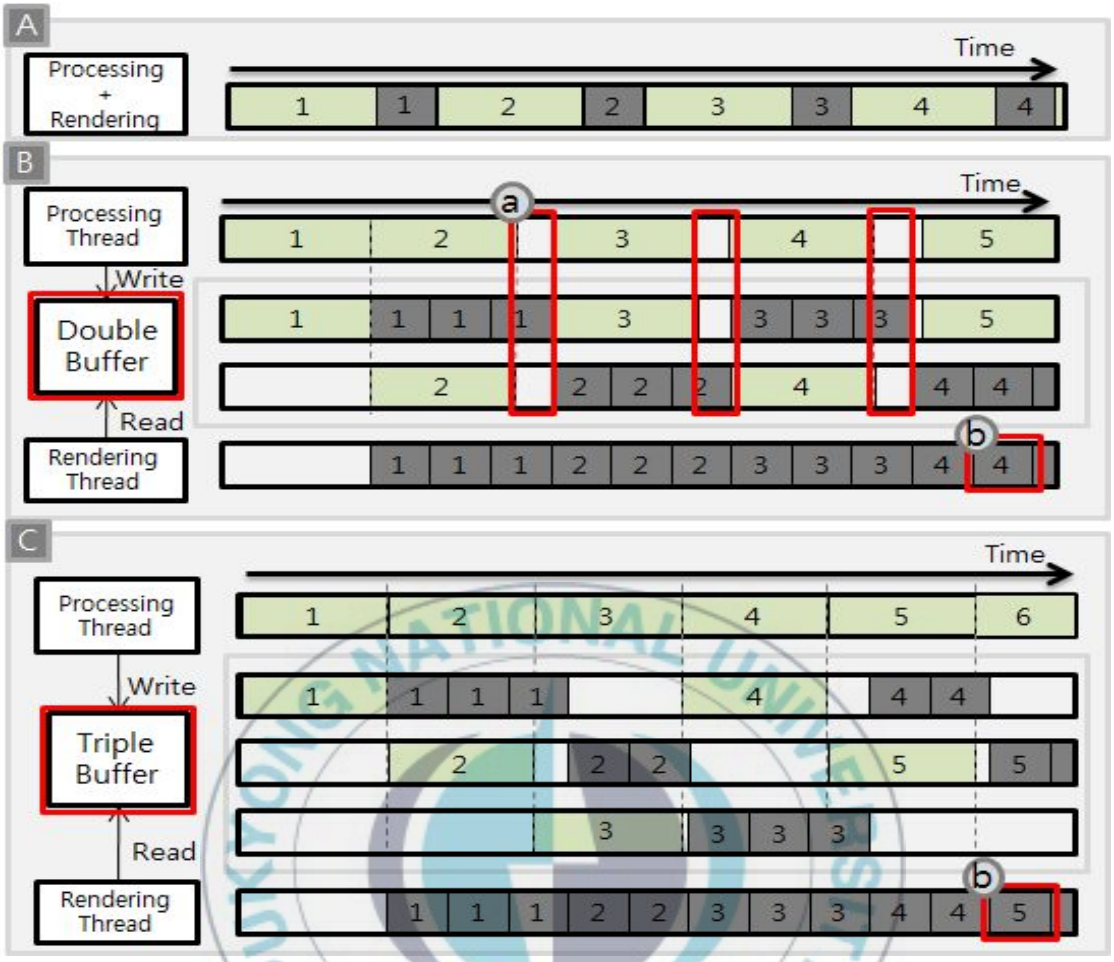

그림 3.5 단일 스레드, 더블 스레드에서의 더블 버퍼
트리플 버퍼의 구조와 처리 속도 비교

모듈과 렌더링 모듈을 각각 스레드로 구현하고, 두 스레드 간의 데이터 공유를 각각 더블버퍼, 트리플 버퍼를 사용한 구조이다. 데이터 처리 모듈에서는 입력된 데이터 및 이벤트를 통해 해당 프로그램의 프로세싱을 하고 최근에 쓰이지 않은 버퍼에 데이터를 쓰는 작업을 반복한다. 렌더링 모듈에서는 최근 쓰인 버퍼에서 프레임 데이터를 읽어 화면에 디스플레이를 반복하는 구조이다.

A에서는 데이터 처리 모듈에서 프로그램 프로세싱 후에 랜더링 모듈을 통해 화면을 업데이트하기 때문에 비교적 느린 속도를 보여준다. B의 수행 과정에서 데이터 처리 모듈 스레드가 최근 쓰이지 않은 버

퍼에 데이터를 쓰려할 때, 렌더링 모듈 스레드가 데이터를 읽고 있는 문제가 발생된다. 이러한 경우 스레드 간 충돌이 일어날 수 있기 때문에 a와 같이 데이터처리 모듈 스레드에서 지연 시간이 발생한다. C의 트리플 버퍼의 구조에서는 추가적으로 현재 사용 중이지 않는 버퍼에 데이터 쓰기가 가능해져 대기 지연 시간이 사라지고, 각각 스레드의 최고속도를 보장 할 수 있기에 결과적으로 b와 같이 비교적 빠른 프레임률을 보여준다. 이때 렌더링 모듈에서 GUI렌더링, 이벤트 감지 등을 담당함으로 데이터 처리 스레드의 부화가 많아져 실시간에 이루어지지 못하더라도 사용자와 자연스러운 상호작용이 가능해진다.

4. 개발 도구 구현

본 장에서는 앞 장에서 제안한 방법들이 적용된 UX를 고려한 3차원 그래픽 프로그램의 GUI 개발 도구 구현 방법과 각각 컨트롤들의 기능을 살펴본다.

4.1. 시스템 개요

제안한 방법을 통해 구현된 시스템은 다음 그림 4.1과 같은 구조를 가진다. 3차원 그래픽 프로그램에서 게임이나 시뮬레이션처럼 물리 프로세싱을 필요로 하는 경우는 주 스레드에서 물리 프로세싱을 하고 화

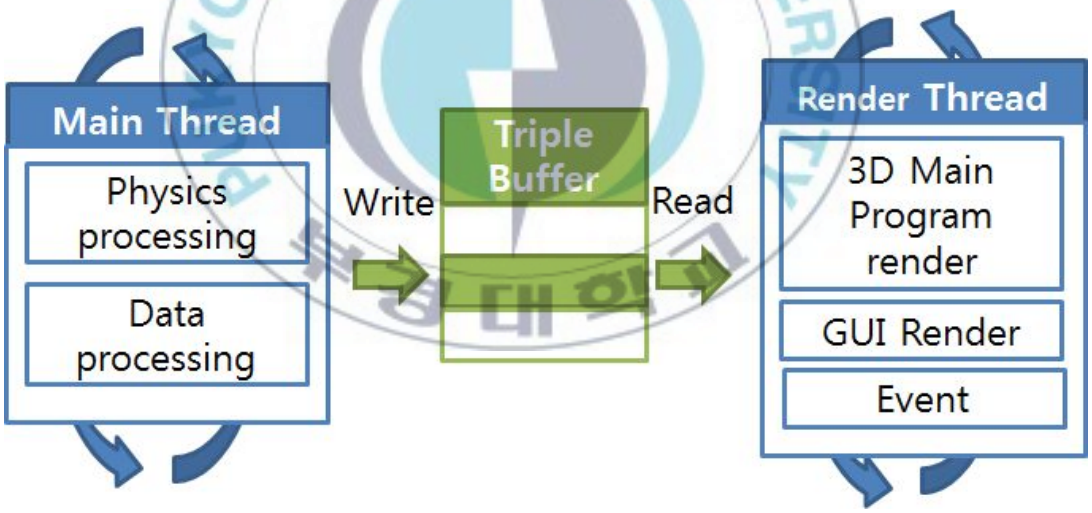

그림 4.1 전체 시스템 흐름

면에 렌더링하거나 보여줄 데이터를 처리 프로세싱을 한다. 그 후 트리플 버퍼를 통해 화면에 그려줄 데이터를 쓰는 하는 과정을 반복하게 된다. 동시에 렌더링 스레드에서는 트리플 버퍼에서 최근에 쓰인 데이터를 읽어 렌더링 하고, 이벤트 감지 및 메시지 처리도 담당한다.

본 논문에서 설계 및 구현한 UX를 고려한 3차원 그래픽 프로그램의 GUI 개발 도구는 WinAPI환경에서의 OpenGL 기반으로 구현하였다. 여기에서 DirectX와 같은 다른 3차원 API 라이브러리를 사용하여 구현이 가능하도록 하기 위해 렌더링부분의 Draw함수만 해당 3차원 API를 사용한다. 따라서 Draw함수만의 포팅을 통해 다른 3차원 API 라이브러리 적용이 가능하도록 구현해 이식성을 높인다.

4.2. 컨트롤 구현

이번 절에서는 뷰-핸들러 디자인 패턴을 적용한 실제 구현된 GUI 개발도구의 각각의 컨트롤의 기능과 구현방법을 살펴본다. 아래 그림 4.2는 개발도구의 구조이다.

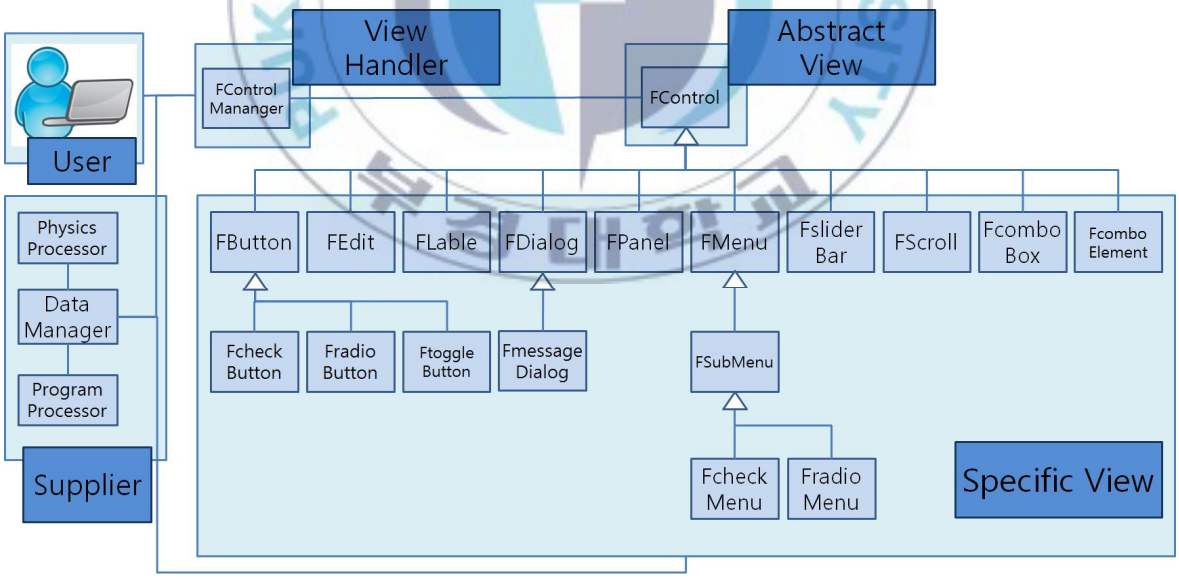

그림 4.2 뷰-핸들러 디자인 패턴이 적용된 개발도구 구조

4.2.1. FControl

뷰-핸들러 디자인 패턴에서 추상뷰에 해당하는 FControl이다. FControl은 상세뷰들의 공통의 속성과 인터페이스를 정의한 공통적인 부분들을 정의 및 구현을 한다. 즉 반경, 컬러, 텍스쳐, 그리기 등에 대해서는 정의하고 구현 한다. GUItype이라는 맴버를 가지도록 구현하는데 현재 컨트롤의 타입을 지정하는 맴버이다. 각 컨트롤의 타입의 Eunm 값은 부록-가를 참조 한다.

각 컨트롤에는 콜백 함수 onMouseEvnetFunc, clickMouseEvnetFunc 이라는 맴버를 통해 원하는 이벤트 함수를 따로 구현할 수 있도록 제공한다. 이 때 이벤트 함수는 int형 리턴 값을 가지는데 특수한 컨트롤은 아래 표 4.1의 값으로 리턴을 해주어야 한다. 다이얼로그는 현제 렌더링 된 화면에서 위에 존재하도록 하여, 뒤의 컨트롤들이 이벤트 감지 및 메시지 처리를 하지 않도록 하기 위함이고 에디트의 경우는 WinAPI의 에디트 컨트롤에 의존한 문자열 처리를 하고 있기 때문 등의 다양한 이유가 있다.

또 m_state 맴버는 각종 상태값을 비트 플래그를 통해 각 값을 설정할 수 있다. 함수의 설정을 통해 화면에 보이게 하거나, 이벤트 감지 여부 사용여부 등의 다양한 상태 값을 변경할 수 있고, 반경의 속성, 즉 화면에 유동적으로 크기가 변경되거나, 고정되는 등의 설정함수 등에서 상태 값의 입력을 통해 해당 속성을 활성화, 비활성할 수 있다. 상태 값의 Enum Value 와 설명은 부록-나를 참조하도록 한다.

Event 명	Enum Value	설명
EVENT_NOTHING	0	특별한 이벤트가 발생하지 않을 경우의 return value
EVNET_MENU	1	Menu를 클릭했을 경우 return value
EVENT_LEAP_MENU	2	LeapMenu를 클릭했을 경우 return value
EVENT_MENU_OPEN_DIALOG	3	Dialog가 열렸을 경우의 return value
EVENT_MENU_CLOSE_DIALOG	4	Dialog가 닫혔을 경우의 return value
EVENT_COMBO_ELEMENT	5	ComboControl의 Element값이 클릭했을 경우 리턴값
EVENT_EDIT	6	EditControl을 클릭했을 경우 return value
EVENT_END	7	EventValue의 마지막 Value

표 1.1 FCtronl Event 리턴값 및 설명

4.2.2. FButton

뷰-핸들러 디자인 패턴에서 상세 뷰에 해당하는 컨트롤 이며, 일반 버튼에 해당한다. 클릭 시 이벤트를 줄 수 있으며 원하는 텍스쳐를 입힐 수 있다.

4.2.3. FToggleButton

뷰-핸들러 디자인 패턴에서 상세 뷰에 해당하는 컨트롤 이며, 버튼과 비슷하지만 첫 클릭 시 오목하게 들어가고 두 번째 클릭 시 다시 원상태로 돌아오는 형태의 버튼이다. 일반 버튼, 다른 컨트롤과 다르게 클릭시 텍스쳐를 하나 더 가지고 있어 클릭 시 설정한 텍스쳐로 전환이 가능한 특징을 가진다. 또 현재 클릭이 되어있는지, 처음 상태인지를 boolean 값을 반환하는 함수를 통해 알 수 있다.

4.2.4. FRadioButton

뷰-핸들러 디자인 패턴에서 추상 뷰에 해당하는 컨트롤 이며, 맴버로 bool m_check와 FRadioGroup *m_RadioGroup을 가지고 있다. 보통 2개 이상의 FRaidoButton을 생성해 사용하고 RadioGroup에 등록을 하고 한 버튼이 눌러졌을 경우 버튼의 m_check 값이 false로 변하게 된다. 특별한 Draw부에서 구현이 없을 경우 현재 텍스쳐 위에 왼쪽 끝에 하얀 원이 그려지고 m_check값이 참일 때 해당원에 검은색으로 채워진다.

4.2.5. FRadioGroup

뷰-핸들러 디자인 패턴에서 상세 뷰에 해당하는 컨트롤 이며, 맴버로 FRadioButton *m_memberRadioButton, int m_selectedIndex를 가진다. 앞에서 설명한 FRadioButton을 맴버로 가지고있어 해당 버튼이 클릭 했을 시 다른 버튼의 m_check를 false로 만들고 클릭된 RadioButton의 index를 기억하고 이값을 받을 수 있다.

4.2.6. FCheckButton

뷰-핸들러 디자인 패턴에서 추상 뷰에 해당하는 컨트롤 이며, 맴버로 bool m_check를 가지고 있다. 클릭 시 해당 m_check가 true일 때는 false , false 일 때는 true가 된다. 특별한 Draw부에서 구현이 없을 경우 현재 Texture 위에 왼쪽 끝에 하얀 사각형이 그려지고 m_check값이 참일 때 해당 사각형에 검은색 체크표시가 그려진다.

4.2.7. FLabel

뷰-핸들러 디자인 패턴에서 추상 뷰에 해당하는 컨트롤 이며, 일반적으로 텍스트 값만을 그리기 위해 구현된 컨트롤이다. 생성자를 통해 이벤트를 감지하는 상태 값이 거짓으로 설정되어 이벤트 감지 및 메시지 처리를 하지 않는다.

4.2.8. FLayer

뷰-핸들러 디자인 패턴에서 추상 뷰에 해당하는 컨트롤 이며, 일반적으로 Text값이 없고 Image를 출력하기 위해 구현된 컨트롤이다. FLabel과 마찬가지로 생성자를 통해 STATE_ENABLE 값이 false로 설정되어 이벤트 감지 및 메시지 처리를 하지 않는다.

4.2.9. FMenu

뷰-핸들러 디자인 패턴에서 추상 뷰에 해당하는 컨트롤 이며, 맴버로 FSubMenu *m_subMenu를 가진다. 화면에 메뉴를 위해 구현된 컨트롤이며 클릭 시 속한 m_subMenu 맴버들이 화면에 그려진다. 또

특별한 이벤트를 구현 할 수 없으며, 이벤트 함수의 리턴값으로 FControl::EVNET_MENU를 리턴한다.

4.2.10. FSubMenu

뷰-핸들러 디자인 패턴에서 추상 뷰에 해당하는 컨트롤 이며, 앞의 FMenu를 클릭시에 화면에 그려질 컨트롤이다. 맴버로 FMenu *m_menu를 가지고 클릭시 이벤트 함수의 return값으로 FControl::EVENT_LEAP_MENU 를 리턴해 같은 레벨의 SubMenu를 모두 보이지 않는 상태로 만든다.

4.2.11. FCombo

뷰-핸들러 디자인 패턴에서 추상 뷰에 해당하는 컨트롤 이며, 클릭시 해당 컨트롤 아래로 설정한 Element 들이 보이고 선택 시 FCombo의 내용이 선택한 값으로 변하는 컨트롤이다. 맴버로 int m_intdrawButtonIndex 와 FScrollControl *m_connectScrollControl를 가지고 있다. FScrollControl를 연결하고 등록된 FComboElement 수가 m_intdrawButtonIndex 보다 작을 경우 스크롤이 그려지고 클릭을 통해 보여지는 FComboElement 설정한다.

4.2.12. FComboElement

뷰-핸들러 디자인 패턴에서 추상 뷰에 해당하는 컨트롤 이며, FCombo 클릭 시 화면에 보여지는 컨트롤이다. 맴버로 FComboControl *m_parentCombo를 가지고 부모가 될 Combo를 setParentCombo 함수를 통해 등록 해야 한다. 해당 컨트롤 클릭 시

TextValue가 m_parentCombo에 복사된다.

4.2.13. FScollBar

뷰-핸들러 디자인 패턴에서 추상 뷰에 해당하는 컨트롤 이며, FCobom의 Element의 수가 설정 값 보다 클 경우 화면에 Draw 된다. 맴버로 FComboControl *m_scrollSetcontrol, FScrollButtonControl m_scrollUpButton, FScrollButtonControl m_scrollDownButton,를 가지고 있다. 생성시 setComboControl 함수를 통해 FCombo와 연결을 해줘야 한다.

4.2.14. FScollButton

뷰-핸들러 디자인 패턴에서 추상 뷰에 해당하는 컨트롤 이며, FScollBar 생성시 자동으로 생성된다. 맴버로 bool m_isUpButton를 가지고 있고 해당 값에 따라 클릭 이벤트를 통해 FComboElement 의 보여질 요소 값을 설정한다.

4.2.15. FPanel

뷰-핸들러 디자인 패턴에서 추상 뷰에 해당하는 컨트롤 이며, 컨테이너 형 컨트롤이라 맴버로 FControl *m_subFControl[SUB_ MAX]을 가지고 있다. 생성시 setEnable함수를 통해 false로 설정되어 이벤트 감지는 하지 않는다. 또 그리기 및 이벤트 감지 함수에서 SubControl에 대한 내부적으로 검사한다.

4.2.16. FDialog

뷰-핸들러 디자인 패턴에서 상세 뷰에 해당하는 컨트롤 이며, 컨테이너 형 컨트롤이라 맴버로 FPanel과 동일하게 FControl *m_subFControl[SUB_ MAX]을 가지고 있다. 추가적으로 FControl *m_okButton, FControl *m_cancelButton를 맴버로 가지고 있다. 또 타이틀 바 설정과 위에서 살펴본 4.2.1 FControl 의 STATE값에 따라 버튼이 바뀐다.

4.2.17. FMessageBox

뷰-핸들러 디자인 패턴에서 상세뷰에 해당하는 컨트롤 이며, FDialog를 상속 받은 함수로 Static으로 하나만 생성된다. 간단한 경고, 확인, 메시지를 화면에 보여주는 컨트롤이다. 맴버로 char *m_titleText char *m_messageText를 가지고 타이틀설정과 , 표시할 메시지를 설정할 수 있다. SetMessageType 함수를 통해 해당 메시지의 버튼 타입과 아이콘을 변경 할 수 있다. 메시지 버튼 타입을 위한 메시지 박스의 Enum값은 부록-다를 참조한다.

4.2.18. Fcontrol Mannger

뷰-핸들러 디자인 패턴에서 뷰-핸들러에 해당하는 FControlManager 이다. 추상뷰에 해당하는 FControl 객체형태의 맴버 객체를 가지고 이를 통해 각각의 생성된 추상뷰들을 다형성을 통해 관리를 한다. 이 때 효율적 관리를 위해 화면에 그려질 객체를 스택

구조로 추가, 삭제 등의 관리를 한다. OpenGL의 렌더링 함수를 통해 화면에 Fcontrol Manager 객체의 렌더 함수를 호출하면 현재 스택에 속한 모든 컨트롤 객체의 렌더 함수를 호출하게 되어 현재 디스플레이 할 인터페이스의 뷰를 그려주게 된다. 또 WinAPI를 통해 마우스 이동, 마우스 클릭, 마우스 버튼 업, 드래그 등의 이벤트를 통해 Fcontrol Mannger 객체의 이벤트 및 메시지 감지에 해당하는 함수를를 호출함으로써 각각의 컨트롤의 이벤트를 감지 및 실행한다. 단 구현 시에는 렌더 함수를 제외한 다른 속성 및 함수에서는 OpenGL과 독립적으로 구현하여 기존의 뷰 핸들러와 소통하게 한다.

4.3. 개발도구 사용

본 논문에서 제안한 UX를 고려한 3차원 그래픽 프로그램의 GUI 개발 도구를 사용하는 구현에서는 다음과 같은 순서에 따른다.

1. FControlManager객체를 생성을 한다.
2. 필요한 FButton, FMenu 등 각종의 뷰들의 객체를 생성하고, 화면에 배치할 위치, 텍스쳐, 텍스트등의 각종 속성을 설정한다.
3. 각각의 이벤트 콜백함수를 구현하고 이벤트 리스너 함수에 해당함수를 연결한다.
4. FControlManger객체의 PushStack함수를 통해 현재화면에 보여질 뷰의 객체를 스택 구조에 추가한다.
5. 3D 랜더링부에서 FControlManager객체의 OnDraw 함수를 호출한다.
6. 이벤트 뷰에서 해당하는 FcontrolManager객체의 이벤트 함수를 호출한다

5. 구현 결과

제안한 UX를 고려한 3차원 그래픽 프로그램의 GUI 개발 도구를 사용하여 구현한 의료 광학장비 뷰어[14][15]의 GUI와 각 컨트롤들이다. 먼저그림 5.1는 좌측에서 순서대로 메뉴, 서브메뉴, 버튼, 패널과 라벨, 레이어, 슬라이더 바 등의 구현 결과이다.

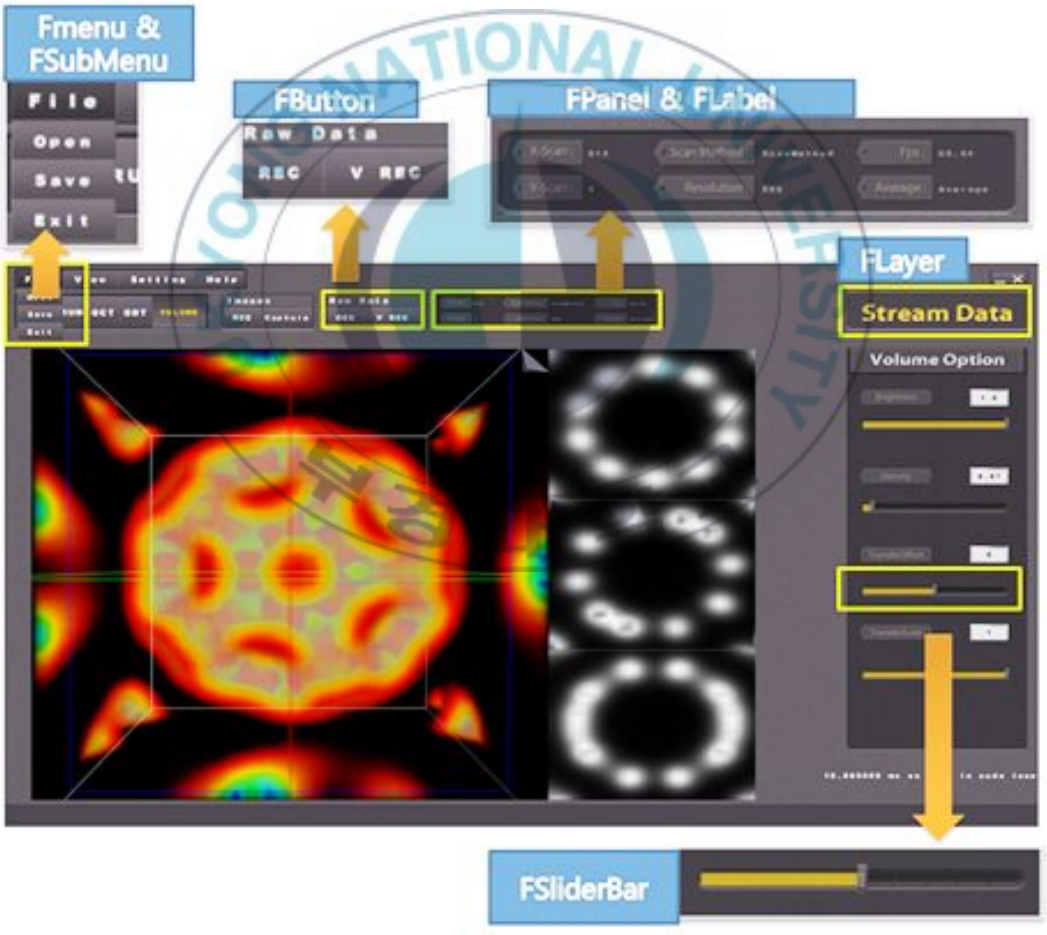

그림 5.1 3차원 의료 광학장비 뷰어에서의 GUI 와 컨트롤 1

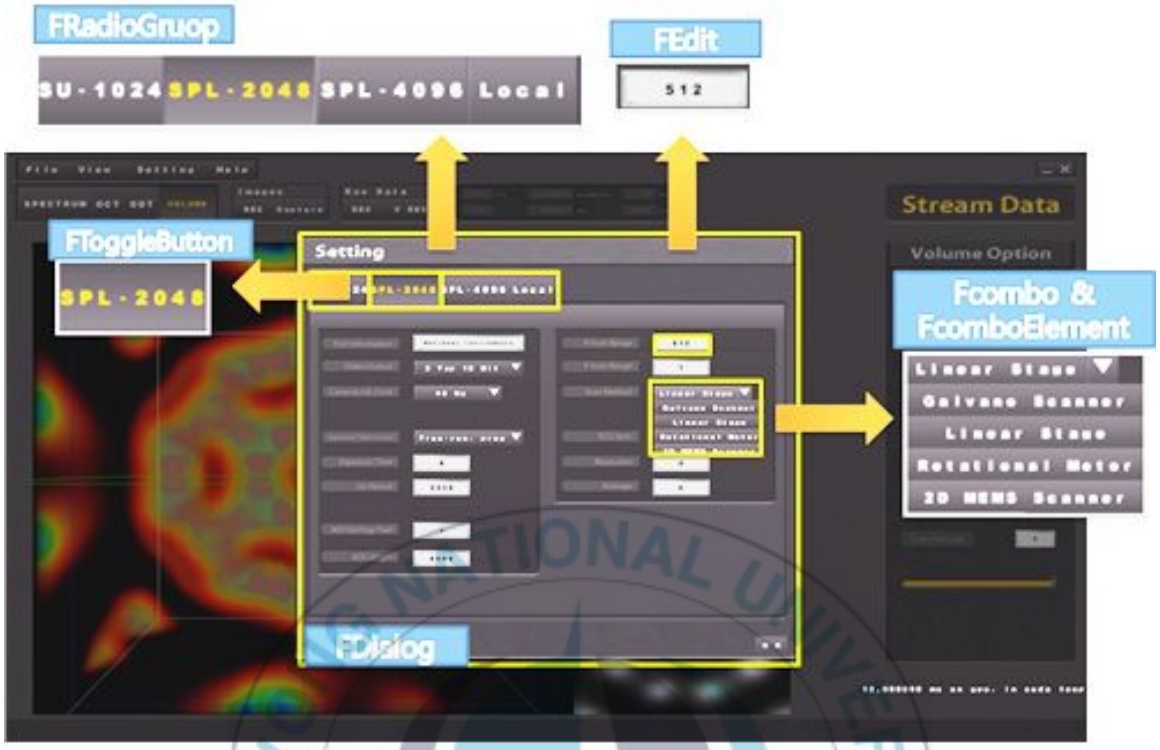

그림 5.2 3차원 의료 광학장비 뷰어에서의 GUI 와 컨트롤 2

이어서 그림 5.2는 라디오 그룹, 에디트, 토글버튼, 콤보박스 및 콤보 엘리먼트, 다이얼로그 등의 구현 결과이다. 기존의 그림 1.1에서 사용된 WinAPI 및 MFC 기반의 의료 광학장비 뷰어의 정형화된 GUI보다 UX를 고려한 다양한 그래픽 효과의 적용과, 사용자의 편의성이 높은 GUI를 구현을 확인 할 수 있었다.

그림 5.3는 제안한 GUI 개발도구를 사용한 3차원 어로 시뮬레이터로 앞의 의료 광학장비에서 사용한 GUI 개발도구를 사용하였지만, UX를 고려하여 어선의 계기판과 같은 GUI를 구현을 하였고, 배의 핸들과 레버를 3차원 모델로 구현하여 더 입체감 있고, 사실적인 GUI 구현 할 수 있었다. 또 빠른 개발 속도와 개발의 용이성을 확인 할 수

있었다.

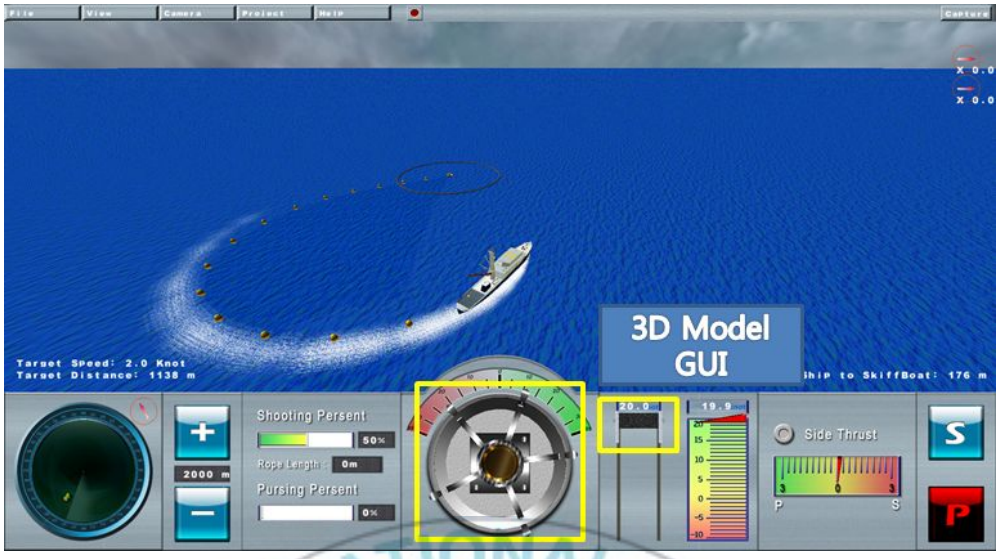

그림 5.3 3차원 어로 시뮬레이터에서의 GUI

구현한 3차원 OCT 의료 광학장비 뷰어에서 GUI부와 프로세스부를 나눠 트리플 버퍼를 사용해 비동기화 함으로써 표 1.2의 시뮬레이

목 록	내 용
운영체제	Window 7, Professional K 64bit
CPU	Inter® Core™ i7- 3970X CPU
메인 Memory	8GB Memory
그래픽 카드	NVIDA GeForce GTX 760 (Memory 4GB , CUDA Core 1024)

표 1.2 시뮬레이션 환경

션 환경에서 실행 하였을 때, 그림 5.4와 같은 결과를 보여주었다. 먼저 좌측의 결과와 같이 다양한 알고리즘이 적용[16][17]된 OCT 영상의 프레임률은 약 5.9 프레임이 상승하였다.GUI부의 렌더링 속도만

을 비교했을 때 우측의 결과와 같이 싱글 스레드에서는 49.96ms마다 화면을 갱신해 이벤트 감지 및 처리를 하고, 더블 스레드에서 트리플 버퍼를 적용하였을 때에는 10.18ms마다 화면을 갱신하고 이벤트 감지 및 처리를 해 빠른 응답속도를 확인 할 수 있었다.

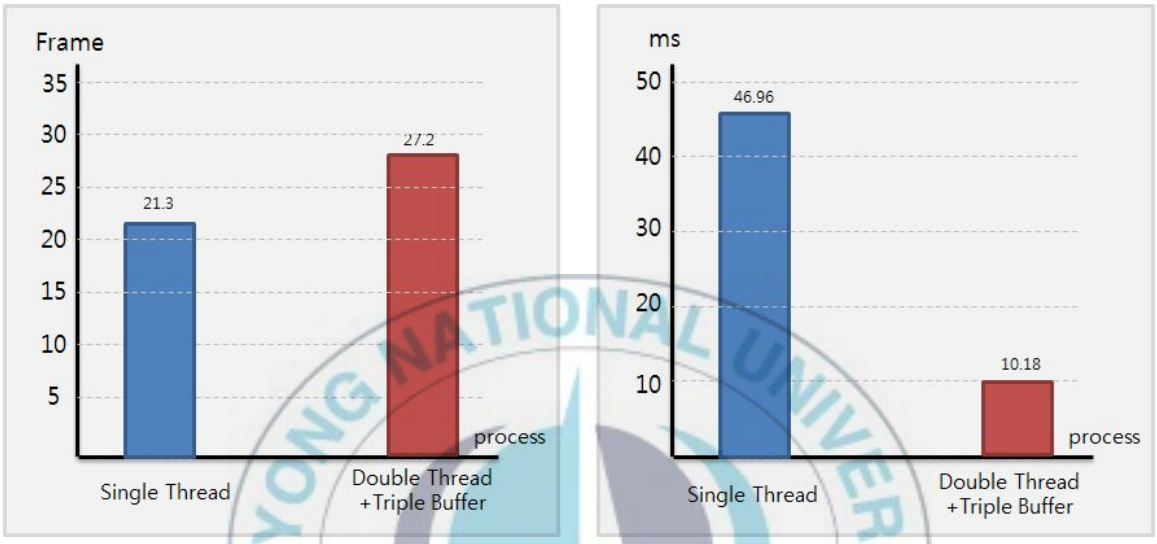

그림 5.4 의료 광학 장비의 프레임율(좌)와 지연시간(우)

6. 결론 및 향후 연구

6.1. 결론

본 논문에서는 GUI가 렌더링 영역 내에 프로그램의 일부처럼 보이도록 일반적인 3차원 그래픽 프로그램 개발의 구조 안에서 GUI를 구현하고, 이를 통해 쉐이더 및 3차원 모델을 사용하는 등의 다양한 그래픽 효과 적용이 가능하고자 하였다. 또 뷰-핸들러 디자인 패턴 구조를 적용해 개발의 용이성과 이식성을 높이고, 다양한 구조가 적용가능하도록 설계해 UX디자인을 고려하여 유저의 편이성을 높이는 시도와, GUI 렌더링 모듈과 3차원 프로세스 처리부를 각각 독립적 구조로 구성해 멀티 프로세싱하고, 트리플 버퍼를 사용하여 비동기화 하여 빠른 응답속도를 보장하는 GUI 개발 도구를 설계 및 구현하고자 하였다.

실제 3차원 시뮬레이터 및 소프트웨어 개발을 통해 본 논문에서 제안하는 방법이 다양한 그래픽 효과를 적용이 가능하고, 개발의 용이함과 사용자와의 상호작용이 자연스러움을 확인하였다.

6.2. 향후 연구

본 논문에서 구현한 GUI 개발도구에선 제공하지 못한 다양한 컨트롤이 존재하고, 기존에 개발한 OpenGL에 종속적인 부분이 존재하여, OpenGL ES 나 DirectX와 같은 다른 3차원 API 라이브러리 포팅을 통한 이식성은 확신할 수가 없었다. 차후 다양한 애니메이션과 쉐이더를 통해 더욱 사실적이고 그래픽적인 특수 효과를 추가적으로 적용에 대한 연구가 필요하다.

참고 문헌

[1] 전준영, “View Handler 디자인 패턴을 적용한 3D GUI 개발 도구 구현”, 2013년도 한국멀티미디어학회 추계학술발표대회 논문집 제16권2호
[2]이건호, “어로 시뮬레이션 모델 개발 및 시뮬레이터 구현, 학위논문(박사),pp 44- 57 ,2009
[3] 백홍선, 김남호, “Direct3D 기반의 Game UI Component 설계”, 한국멀티미디어학회 학술발표논문집, Vol.2005 No.1
[4] 두경일, “사용자 태도에 근거한 가치공학적 UX/UI 디자인 평가모형 연구”, 한국디자인지식학회, 2012.02
[5] 김형년, “다이나믹 유저 인터페이스 디자인을 적용한 GUI 설계 및 구현”, 디자인지식저널, Vol.8 No.- [2008]
[6] Alben, 1996; Hassenzahl & Tractinsky, 2006; Law et al.,2008; Nielsen & Norman Group, n.d.; wikipedia, n.d.
[7]전종홍, 이승윤, “차세대 웹에서의 UI/UX 기술 동향 ”정보과학회지 , 제 29 권 8호, pp.9-17, 2011.
[8] 권애경,” UX 모델과 UX 디자인 프로세스 기반한 UX 디자이너 역량모델 및 교육 방향에 관한 연구 ” 학위논문(석사) - 한동대학교 일반대학원 : 문화미디어디자인학과 2013. 2
[9] David Zeltzer “Autonomy, interaction, and presence” Teleoperators and Virtual Environments. Vol 1, No1, pp127-132, 1992
[10] 이수경, 임창영, “웹에서의 3차원 정보 전달에 관한 연구”, 디자인학연구, Vol.11, No.1, pp57, 2001
[11]유상진, “휴대용 멀티미디어 기기를 위한 임베디드 리눅스 기반 GUI 라이브러리의 설계 및 구현”, 학위논문(석사), 수원대학교 대학원: 컴퓨터학과 pp35-42, 2005. 2
[12]Frank Buschmann, Regine Menunier, Hans Rohnert, Peter Sommerlad, Michael Stal, “Pattern-Oriented Software Architecture, Volume 1: A System of Patterns” , John Wiley & Sons, pp 307-322 , 1997
[13] Khan, Shujjat, Donald Bailey, and G. Gupta. "Simulation of Triple Buffer Scheme." Second International Conference on Computer and Electrical Engineering. 2009.
[14] 신승원, “OCT 영상의 화질 개선을 위한 방향성 필터의 자동 생성 및 적용”, 학위논문(박사)-- 건국대학교 대학원 : 의학공학과 의학공학 2013. 2

[15] Abdulnasir Hossen, “ A NEW FAST APPROXIMATE HILBERT TRANSFORM WITH DIFFERENT APPLICATIONS “. Information Engineering Department, Sultan Qaboos University, P.O.Box 33 Al-Khod, 123
[16] Shuichi Makita, Tapio Fabritius, and Yoshiaki Yasuno , “Full-range, high-speed, high-resolution 1-μm spectral-domain optical coherence tomography using BM-scan for volumetric imaging of the human posterior eye”, Optics Express, Vol. 16, Issue 12, pp. 8406-8420 (2008)
[17] Kang Zhang, Student Member, IEEE, and Jin U. Kang, Member, IEEE, “Graphics Processing Unit-Based Ultrahigh Speed Real-Time Fourier Domain Optical Coherence Tomography”, IEEE JOURNAL OF SELECTED TOPICS IN QUANTUM ELECTRONICS, VOL. 18, NO. 4, JULY/AUGUST 2012

부 록

가. FCtronl GUITYPE 설정값

GUITYPE 명	Enum Value
GUITYPE_FCONTROL	0
GUITYPE_BUTTON	1
GUITYPE_TOGGLEBUTTON	2
GUITYPE_EDITCONTROL	3
GUITYPE_LABEL	4
GUITYPE_DIALOG	5
GUITYPE_MENU	6
GUITYPE_SUBMENU	7
GUITYPE_RADIOBUTTON	8
GUITYPE_RADIOGROUP	9
GUITYPE_CHECKBOXBUTTON	10
GUITYPE_PANEL	11
GUITYPE_COMBO	12
GUITYPE_COMBOELEMENT	13
GUITYPE_LAYER	14
GUITYPE_SCROLL	15
GUITYPE_SCROLLBUTTON	16
GUITYPE_MESSAGEBOX	18
GUITYPE_END	19

나. FControl 의 STATE의 설정 flag값과 설명

Flag STATE 명	Enum Value	설명
STATE_VISIBLE	0x00000001	화면에 Draw (렌더링) 되도록 한다.
STATE_ENABLE	0x00000002	이벤트에 반응하지 않도록 한다.
STATE_ENABLEBLEND	0x00000004	텍스쳐에 알파 값을 설정하도록 한다.
STATE_BOUND_BLEND	0x00000008	Control의 Bound부분의 알파 값이 적용되도록 한다.
STATE_BOUND_IGNORE	0x10000000	렌더링 화면 크기를 무시하도록 하고 절대 값으로 Bound를 설정한다.
STATE_BOUND_TOP_ALIGN	0x00000010	렌더링 화면에 Bound의 Top은 위치 값으로 고정한다.
STATE_BOUND_BOTTOM_ALIGN	0x00000020	렌더링 화면에 Bound의 Bottom 은 위치 값으로 고정한다.
STATE_BOUND_LEFT_ALIGN	0x00000040	렌더링 화면에 Bound의 Left는 위치 값으로 고정한다.
STATE_BOUND_RIGHT_ALIGN	0x00000080	렌더링 화면에 Bound의 Right는 위치 값으로 고정한다.
STATE_BOUND_TOP_REALTIVE	0x00000100	렌더링 Bound의 Top은 화면의 높이 비율에 맞도록 위치 값을 변경한다.
STATE_BOUND_BOTTOM_REALTIVE	0x00000200	렌더링 Bound의 Bottom은 화면의 높이 비율에 맞도록 위치 값을 변경한다.
STATE_BOUND_LEFT_REALTIVE	0x00000400	렌더링 Bound의 Left은 화면의 높이 비율에 맞도록 위치 값을 변경한다.
STATE_BOUND_RIGHT_REALTIVE	0x00000800	렌더링 Bound의 Right은 화면의 높이 비율에 맞도록 위치 값을 변경한다.

Flag STATE 명	Enum Value	설명
STATE_TEXT_ALIGN_LEFT	0x00001000	Text값 출력 시 Bound의 Left 기준 정렬로 위치 값으로 고정한다.
STATE_TEXT_ALIGN_MIDDLE	0x00002000	Text값 출력 시 Bound의 중간을 기준 정렬로 위치 값으로 고정한다.
STATE_TEXT_ALIGN_RIGHT	0x00004000	Text값 출력 시 Bound의 Right 기준 정렬로 위치 값으로 고정한다.
STATE_DIALOG_OK	0x00010000	Dialog 생성 시 OK버튼만 생성하도록 한다.
STATE_DIALOG_OK_CANCEL	0x00020000	Dialog 생성 시 OK, CANCEL 버튼을 생성하도록 한다.
STATE_DIALOG_NEXT	0x00040000	Dialog 생성 시 Next버튼을 생성하도록 한다.
STATE_DIALOG_DONE	0x00080000	Dialog 생성 시 Done버튼을 생성하도록 한다.
STATE_END	0x80000000	STATE값의 마지막 값

다. MessageBox의 Type Enum값

MessageType 명	Enum Value
FMB_OK	0
FMB_OKCANCEL	1
FMB_YESNO	2
FMB_YESNOCANCEL	3
FMB_ICONINFORMATION	4
FMB_ICONEXCLAMATION	5
FMB_ICONSTOP	6
FMB_ICONQUESTION	7
FMB_END	8