KAERI/TR-980/98

윈도우즈 환경에서의 원격로보트 제어시스템 개발

Development of Windows based Control System for the Telerobotics

한국원자력연구소

제 출 문

한국원자력연구소장 귀하

본 보고서를 1998 년도 기술보고서로 제출합니다.

제목 : 윈도우즈 환경에서의 원격로보트 제어시스템 개발

1998. 2. 25

주 저 자 : 선임연구원 김 병 수

공 저 자 : 책임연구원 김 승 호
선임기술원 서 용 칠
선임연구원 김 창 회
선임연구원 황 석 용
선임연구원 김 기 호
선임연구원 정 승 호
선임기원 이 영 광

연구조원 홍 준 표

감수위원 : 책임연구원 윤 지 섭

요 약 문

중수로형 원자력발전소내 고방사능 지역과 같이 인간의 접근이 불가능한 지역에서 인간을 대신하여 감시작업을 수행할 수 있는 이동로보트의 효율적인 제어를 위하여 윈도우즈 환경 관리제어시스템을 개발하였다. 개발된 시스템은 크게 윈도우즈 메뉴 프로그램, 소켓을 이용한 통신 프로그램, 힘반향 조이스틱 프로그램 및 OpenGL을 이용한 3차원 그래픽 프로그램으로 구성하였다. 또한 조이스틱의 조작성을 향상시키기 위하여 이동로보트를 위한 힘반향 제어 알고리즘을 개발하였다.

설계된 윈도우즈형 관리제어시스템과 힘반향 제어 알고리즘의 효용성을 평가하기 위하여 가상환경에서 조이스틱 동작 테스트를 수행하였다. 테스트 결과 설계된 윈도우즈형 관리제어시스템의 효용성이 매우 크고 사용하기가 용이함을 알수 있었다. 또한 힘반향 제어알고리즘 실험결과 사람에 따라 다소 차이는 있으나 일반 원격제어 방식에 비하여 매우 효율적임을 확인할 수 있었다.

향후 본 연구에서 개발된 결과는 KAEROT의 차기 제품에 적용될 예정이다.

ABSTRACT

The WSCS (Windows-based Supervisory Control System) has been developed for the efficient control of the mobile robot in the harzardous area, such as reactor surroundings of HPWR (Heavy Pressurized Water Reactor). The WSCS is basically computer program which consists windows menu-program, socket-based communication program, force reflection joystick program, and OpenGL-based 3D graphic program. Also, the WSCS includes the force reflection control algorithm of a master control device (in this case, joystick) for the enhanced operability.

To evaluate the effectiveness of the designed WSCS and the force reflection control algorithm, a series of experiments has been made in such a way that human operators command the desired motion of robot by manipulating the joystick in the virtual environment. As a result, it was proven that the designed WSCS is very easy-to-use and effective. Also, the developed force reflection algorithm is more efficient than that of general teleoperation, even though there are some differences in human dexterity.

In near future, the WSCS will be applied in the next version of KAEROT.

CONTENTS

ABSTRACT

Chapter 1. Introduction ---------- 1

Chapter 2. Main Discourse ---------- 2

Section 1. Windows Program ---------- 6

Section 2. Communication Program ---------- 9

Section 3. Force Reflection Joystick ---------- 39

Section 2. Graphic Program ---------- 46

Chapter 3. Conclusion ---------- 65

References ---------- 66

목 차

요 약 문

제 1 장 서 론 ---------- 1

제 2 장 본 론 ---------- 2

제 1 절 원도우즈 프로그래밍 ---------- 2

제 2 절 통신 프로그램 ---------- 9

제 3 절 힘반향 조이스틱 프로그램 ---------- 39

제 4 절 그래픽 프로그램 ---------- 46

제 3 장 결 론 ---------- 65

참 고 문 헌 ---------- 66

부 록 ---------- 67

그 림 목 차

그림 1. Visual C++ 프로그래밍 ---------- 7
그림 2. OSI 7 Layer 개념도 ---------- 9
그림 3. 데이터 링크/물리적 계층의 개념도 ---------- 10
그림 4. TCP/IP 모델 ---------- 12
그림 5. 이동로보트의 통신 개념도 ---------- 13
그림 6. 통신 흐름 순서 ---------- 13
그림 7. 서버/클라이언트의 연결 및 동작순서 ---------- 15
그림 8. 통신 프로그램 화면 ---------- 38
그림 9. 힘반향 조이스틱의 외관과 좌표계 ---------- 39
그림 10. 힘반향 조이스틱의 데이터 흐름도 ---------- 40
그림 11. 힘반향 조이스틱의 가상환경 실험 프로그램 ---------- 45
그림 12. 더블 버퍼링 ---------- 50
그림 13. PIXELFORMATDESROPTOR 구조체 ---------- 51
그림 14. 프로그램 실행화면 ---------- 64

표 목 차

표 1. 위글함수의 종류 ---------- 53

제 1 장. 서 론

이동 로보트 제어를 위한 관리제어시스템은 최근 PC에서의 윈도우즈 개발환경이 급속하게 발달함에 따라 워크스테이션의 X 윈도우즈에서 PC의 윈도우즈 운영 체제로 변천하고 있다. 특히 Visual C++ 과 같은 객체 지향 프로그램의 기능 및 성능이 크게 향상되고, 소프트웨어 호환성이 높아짐에 따라 대부분의 관리 제어프로그램이 윈도우즈 프로그램으로 포팅 및 프로그래밍 되고 있다.

개인용 컴퓨터의 운영 체제(Operating System)로, 1990년 초부터 선보인 마이크로소프트사의 윈도우즈는 사용자들의 편이성과 더불어 실질적인 윈도우즈 시대로의 개막을 알렸다. 각종 응용 프로그램은 도스용에서 윈도우즈용으로 바뀌어 출시되고, 앞으로 발표될 프로그램들 역시 '윈도우즈 버전'이다. 이처럼 윈도우즈 시대에 걸맞은 프로그램을 개발하기 위해서는 당연히 윈도우즈용 개발 도구가 필요할 것이고, 그에 따라 출시된 제품들 중 하나가 바로 마이크로소프트사의 Visual C++이다. Visual C++ 컴파일러의 특징은 시각적으로 뛰어난 환경을 제공하고 있으며, 윈도우즈 프로그래밍에서 불필요한 반복 작업을 제거하여 프로그래밍의 생산성을 높여줄 뿐만 아니라 전용 클래스 라이브러리인 MFC (Microsoft Foundation Classes)를 통해 누구나 쉽게 윈도우즈용 프로그램을 작성할 수 있도록 해주고 있고, AppWizard 및 ClassWizard 를 포함하고 있어서, MFC 기반의 프로그램을 Document/View 클래스로 나누어 프로그래머로 하여금 쉽게 개발이 가능하도록 하고 있다.

본 보고서에서는 MFC와 Visual C++ 컴파일러를 사용하여 로보트 관리제어시스템의 주 메뉴 프로그램, 네트워크 통신 프로그램, 힘 반향 조이스틱 프로그램 및 OpenGL 3D 그래픽 프로그램에 대해 다루고자 한다.

제 2 장. 본 론

제 1 절. 윈도우즈 프로그래밍

1. Win32와 Win16

1992년부터 PC의 운영체계는 Win16으로부터 Win32로 이동하기 시작했다. Win16은 마이크로소프트 Windows 3.1과 Windows for Workgroup 등을 토대로 하는 16비트 Win32 API (Application Program Interface) 를 의미하며, Win32는 마이크로소프트 Windows 95와 Microsoft Windows NT에 의해 지원되는 새로운 32비트 Win32 API를 의미한다.

Visual C++ 버전 5.0은 Win32용 응용 프로그램을 개발하기 위한 제품이다. 마이크로소프트에서는 더 이상 Win16을 이용하여 프로그램을 개발하지 않을 것을 권고하고 있다. 하지만 주위에 있는 기존의 많은 PC들은 새로운 버전의 윈도우즈를 지원하기에는 부적합한 것들이 많다. 이러한 점을 고려하여 마이크로소프트는 Windows 3.1과 Windows for Workgroup을 사용하는 기존의 PC들을 위하여 Win32s라는 라이브러리를 제공하고 있으며, Win32를 이용하여 작성된 프로그램들을 16비트 운영체제에서 동작이 가능하도록 해주고 있다.

2. 메시지 프로그래밍

C로 MS-DOS용 응용 프로그램을 작성할 때 반드시 요구되는 함수가 main 함수이다. 사용자가 프로그램을 실행할 때 운영 체제는 이 함수를 가장 먼저 호출하게 된다. 사용자의 키 입력을 받아들이거나 그 외의 운

영 체제 요소를 사용하고자 할 때 프로그램에서는 getchar 등과 같은 적절한 함수를 호출하게 된다.

윈도우즈 운영 체제는 프로그램을 출발시킬 때 프로그램의 WinMain 함수를 호출한다. 따라서 응용 프로그램의 어느 부분엔 가는 WinMain 함수가 반드시 존재해야 한다. 윈도우즈의 가장 중요한 과제는 응용 프로그램의 메인 윈도우를 만드는 일이며, 메인 윈도우는 윈도우즈로부터 전달되는 메시지를 처리한다.

MS-DOS용 프로그램과 윈도우즈용 프로그램간의 기본적인 차이점은 MS-DOS용 프로그램은 사용자 입력을 얻기 위해 운영 체제를 호출하는데 비해서, 윈도우즈용 프로그램에서는 운영 체제가 보내주는 메시지를 처리하는 방법으로 사용자 입력을 다룬다는 점이다. 윈도우즈 메시지들은 엄격하게 정의되어 있으며 모든 프로그램에 적용된다. 예를 들면, WM_CREATE 메시지는 윈도우가 만들어질 때 생성되며, WM_LBUTTONDOWN 메시지는 마우스 왼쪽 버튼을 누를 때 생성되며, WM_CHAR 메시지는 사용자가 문자를 입력할 때 발생되며, WM_CLOSE 메시지는 사용자가 윈도우를 닫을 때 발생된다. 모든 메시지는 커서 좌표값이나 키코드 값 등의 정보를 전달하는 두 개의 32비트 파라미터를 가지고 있다. WM_COMMAND 메시지는 사용자가 메뉴를 선택하거나, 다이얼로그 박스 버튼을 누를 때 해당 윈도우에게 보내진다. 명령 메시지 파라미터들은 윈도우의 메뉴 배치에 따라 달라진다. 또한 프로그래머가 "사용자 메시지"를 정의할 수도 있다.

3. Windows Graphics Device Interface (GDI)

MS-DOS 프로그램에서는 비디오 메모리나 프린트 포트에 직접적으로 쓰기를 한다. 이런 테크닉의 단점은 비디오 보드나 프린터 모델마다에 대

해서 드라이버 소프트웨어를 제공해주어야 한다는 것이다.

윈도우즈에서는 윈도우즈 그래픽 사용자 인터페이스(GDI)라고 하는 추상 층을 도입했다. 윈도우즈가 비디오 보드나 프린터를 위한 드라이버 소프트웨어를 제공해주기 때문에 프로그램에서는 시스템에 연결되어 있는 비디오 보드나 프린터의 타입에 관해서는 알지 못해도 상관없다. 프로그램에서는 하드웨어 주소를 사용하지 않고 대신에 디바이스 컨텍스트(device context))라는 데이터 구조체를 참조하는 GDI 함수를 호출한다. 윈도우즈는 디바이스 컨텍스트 객체를 물리적인 디바이스에 매핑(mapping)하고 해당 입력 명령들을 생성한다. GDI는 비디오를 직접 억세스하는 것만큼이나 빠르며, 윈도우즈용의 여러 응용 프로그램들이 표시 장치를 공유할 수 있게 해준다.

4. 리소스 기반 프로그래밍

MS-DOS에서 데이터 중심 프로그래밍을 하려면, 초기화 상수를 사용해서 데이터 코드를 작성하거나, 프로그램이 읽어들일 별도의 데이터 파일을 제공해 주어야 한다. 윈도우즈용 프로그래밍에 있어서는 만들어져 있는 여러 가지 포맷을 사용해서 데이터를 리소스 파일에 저장한다. 링커는 이 이진 리소스 파일과 C++ 컴파일러의 결과 파일을 조합해서 실행 프로그램을 만든다. 리소스 파일에는 비트맵, 아이콘, 메뉴 정의, 다이얼로그 박스, 문자열 등이 포함된다. 또한 이 파일에서는 프로그래머가 정의하는 주문형 리소스도 포함될 수 있다.

프로그램을 작성할 때 텍스트 에디터를 사용해도 상관은 없지만, 일반적으로는 WYSIWYG(What You See Is What You Get) 원리에 맞는 툴을 사용해서 리소스들을 편집하는 것이 좋다. 예를 들어, 다이얼로그 박스를 설계한다면, 아이콘들이 배열되어 있는 컨트롤 팔레트로부터 요소(버튼,

리스트 박스 등)를 선택한 다음 마우스를 사용해서 이 요소의 위치를 정하고 크기를 변경하게 된다. Visual C++ 그래픽 에디터를 이용하면 표준 리소스들을 효율적으로 편집할 수 있다.

5. 메모리 관리

새 버전의 윈도우즈를 사용하면 메모리 관리가 더 쉽다. 메모리 핸들이 잠금(lock)되거나 잡음이 나거나 하는 일들을 걱정하지 않아도 된다. 프로그래머는 원하는 대로 메모리를 맘껏 할당할 수 있고, 그러면 그에 따른 세부적인 일들은 윈도우즈가 알아서 해준다.

6. 동적 링크 라이브러리들(DLLs)

MS-DOS 환경에서는 프로그램 구축 시에 모든 객체 모듈이 동적으로 링크된다. 윈도우즈에서는 특수하게 만들어진 라이브러리들을 런타임시에 로드되어 링크될 수 있게 하는 동적 링크를 허용해준다. 복수의 응용 프로그램들이 동적 링크 라이브러리(DLLs)를 공유할 수 있으며, 그럼으로써 메모리와 디스크 소모량을 줄일 수 있다. DLL들은 별개 적으로 테스트할 수 있으므로, 동적 링크를 이용하면 프로그램의 모듈 성을 향상시킬 수 있다. 윈도우즈 설계자들은 DLL들을 C 언어와 함께 사용하기 위해 만들었으며, C++에서는 원래의 것을 계속해서 보완해왔다. MFC 개발자들은 모든 응용 프로그램 프레임워크 클래스들을 소수의 미리 만들어진 (ready-built) DLLs로 조합해 놓았다. 따라서 프로그래머들은 응용 프로그램 프레임워크 클래스들을 정적으로나 동적으로 자신의 응용 프로그램에 링크할 수 있게 된 것이다. 더군다나 프로그래머 자신이 MFC DLL들을 토대로 자신만의 고유한 DLL들을 만들 수도 있다.

7. Win32 Application Programming Interface (API)

초기의 윈도우즈 프로그래머들은 C 언어를 사용해서 Win16 응용 프로그래밍 인터페이스(API)를 위한 프로그램을 작성했었다. 요즘에는 32비트 응용 프로그램을 작성하기 위해 직접적으로든 간접적으로든 Win32 API를 사용한다. 대부분의 Win16 함수들은 동일한 기능의 Win32 함수를 가지고 있지만, 각 함수에 사용되는 파라미터들은 다르다. Win32 API는 새로운 함수들을 많이 제공해주는데, 그 중에는 초기에는 MS-DOS 호출을 이용해 처리되던 디스크 I/O 함수들도 포함되어 있다. 16비트 Visual C++를 사용하는 MFC 프로그래머의 경우에는 Win16이나 Win32 어느 것에나 함께 사용할 수 있도록 되어 있는 MFC 표준으로 작성하기 때문에 프로그래머가 이러한 API 차이점들에 접근할 수 없었다.

8. Visual C++ 구성 요소들

Visual C++는 두 개의 완전한 윈도우즈 응용 프로그램 개발 시스템이 한 제품으로 통합된 것이다. 따라서 이 제품을 사용하는 경우에는 C-언어 윈도우즈 프로그램을 Win32 API를 이용해서 작성할 수 있다. Charles Petzold의 "Programming Windows 3.1"(Microsoft Press, 1992) 등을 포함해서 Win16에 관한 많은 책들이 Win32에 관한 내용으로 교체되고 있다. (이 책의 새 제목은 "Programming Windows 95"(Microsft Press, 1996)이다.) 그래픽 에디터 등을 포함한 여러 가지 Visual C++ 툴들을 이용하면 저 수준의 Win32 프로그래밍을 좀더 쉽게 프로그래밍을 할 수 있을 뿐만 아니라 손쉬운 디버깅을 할 수 있다.

9. Microsoft Developer Studio와 프로그램을 만드는 과정

Developer Studio는 Windows용 통합 개발 환경(IDE)으로서 Visual C++, Microsoft FORTRAN, 그외 일부 제품들과 공유된다. 이 IDE는 Windows용 Quick C를 토대로 하던 원래의 Visual Workbench로부터 발전된 개발 환경이다. 현재의 개발 환경에는 도킹 윈도우, 툴바, 매크로들을 실행하는 최적화 가능한 에디터 등이 포함되어 있다.

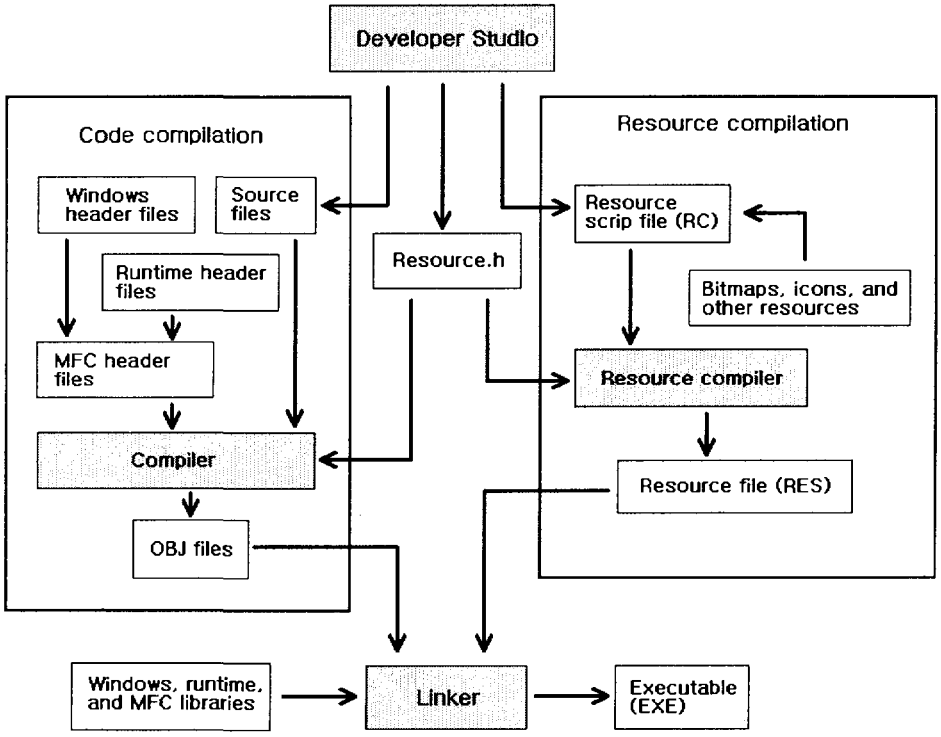

그림 1. Visual C++ 프로그래밍

내장 도움말 시스템은 Microsoft Developer Network(MSDN)에 사용되던 기술을 기초로 한 도움말 시스템으로서, 예전의 윈도우즈에 포함되어 있는 도움말 뷰어(Viewer)인 WINHELP보다 훨씬 더 강력하다.

Visual C++는 프로젝트 단위로 관리된다. 프로젝트란 서로 연관되어 있는 파일들의 집합으로서 이 파일들이 컴파일되고 링크되어서 Windows 기반 실행 프로그램이나 DLL을 만들게 된다. 프로젝트 소스 파일들은 일반적으로 별도의 서브디렉토리에 저장된다. 또한 프로젝트는 인클루드 파일이나 라이브러리 등과 같이 프로젝트 서브디렉토리 외부에 존재하는 파일들과도 연관성을 갖는다. Visual C++에서 각 프로젝트마다 에는 메이커 파일(확장자 .DAK)과 프로젝트 Workspace 파일(확장자 .ASW(v5.0), .PDP(v4.x))이 포함된다.

제 2 절. 로보트 제어를 위한 네트워크 통신 프로그램

1. OSI 7-LAYER의 구조

1980년대 초에 ISO는 업체들이 네트워크를 구현할 때 참조로 할 네트워크 모델의 필요성을 인식하고 1984년 OSI 참조 모델을 발표하게 되었다. OSI 참조 모델은 응용 프로그램의 정보가 네트워크 매체를 통해 다른 컴퓨터의 응용 프로그램에 어떻게 전달되는가를 설명한다.

통신 기능은 일련의 계층 집단으로 분할되며, 각 계층은 다른 시스템과 통신하려는 데에 필요한 관련된 기능을 수행한다. 각 계층은 그 기능들의 세부 내용을 은폐하고 보다 원시적인 기능을 수행하기 위하여 바로 아래에 있는 계층에 의존한다.

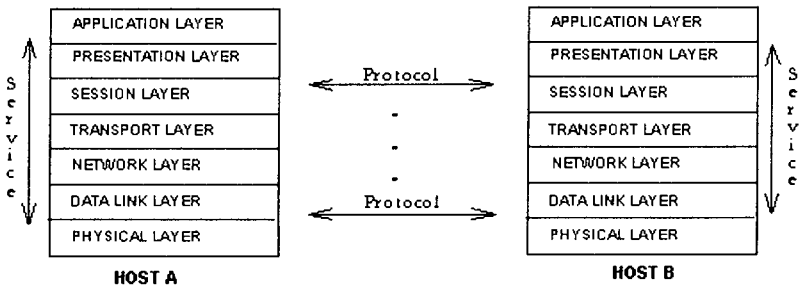

그림 2. OSI 7-Layer 개념도

○ 1 계층 : 물리적 계층(Physical Layer)

물리적 계층은 시스템간에 물리적 링크를 작동시키거나 유지하며 전기, 기계, 절차 그리고 기능적 측면의 문제들을 정의한다.

○ 2 계층 : 데이터 링크 층(데이터-link Layer)

데이터 링크 층은 물리적 링크를 통한 신뢰성 있는 데이터 전송을 제

공한다. 이 계층은 물리적 어드레싱, 네트워크 토폴로지, 회선 사용 규칙, 오류 검출, 프레임 전달 그리고 흐름 제어 등에 관계한다.

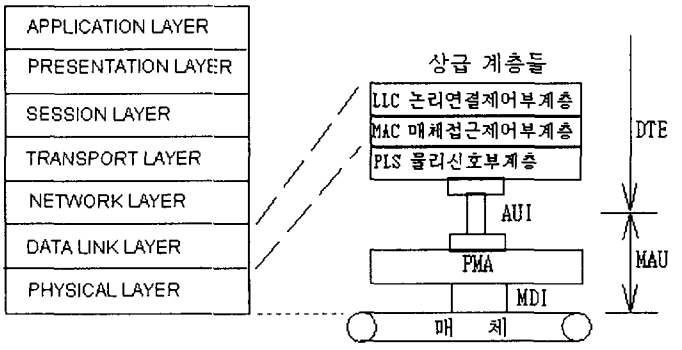

그림 3. 데이터 링크/물리적 계층의 개념도

○ 3 계층 : 네트워크 층(Network Layer)

네트워크 층은 다른 장소에 위치한 두 시스템간에 연결성과 경로 선택을 제공한다. 라우팅 프로토콜이 서로 연결된 네트워크를 통한 최적 경로를 선택하며 네트워크 층의 프로토콜은 선택된 경로를 따라 정보를 보낸다.

○ 4 계층 : 트랜스포트 층(Transport Layer)

애플리케이션, 프리젠테이션, 그리고 세션 층이 애플리케이션에 관련되어 있다면 하위의 네 계층은 데이터 전송에 관련되어 있다. 트랜스포트 층은 데이터 전송 서비스를 제공하는 층이다. 즉 인터네트워크 상에서 얼마나 신뢰성 있는 데이터 전송이 이루어지는가 등의 문제에 트랜스포트 층이 관련되어 있다. 신뢰성 있는 서비스를 제공하기 위해

트랜스포트 층은 가상 회로의 구축, 유지 및 종료, 전송 오류 검출 및 복구 그리고 정보 흐름 제어의 절차를 제공한다.

○ 5 계층 : 세션 층(Session Layer)

세션 층은 애플리케이션간에 세션을 구축하고 관리하며 종료시키는 역할을 하는 층이다. 세션 층은 프리젠테이션 층 사이의 대화를 동기 시키며 데이터 교환을 관리한다.

○ 6 계층 : 프리젠테이션 층(Presentation Layer)

프리젠테이션 층은 한 시스템의 애플리케이션에서 보낸 정보를 다른 시스템의 애플리케이션 층이 읽을 수 있도록 하는 층이다.

○ 7 계층 : 애플리케이션 층(Application Layer)

애플리케이션 층은 OSI 모델에서 유저와 가장 가까운 층이다. 이 계층은 OSI의 다른 어떤 계층에도 서비스를 제공하지 않는다는 점에서 다른 계층과 다르다. 이 계층에는 스프레드 쉬트, 워드 프로세싱 등이 속한다.

TCP/IP 모델은 현재 네트워크 망에서 가장 널리 사용되고 있는 네트워크 프로토콜로서, 위에서 보인 OSI 7-Layer에 해당하는 TCP/IP의 모델은 다음의 그림과 같다. 그림의 오른편에는 TCP/IP 모델에서의 데이터의 가공 단계를 보여주고 있다.

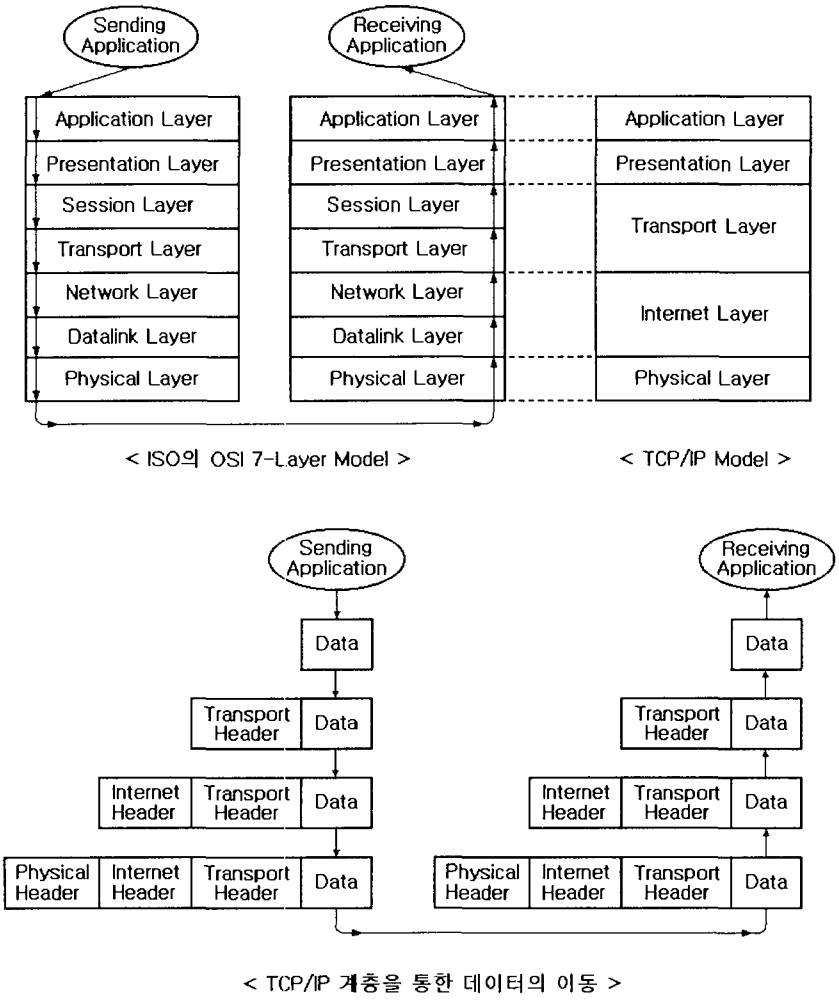

그림 4. TCP/IP의 모델

2. TCP/IP 프로토콜을 이용한 네트워크 프로그램

원전 1차 계통 고방사선지역 점검 보수용 이동 로보트인 KAEROT/m1은 다음 그림과 같이 크게 나누어 중앙 제어부와 원격 로보트 부분으로 나눌 수 있다.

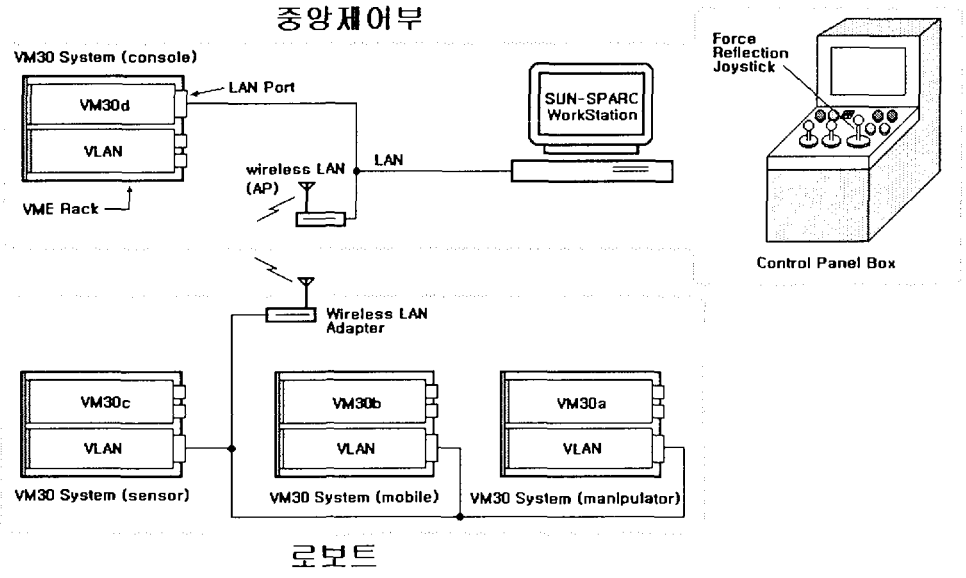

그림 5. 이동 로보트의 통신 개념도

이 둘 간의 물리적 연결은 IEEE 402.3 규약을 만족하는 무선 네트워크 통신을 하게 된다. 무선 네트워크 통신에서 주(主)가 되는 AP(Access Point)와 이동 로보트에 실리는 네트워크 어댑터 사이에서는 앞 절에서 설명한 TCP/IP 프로토콜을 사용하여 통신을 하게 된다.

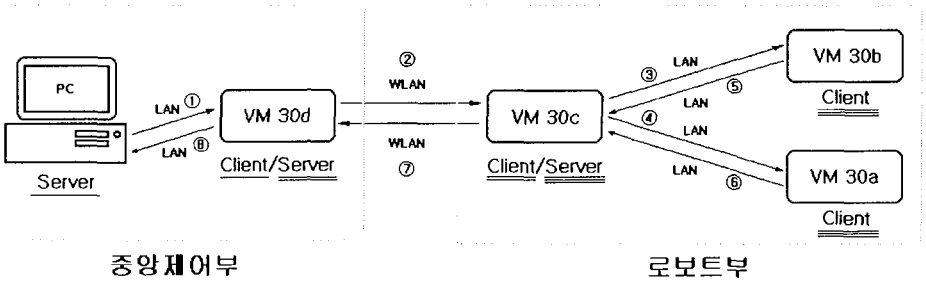

그림 6. 통신 흐름 순서

여기서 시스템의 전체 서버(Server) 역할은 중앙 제어부를 담당하는 윈도우즈를 운영체제로 하는 PC가 되며, VM30d, VM30c는 중간에서 서버와 클라이언트(Client) 역할을 동시에 하게 된다.

이와 같은 서버/클라이언트 모델을 TCP/IP 프로토콜에 맞춰 코딩하는 순서는 다음과 같다. 이 과정은 유닉스에서의 C/C++ 컴파일러, Visual C++에서 소켓 프로그래밍 양식으로 같은 함수명들이 사용된다.

우선 서버측에서는 소켓을 생성한 후, 바인딩하여 생성된 소켓에 이름을 부여하게 된다. 그런 다음 클라이언트의 연결을 기다리는 리스닝 함수를 호출한 후, 대기 상태에 들어간다. 한편 클라이언트 역시 소켓을 이용한 네트워크 통신을 하기 위해 소켓을 생성한다. 그런 다음, connect 함수를 호출하여 서버에 접속을 시도한다. 이 때 서버측 프로그램에서는 이러한 클라이언트의 connect 함수에 대응하여 클라이언트를 받아들이는 accept 함수를 실행하게 된다. 이렇게 연결이 되고 나면, 서버와 클라이언트는 recv, send 함수를 사용하여 데이터 통신을 하게 된다. 그런 다음 통신 과정이 끝나면 서버와 클라이언트는 close 함수로 맨처음 생성해놓았던 소켓을 닫음으로써 네트워크 통신은 끝나게 된다.

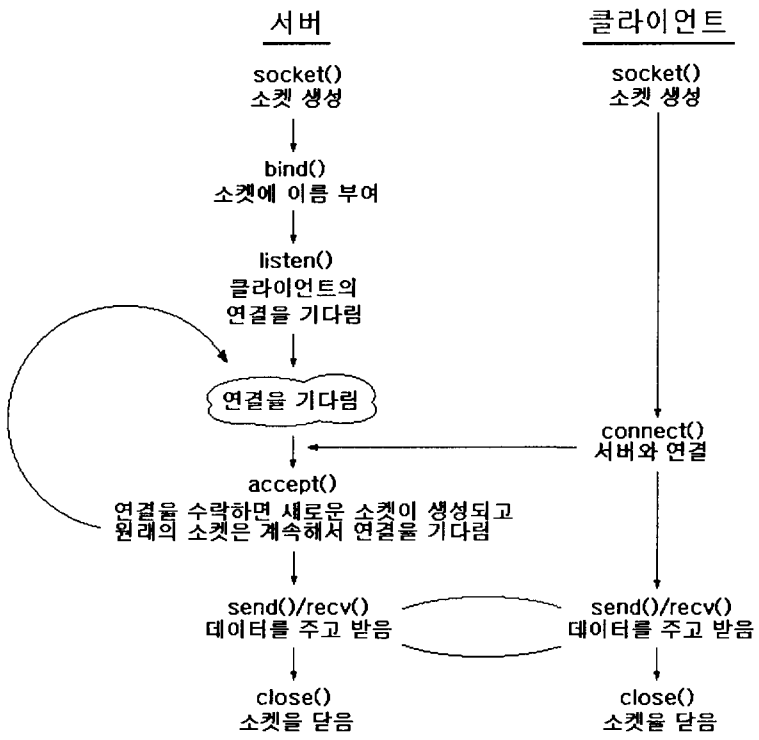

그림 7. 서버/클라이언트의 연결 및 동작 순서

다음은 UNIX/VxWorks 운영 체제 간의 네트워크 서버의 소스 코드이다. 여기서는 위에 보인 서버/클라이언트 연결에서의 서버 동작 순서를 보여주고 있다. 다음에 보이는 유닉스 서버 코드에서는 클라이언트로부터 1 문자를 전송 받아 화면에 보여주는 역할을 한다.

```
/* server.c - server process for simple client/server network demo */
static char *copyright = "Copyright 1986-1989, Wind River Systems, Inc.";
/*
modification history
```

```
---------------------

01i,31oct91,rrr  passed through the ansification filter
                        -changed copyright notice
01h,25oct90,lpf  put back the HP version's bcopy.
01g,07jun89,gae  used ntohs on SERVER_NUM; changed SOCKADDR to struct sockaddr.
01c,06apr87,gae  caught some lint.  Got rid of excess includes.
01b,12jan87,jlf  minor rearrangement, to meet WRS coding conventions.
                        changed <> to " in includes.
01a,17sep86,llk  written.
*/
/*
DESCRIPTION
This is a simple demonstration of the server-client relationship.
This is the server. The other half of the demonstration is in client.c
*/

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include "serverDemo.h"

#if     defined(HOST_HP)
void bzero (s, n)
    char *s;
    int n;
    {
    memset (s, '\0', n);
    }
#endif

/*******************************************************************************
*
* main - server process
*
* This is a simple server program which communicates with a client
```

```
* through a socket.  It reads (recv's) characters, one at a time,
* from the client and echos them to standard output.  When the client
* goes away, the server also goes away (the server reads 0 characters
* from the socket).
*
* The server runs on UNIX, client runs on VxWorks.
*
* See the manual page on sockets for more information.
*/
main ()
    {
    int                     sock, snew;         /* socket fd's */
    struct sockaddr_in      serverAddr;         /* server's address */
    struct sockaddr_in      clientAddr;         /* client's address */
    int                     client_len;         /* length of clientAddr */
    char                    c;
    extern int              errno;              /* for UNIX error referencing */

    /* Zero out the sock_addr structures.
     * This MUST be done before the socket calls.
     */
    bzero (&serverAddr, sizeof (serverAddr));
    bzero (&clientAddr, sizeof (clientAddr));

    /* Open the socket.
     * Use ARPA Internet address format and stream sockets.
     * Format described in "socket.h".
     */
    sock = socket (AF_INET, SOCK_STREAM, 0);
    if (sock == -1)
          exit (1);

    /* Set up our internet address, and bind it so the client can connect. */
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(SERVER_NUM);
```

```
printf ("\nBinding SERVER\n", serverAddr.sin_port);
if (bind (sock, (struct sockaddr *)&serverAddr, sizeof (serverAddr)) == -1)
        {
        printf ("bind failed, errno = %d\n", errno);
        close (sock);
        exit (1);
        }

/* Listen, for the client to connect to us. */
printf ("Listening to client\n");
if (listen (sock, 2) == -1)
        {
        printf ("listen failed\n");
        close (sock);
        exit (1);
        }

/* The client has connected.  Accept, and receive chars */
printf ("Accepting CLIENT\n");
client_len = sizeof (clientAddr);
snew = accept (sock, (struct sockaddr *)&clientAddr, &client_len);
if (snew == -1)
        {
        printf ("accept failed\n");
        close (sock);
        exit (1);
        }
printf ("CLIENT: port = %d: family = %d: addr = %lx:\n",
                ntohs(clientAddr.sin_port), clientAddr.sin_family,
                ntohl(clientAddr.sin_addr.s_addr));

/* repeatedly recieve characters from client and put on stdout */
for (;;)
        {
```

```
        if (recv (snew, &c, 1, 0) == 0)
            {
            /* client has disappeared */
            break;
            }
        putchar (c);
        }

    /* close the socket from the UNIX side */
    close (sock);
    close (snew);
    printf ("\n...goodbye\n");
    }
```

다음은 VxWorks 운영 체제 하에서의 네트워크 클라이언트의 소스 코드이다. 여기서는 위에 보인 서버/클라이언트 연결에서의 서버의 동작 순서를 보여주고 있다. 이 클라이언트 코드에서는 서버 측에 1 문자를 전송하는 역할을 한다.

```
/* client.c - client task for simple client/server network demo */

/* Copyright 1984-1993 Wind River Systems, Inc. */
#include "copyright_wrs.h"

/*
DESCRIPTION
This is a simple demonstration of the server-client relationship.
This is the client. The other half of the demonstration is in server.c

EXAMPLE
```

```
On the UNIX host "wrs":
    % server

On the VxWorks target:
    -> ld < client.o
    -> client "wrs"

When you type characters into VxWorks, they will be echoed by the server
on UNIX.  Type control-D to terminate.
*/

#include "vxWorks.h"
#include "fioLib.h"
#include "stdio.h"
#include "unistd.h"
#include "string.h"
#include "usrLib.h"
#include "errnoLib.h"
#include "hostLib.h"
#include "sockLib.h"
#include "socket.h"
#include "inetLib.h"
#include "in.h"
#include "serverDemo.h"

IMPORT char sysBootHost [];     /* VxWorks saves name of host booted from */

/* make 'clientSock' a global variable so that if something goes wrong
 * when the client runs on VxWorks, the socket can be closed.
 */
int clientSock;             /* socket opened to server */


/*******************************************************************************
*
```

```
* client - client task
*
* This is a simple client program which connects to the server
* via a socket.  It reads characters from standard input and sends
* them onto the server through the socket.  It stops when it reads
* 0 characters (receives an EOF).
*
* The server runs on UNIX, client runs on VxWorks.
*
* RETURNS: OK or ERROR
*/

STATUS client (hostName)
    char *hostName; /* name of host running server, 0=boot host */
    {
    struct sockaddr_in  serverAddr;      /* server's address */
    struct sockaddr_in  clientAddr;      /* client's address */
    int         nBytes;                  /* number of bytes read from stdin */
    char        c;

    if (hostName == NULL)
        hostName = sysBootHost;

    /* Zero out the sock_addr structures.
     * This MUST be done before the socket call.
     */

    bzero ((char *) &serverAddr, sizeof (serverAddr));
    bzero ((char *) &clientAddr, sizeof (clientAddr));

    /* Open the socket.
     * Use ARPA Internet address format and stream sockets.
     * Format described in "socket.h".
     */
```

```
clientSock = socket (AF_INET, SOCK_STREAM, 0);

if (clientSock == ERROR)
    return (ERROR);

serverAddr.sin_family = AF_INET;
serverAddr.sin_port   = htons(SERVER_NUM);

/* get server's Internet address */

if ((serverAddr.sin_addr.s_addr = inet_addr (hostName)) == ERROR &&
    (serverAddr.sin_addr.s_addr = hostGetByName (hostName)) == ERROR)
    {
    printf ("Invalid host: \"%s\"\n", hostName);
    printErrno (errnoGet ());
    close (clientSock);
    return (ERROR);
    }

printf ("Server's address is %x:\n", htonl (serverAddr.sin_addr.s_addr));

if (connect (clientSock, (struct sockaddr *)&serverAddr,
                sizeof (serverAddr)) == ERROR)
    {
    printf ("Connect failed:\n");
    printErrno (errnoGet ());
    close (clientSock);
    return (ERROR);
    }

printf ("Connected...\n");

/* repeatedly read from standard input and send to socket until EOF */

while (TRUE)
```

```
        {
        /* read a character from standard input */

/*      if ((nBytes = read (STD_IN, &c, 1)) == 0)
            {
             client read an EOF; exit the loop.
            break;
            }
        else if (nBytes != 1)
            {
            printf ("CLIENT read error, %d bytes read\n", nBytes);
            printErrno (errnoGet ());
            break;
            } */

        /* send byte to server */

        if (send (clientSock, "demo" /* &c */ , 1, 0) != 1)
            {
            printf ("CLIENT write error:\n");
            printErrno (errnoGet ());
            }
        }

    /* close socket from the VxWorks side */

    close (clientSock);

    printf ("\n...goodbye\n");
    return (OK);
    }
```

다음은 UNIX나 VxWorks, 혹은 기타 플랫폼간에서의 Windows 95 환경에

서의 클라이언트 클래스의 소스 코드이다. 비동기 소켓 통신을 지원하는, MFC 내의 클래스인 CAsyncSocket에서 계승받아 클래스를 생성하였다.

```
////////////////////////////////////////////////////////////////////////////
// clientsock.h
//클라이언트용 소켓 프로그램 헤더

// 사용자 메시지 정의
#define WM__RECEIVE__DATA WM__USER+2

class CClientSock : public CAsyncSocket
{
public:
        CClientSock();
        void SetWnd(HWND hwnd);       // 메시지를 전달할 HWND를 설정하는 함수
        virtual void OnReceive( int nErrorCode );  // 새로운 데이터가 들어왔을 때 실행되는 함수
public:
        HWND m__pHwnd;    // 메시지를 전달할 HWND
};
////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////
//Clientsok.cpp
//클라이언트용 소켓 클래스 함수
#include "stdafx.h"
#include "ClientSok.h"

// 생성자는 아무일도 안 하고 모든 생성을 CAsyncSocket에 넘겨준다.
CClientSock::CClientSock()
{
        CAsyncSocket::CAsyncSocket();
}
// 데이터가 들어왔을 때 현재 hwnd에 새로운 데이터가 들어
```

```
// 왔다는 메시지를 날려 주기 위한 hwnd를 설정한다.
void CClientSock::SetWnd(HWND hwnd)
{
        m_pHwnd=hwnd;
}

// 새로운 데이터가 들어왔을 때 현재의 hwnd에
// USER+2인 메시지 WM_RECEIVE를 보낸다.
void CClientSock::OnReceive( int nErrorCode )
{
        TRACE("Errocode = %d",nErrorCode);
        SendMessage(m_pHwnd,WM_RECEIVE_DATA,0,0);
}
////////////////////////////////////////////////////////////////////////////
```

서버 클래스는 제일 먼저 클라언트를 기다린 다음 새로운 클라이언트가 오면 클라이언트와 같은 클래스를 만들어서 링크시켜 주면 된다. 다음의 코드는 Windows 환경 하에서의 서버 프로그램의 코드를 보여주고 있다.

```
////////////////////////////////////////////////////////////////////
// serversock.h
// 서버용 메인소켓 프로그램

#include "clientsock.h"
#define WM_ACCEPT_SOCKET WM_USER+1

class CServerSock : public CAsyncSocket
{
public:
        CSeverSock();
```

```
        void SetWnd(HWND hwnd);       // 메시지를 전달할 HWND 설정
        CClientSock* GetAcceptSocket(); // 현재 클라이언트와 연결된 소켓을 받는다.

        virtual void OnAccept( int nErrorCode );    //새로운 클라이언트가 연결되었을 때
public:

        CClientSock m_pChild; // 새로운 클라이언트가 연결되었을 때 Accept할 소켓 클래스 변수
        HWND m_pHwnd;
};
/////////////////////////////////////////////////////////////////////

/////////////////////////////////////////////////////////////////////
// ServerSock.cpp
// 서버 클래스 함수 소스
#include "stdafx.h"
#include "serversok.h"

// 생성자
CSeverSock::CSeverSock()
{
        CAsyncSocket::CAsyncSocket();
}

// 메시지를 전달할 윈도우 설정
void CSeverSock::SetWnd(HWND hwnd)
{
        m_pHwnd=hwnd;
}

// 새로운 클라이언트가 연결되었을 때
// 실행되는 함수
void CSeverSock::OnAccept( int nErrorCode )
{
        TRACE("Errocode = %d",nErrorCode);
        Accept(m_pChild);   //클라이언트 클래스인 m_pChild로 연결시켜 준다.
```

```
        // 새로운 클라이언트가 연결되었다는
        // WM_ACCEPT_SOCKET 메시지를 전달한다.
        SendMessage(m_pHwnd,WM_ACCEPT_SOCKET,0,0);
}

// OnAccept 함수 실행 후 새로운 클라이언트와 연결된
// m_pChild 소켓을 넘겨준다.
CClientSock* CSeverSock::GetAcceptSocket()
{
        return &m_pChild;
}
//////////////////////////////////////////////////////////////////////////
```

위의 서버/클라이언트 소스 코드는 원하는 클래스 내에 포인터 내지 변수로 삽입시켜 이벤트 처리를 하면 소켓 통신 프로그램이 완성된다.

다음 소스 코드는 위의 클라이언트 소스 코드를 이용하여 작성한 클라이언트 프로그램의 작성 예이다. 서버와의 정해진 스트럭쳐의 구조를 전송하는 것으로, 뷰 클래스는 CFormView를 사용하였다.

```
// ManiTestView.h : interface of the CManiTestView class
//
//////////////////////////////////////////////////////////////////////////

#if
!defined(AFX_MANITESTVIEW_H___33AAEE41_90DC_11D1_A405_0000E80D6B74___INCLUDED_)
#define AFX_MANITESTVIEW_H___33AAEE41_90DC_11D1_A405_0000E80D6B74___INCLUDED_

#include "ClientSocket.h"        // Added by ClassView
```

```
#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

class CManiTestView : public CFormView
{
protected: // create from serialization only
        CManiTestView();
        DECLARE_DYNCREATE(CManiTestView)

public:
        //{{AFX_DATA(CManiTestView)
        enum { IDD = IDD_MANITEST_FORM };
        BYTE    m_bReceiveData0;
        BYTE    m_bReceiveData1;
        BYTE    m_bReceiveData2;
        BYTE    m_bReceiveData3;
        BYTE    m_bReceiveData4;
        BYTE    m_bReceiveData5;
        BYTE    m_bSendData0;
        BYTE    m_bSendData1;
        BYTE    m_bSendData2;
        BYTE    m_bSendData3;
        BYTE    m_bSendData4;
        BYTE    m_bSendData5;
        BYTE    m_bSendData6;
        BYTE    m_bReceiveStatus;
        CString m_sMonitor;
        //}}AFX_DATA

// Attributes
public:
        CManiTestDoc* GetDocument();

// Operations
```

```
public:

// Overrides
        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CManiTestView)
        public:
        virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
        virtual void OnInitialUpdate();
        protected:
        virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
        virtual BOOL OnPreparePrinting(CPrintInfo* pInfo);
        virtual void OnBeginPrinting(CDC* pDC, CPrintInfo* pInfo);
        virtual void OnEndPrinting(CDC* pDC, CPrintInfo* pInfo);
        virtual void OnPrint(CDC* pDC, CPrintInfo*);
        //}}AFX_VIRTUAL

// Implementation
public:
        CClientSocket *m_pClientSocket;
        virtual ~CManiTestView();
#ifdef _DEBUG
        virtual void AssertValid() const;
        virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Generated message map functions
protected:
        //{{AFX_MSG(CManiTestView)
        afx_msg void OnSend();
        afx_msg LONG OnReceive(UINT, LONG);
        afx_msg void OnLetsConnect();
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
```

```
};

#ifndef __DEBUG  // debug version in ManiTestView.cpp
inline CManiTestDoc* CManiTestView::GetDocument()
   { return (CManiTestDoc*)m_pDocument; }
#endif

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately before the previous line.

#endif //
!defined(AFX_MANITESTVIEW_H___33AAEE41_90DC_11D1_A405_0000E80D6B74___INCLUDED_)

// ManiTestView.cpp : implementation of the CManiTestView class
//

#include "stdafx.h"
#include "ManiTest.h"

#include "ManiTestDoc.h"
#include "ManiTestView.h"

#ifdef __DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

#define SERVER_ADDR          "147.43.74.56"
//#define SERVER_ADDR        "147.43.74.54"
#define PORT_NUM  1100

// definition of mani_command
```

```
#define CONSOLE_MODE        1
#define MASTER_MODE         2
#define DRIVER_CLOSE_MODE   5
#define RELAY_MODE          6

struct {
        BYTE mani_command;
        BYTE mani_move[7];
} PC_to_vm30;

struct {
        BYTE mani_status;
        BYTE mani_move[6];
} PC_from_vm30;

/////////////////////////////////////////////////////////////////////////////
// CManiTestView

IMPLEMENT_DYNCREATE(CManiTestView, CFormView)

BEGIN_MESSAGE_MAP(CManiTestView, CFormView)
        //{{AFX_MSG_MAP(CManiTestView)
        ON_BN_CLICKED(IDC_SEND, OnSend)
        ON_BN_CLICKED(IDC_CONNECT, OnLetsConnect)
        //}}AFX_MSG_MAP
        // Standard printing commands
        ON_COMMAND(ID_FILE_PRINT, CFormView::OnFilePrint)
        ON_COMMAND(ID_FILE_PRINT_DIRECT, CFormView::OnFilePrint)
        ON_COMMAND(ID_FILE_PRINT_PREVIEW, CFormView::OnFilePrintPreview)
        ON_MESSAGE(WM_RECEIVE_DATA,OnReceive)       // added
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CManiTestView construction/destruction
```

```
CManiTestView::CManiTestView()
	: CFormView(CManiTestView::IDD)
{
	//{{AFX_DATA_INIT(CManiTestView)
	m_bReceiveData0 = 0;
	m_bReceiveData1 = 0;
	m_bReceiveData2 = 0;
	m_bReceiveData3 = 0;
	m_bReceiveData4 = 0;
	m_bReceiveData5 = 0;
	m_bSendData0 = 0;
	m_bSendData1 = 0;
	m_bSendData2 = 0;
	m_bSendData3 = 0;
	m_bSendData4 = 0;
	m_bSendData5 = 0;
	m_bSendData6 = 0;
	m_bReceiveStatus = 0;
	m_sMonitor = _T("");
	//}}AFX_DATA_INIT
	// TODO: add construction code here

}

CManiTestView::~CManiTestView()
{
	delete m_pClientSocket;
}

void CManiTestView::DoDataExchange(CDataExchange* pDX)
{
	CFormView::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CManiTestView)
	DDX_Text(pDX, IDC_EDIT_MOVE_R0, m_bReceiveData0);
	DDV_MinMaxByte(pDX, m_bReceiveData0, 0, 255);
```

```
	DDX_Text(pDX, IDC_EDIT_MOVE_R1, m_bReceiveData1);
	DDV_MinMaxByte(pDX, m_bReceiveData1, 0, 255);
	DDX_Text(pDX, IDC_EDIT_MOVE_R2, m_bReceiveData2);
	DDV_MinMaxByte(pDX, m_bReceiveData2, 0, 255);
	DDX_Text(pDX, IDC_EDIT_MOVE_R3, m_bReceiveData3);
	DDV_MinMaxByte(pDX, m_bReceiveData3, 0, 255);
	DDX_Text(pDX, IDC_EDIT_MOVE_R4, m_bReceiveData4);
	DDV_MinMaxByte(pDX, m_bReceiveData4, 0, 255);
	DDX_Text(pDX, IDC_EDIT_MOVE_R5, m_bReceiveData5);
	DDV_MinMaxByte(pDX, m_bReceiveData5, 0, 255);
	DDX_Text(pDX, IDC_EDIT_MOVE_S0, m_bSendData0);
	DDV_MinMaxByte(pDX, m_bSendData0, 0, 255);
	DDX_Text(pDX, IDC_EDIT_MOVE_S1, m_bSendData1);
	DDV_MinMaxByte(pDX, m_bSendData1, 0, 255);
	DDX_Text(pDX, IDC_EDIT_MOVE_S2, m_bSendData2);
	DDV_MinMaxByte(pDX, m_bSendData2, 0, 255);
	DDX_Text(pDX, IDC_EDIT_MOVE_S3, m_bSendData3);
	DDV_MinMaxByte(pDX, m_bSendData3, 0, 255);
	DDX_Text(pDX, IDC_EDIT_MOVE_S4, m_bSendData4);
	DDV_MinMaxByte(pDX, m_bSendData4, 0, 255);
	DDX_Text(pDX, IDC_EDIT_MOVE_S5, m_bSendData5);
	DDV_MinMaxByte(pDX, m_bSendData5, 0, 255);
	DDX_Text(pDX, IDC_EDIT_MOVE_S6, m_bSendData6);
	DDV_MinMaxByte(pDX, m_bSendData6, 0, 255);
	DDX_Text(pDX, IDC_EDIT_STATUS, m_bReceiveStatus);
	DDV_MinMaxByte(pDX, m_bReceiveStatus, 0, 255);
	DDX_Text(pDX, IDC_MONITOR, m_sMonitor);
	//}}AFX_DATA_MAP
}

BOOL CManiTestView::PreCreateWindow(CREATESTRUCT& cs)
{
	// TODO: Modify the Window class or styles here by modifying
	//  the CREATESTRUCT cs
```

```
	return CFormView::PreCreateWindow(cs);
}

/////////////////////////////////////////////////////////////////////////////
// CManiTestView printing

BOOL CManiTestView::OnPreparePrinting(CPrintInfo* pInfo)
{
	// default preparation
	return DoPreparePrinting(pInfo);
}

void CManiTestView::OnBeginPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
	// TODO: add extra initialization before printing
}

void CManiTestView::OnEndPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
	// TODO: add cleanup after printing
}

void CManiTestView::OnPrint(CDC* pDC, CPrintInfo*)
{
	// TODO: add code to print the controls
}

/////////////////////////////////////////////////////////////////////////////
// CManiTestView diagnostics

#ifdef __DEBUG
void CManiTestView::AssertValid() const
{
	CFormView::AssertValid();
}
```

```
void CManiTestView::Dump(CDumpContext& dc) const
{
	CFormView::Dump(dc);
}

CManiTestDoc* CManiTestView::GetDocument() // non-debug version is inline
{
	ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CManiTestDoc)));
	return (CManiTestDoc*)m_pDocument;
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CManiTestView message handlers

void CManiTestView::OnInitialUpdate()
{
	CFormView::OnInitialUpdate();

	// TODO: Add your specialized code here and/or call the base class

	// Resize and fit to dialog size
	GetParentFrame()->RecalcLayout();
	ResizeParentToFit( /*FALSE*/ );

	// set check radio button (initialize)
	((CButton *)GetDlgItem(IDC_RADIO_CONSOLE))->SetCheck(1);

	// create socket object
	m_pClientSocket = new CClientSocket;

	// Socket creation
	m_sMonitor += "Creating socket...";
	((CEdit *)GetDlgItem(IDC_MONITOR))->SetSel(-1);
```

```
        SetDlgItemText(IDC_MONITOR, m_sMonitor);
        if (m_pClientSocket->Create(PORT_NUM) != 0)
                m_sMonitor += "O.K.\r\n";
        else
                m_sMonitor += "Failed.\r\n";
        ((CEdit *)GetDlgItem(IDC_MONITOR))->SetSel(-1);
        SetDlgItemText(IDC_MONITOR, m_sMonitor);

        // set HWND to receive data
        m_pClientSocket->SetWnd(this->m_hWnd);
}

void CManiTestView::OnLetsConnect()
{
        // TODO: Add your control notification handler code here

        // Connect to server
        m_sMonitor += "Connect to server...";
        ((CEdit *)GetDlgItem(IDC_MONITOR))->SetSel(-1);
        SetDlgItemText(IDC_MONITOR, m_sMonitor);

        int i = m_pClientSocket->Connect(SERVER_ADDR, PORT_NUM);
//      TRACE("**************************\n");
//      TRACE(" Connect() result is %d\n", i);
//      TRACE("**************************\n");
//      TRACE("GetLastError() is %d\n", GetLastError());

        if (i != 0)
                m_sMonitor += "O.K.\r\n";
        else
                m_sMonitor += "Failed.\r\n";

        ((CEdit *)GetDlgItem(IDC_MONITOR))->SetSel(-1);
        SetDlgItemText(IDC_MONITOR, m_sMonitor);
}
```

```
void CManiTestView::OnSend()
{
	// TODO: Add your control notification handler code here
	UpdateData();

	if (((CButton *)GetDlgItem(IDC_RADIO_CONSOLE))->GetCheck() == 1)
		PC_to_vm30.mani_command = CONSOLE_MODE;
	else if (((CButton *)GetDlgItem(IDC_RADIO_MASTER))->GetCheck() == 1)
		PC_to_vm30.mani_command = MASTER_MODE;
	else if (((CButton *)GetDlgItem(IDC_RADIO_DRIVER))->GetCheck() == 1)
		PC_to_vm30.mani_command = DRIVER_CLOSE_MODE;
	else if (((CButton *)GetDlgItem(IDC_RADIO_RELAY))->GetCheck() == 1)
		PC_to_vm30.mani_command = RELAY_MODE;

	PC_to_vm30.mani_move[0] = m_bSendData0;
	PC_to_vm30.mani_move[1] = m_bSendData1;
	PC_to_vm30.mani_move[2] = m_bSendData2;
	PC_to_vm30.mani_move[3] = m_bSendData3;
	PC_to_vm30.mani_move[4] = m_bSendData4;
	PC_to_vm30.mani_move[5] = m_bSendData5;
	PC_to_vm30.mani_move[6] = m_bSendData6;

	m_pClientSocket->Send(&PC_to_vm30, sizeof(BYTE) * 8);
}

LONG CManiTestView::OnReceive(UINT wParam, LONG lParam)
{
	m_pClientSocket->Receive(&PC_from_vm30, sizeof(BYTE) * 8);

	CButton* btn = (CButton *)GetDlgItem(IDC_EDIT_STATUS);
	SetDlgItemInt(IDC_EDIT_STATUS, PC_from_vm30.mani_status, FALSE);

	btn = (CButton *)GetDlgItem(IDC_EDIT_MOVE_R0);
	SetDlgItemInt(IDC_EDIT_MOVE_R0, PC_from_vm30.mani_move[0], FALSE);
```

```
    btn = (CButton *)GetDlgItem(IDC_EDIT_MOVE_R1);
    SetDlgItemInt(IDC_EDIT_MOVE_R1, PC_from_vm30.mani_move[1], FALSE);
    btn = (CButton *)GetDlgItem(IDC_EDIT_MOVE_R2);
    SetDlgItemInt(IDC_EDIT_MOVE_R2, PC_from_vm30.mani_move[2], FALSE);
    btn = (CButton *)GetDlgItem(IDC_EDIT_MOVE_R3);
    SetDlgItemInt(IDC_EDIT_MOVE_R3, PC_from_vm30.mani_move[3], FALSE);
    btn = (CButton *)GetDlgItem(IDC_EDIT_MOVE_R4);
    SetDlgItemInt(IDC_EDIT_MOVE_R4, PC_from_vm30.mani_move[4], FALSE);
    btn = (CButton *)GetDlgItem(IDC_EDIT_MOVE_R5);
    SetDlgItemInt(IDC_EDIT_MOVE_R5, PC_from_vm30.mani_move[5], FALSE);

    UpdateData();

    return TRUE;
}
```

다음에 보이는 화면은 위에서 작성한 클라이언트 프로그램의 실행 화면으로, 서버는 VxWorks 기반의 VM30 보드와 서로 네트워크 데이터 통신을 수행한 결과 화면을 보여주고 있다.

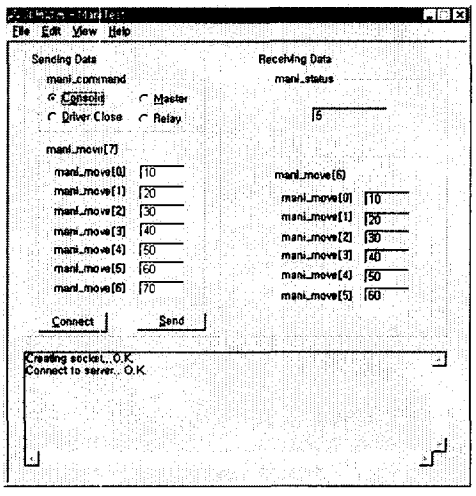

그림 8. 통신 프로그램의 수행 화면

제 3 절. 로보트 제어를 위한 힘 반향 조이스틱 프로그램

1. 힘 반향 조이스틱

이동 로보트와 장애물과의 충돌을 방지할 수 있도록 로보트 둘레에 거리 감지 센서를 부착하여 로보트 주위의 장애물에 대한 정보를 얻은 후, 이 정보를 가지고 조작자의 조이스틱에 힘으로 느낄 수 있도록 하면 이동 로보트의 보다 안정된 주행을 보장할 수 있게 된다.
여기서는 힘 반향 알고리즘을 실험하기 위해 사용한 조이스틱은 그림 2-2에 보이는 CH Products사에서 개발한 가상 현실(virtual reality) 제품들 중 하나인 Force FX이다.

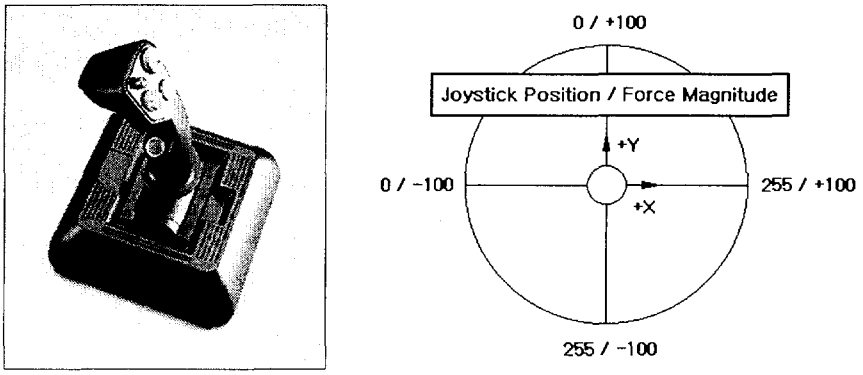

그림 9. 힘 반향 조이스틱의 외관과 좌표계

CH Products사는 프로그램 개발을 위하여 I-FORCE API/SDK를 제공하고 있으며, 본 실험에서는 X축과 Y축에 대한 극좌표값(값의 범위;-100~+100)으로 이루어진 벡터 단위의 힘을 반향시켜주는 함수를 사용하였

다.[14] 그림 2-3는 실험에 사용한 힘 반향 조이스틱의 데이터 흐름도를 보여주고 있다.

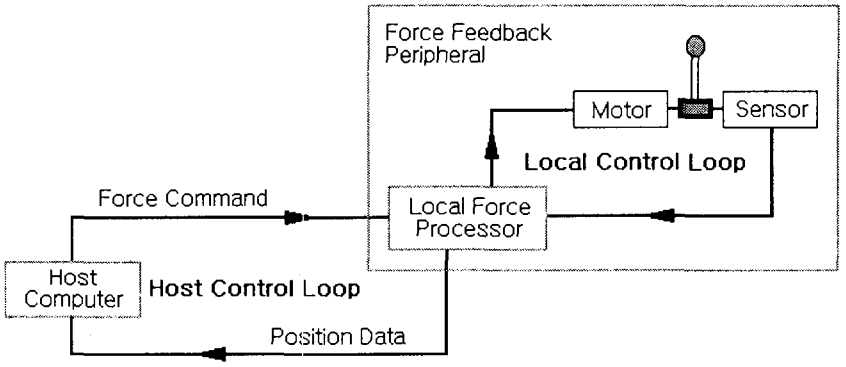

그림 10. 힘 반향 조이스틱의 데이터 흐름도

이 구성도에서 힘 반향 조이스틱은 마스터/슬레이브(Master/Slave)의 구조를 가진 원격 제어 시스템의 마스터 부분에 해당된다. 호스트 컴퓨터와 조이스틱은 양방향 통신(Bi-directional Communication)을 행하며, 분산 처리(Distribution Processing)의 구조를 갖고 있다.

2. 힘 반향 조이스틱 프로그램

여기서는 CH Products사에서 제공하는 힘반향 조이스틱 라이브러리를 사용하는 방법에 대해 기술하고자 한다.

우선 IFORCE API/SDK를 사용하기 위해, Visual C++의 세팅 부분의 옵션을 고쳐 주어야 한다. Visual C++ 5.0을 기준으로, 메뉴의 Project - Settings를 선택한다. 그런 다음 C/C++ 탭 컨트롤을 선택하고 Category라고 적힌 콤보 박스의 항목을 Precompiled Headers로 맞춘다. 그러면 하위 메뉴들이

바뀜을 확인할 수 있는데, 디폴트(default) 세팅은 Use precompiled header file (.pch)로 되어 있다. 이 옵션을 Automatic use of precompiled headers로 바꾼다. 이렇게 하지 않으면, IFORCE.DLL을 실행시 로딩하는 역할을 하는 iforceld.c 파일 컴파일시 fatal error를 발생시키기 때문이다. 또한 IFORCE.DLL은 C++ 코드가 아닌, C 코드로 작성되었기 때문에 DLL 내에 구현된 함수의 선언부가 들어 있는 i-force.h, iforceld.h를 컴파일러에게 C 코드라는 것을 알려 주어야 한다. 따라서 IFORCE.DLL 함수가 호출되는 소스 코드의 앞 부분에 다음과 정의하여야 한다.

```
extern "C" {
#include "i-force.h"      // IFORCE DLL이 C 코드로 작성되었기
#include "iforceld.h"     // 때문에 C 함수임을 컴파일러에게 알려줌
}
```

위와 같이 헤더 파일을 삽입시켰다면, 프로그램 실행시 IFORCE.DLL을 읽어 들이고, 프로그램 종료시 IFORCE.DLL을 해제시켜야 한다.

```
CJoySimView::CJoySimView() {
    // Load IFORCE.DLL
    if (IForceInit() == FALSE) // Load IFORCE.DLL
        AfxMessageBox("IFORCE.DLL load failed.\n");
    else
      TRACE("IFORCE.DLL loaded successfully.\n");
```

```
    // Connect to I-FORCE joystick
    if (!AutoDetectIForce())
        AfxMessageBox("Cannot auto-detected I-FORCE joystick.");
    else
        TRACE("I-FORCE joystick was auto-detected.\n");
}

CJoySimView::~CJoySimView() {
    pCloseStick();  // Disconnect I-FORCE joystick (Clear status)
    TRACE("Clears all status and closes communications.\n");

    IForceEnd();  // Unload IFORCE.DLL
    TRACE("Unloaded IFORCE.DLL file.\n");
}
```

위 소스의 생성자 함수에서 호출하는 AutoDetectIForce 함수는 현재 통신 포트들을 검사하여 힘 반향 조이스틱 ForceFX가 연결된 시리얼 포트를 찾는 함수이다.

```
int CJoySimView::AutoDetectIForce(void) {  // 조이스틱이 연결된
    int COMPort;                           // 포트를 자동으로 찾기
    BOOL IFResult;
```

```
// COM 포트를 1번부터 4번까지 검사하여 힘 반향 조이스틱이
// 연결된 포트를 찾는다.
for (COMPort = 1; COMPort <= 4; COMPort++) {
    pSetJoystickPort(COMPort);  // 조이스틱 세팅 함수 호출
    IFResult = pInitStick(&StickRec);  // 조이스틱 초기화
    if (IFResult) {  // 조이스틱이 해당 포트에 제대로 연결되었다면
        TRACE("Connected to ForceFX on COM%d port.\n",
            COMPort);
        TRACE("Version %d.%d.%d, Data %d/%d/%d, S/N %lu, ",
            StickRec.Version.major, StickRec.Version.minor,
            StickRec.Version.subminor,
            StickRec.Date.month, StickRec.Date.day,
            StickRec.Date.year,
            StickRec.SerialNumber);
        TRACE("Model %u, ForceAxes %d, PosAxes %d\n",
            StickRec.ForceAxes, StickRec.PositionAxes);

        // Enable Forces, wait for return value of true
        while (!(IFResult = pEnableForces()))  // 힘 반향을 가능케
            sleep(50);

        // Wait for serial line to settle
        sleep(100);
        // Enable digital position reporting every 20ms
        pEnablePositionReporting(20);
```

```
            return TRUE;
        }
        else {  // 해당 포트에서 힘 반향 조이스틱을 찾지 못했다면
            TRACE("Unable to connect to I-FORCE Joystick on ");
            TRACE("COM %d port.\n", COMPort);
            sleep(100);
            pCloseStick();
        }
    }
    TRACE("No I-FORCE Stick found.\n");
    return FALSE;
```

이렇게 힘 반향 조이스틱의 초기화 및 해제 루틴을 작성한 다음에 해야 할 일은 타이머를 동작시키고 타이머 루틴에 의한 조이스틱 좌표값을 읽는 부분이다.

```
void CJoySimView::OnInitialUpdate() {  // OnInitialUpdate에 overide
    CView::OnInitialUpdate();

    m_nTimer = SetTimer(1, 50, NULL);  // 타이머를 설치
    ASSERT(m_nTimer != 0);  // 타이머가 제대로 설치되었는지 검사
    TRACE("Timer running...\n");
}
```

```
void CJoySimView::OnTimer(UINT nIDEvent) {
    pGetStickPosition(&JP);  // 조이스틱의 좌표 값을 읽음
    JoyX = JP.x;
    JoyY = JP.y;
    JoyButton = JP.buttons;
    Invalidate();  // 화면을 다시 그림
    CView::OnTimer(nIDEvent);
}
```

위의 조이스틱 관련 API 함수를 사용하여 작성한 프로그램의 소스 코드는 <부록>에 실었다. 다음 그림은 <부록>에 실은 힘 반향 조이스틱의 가상 환경 실험 프로그램의 실행 화면이다.

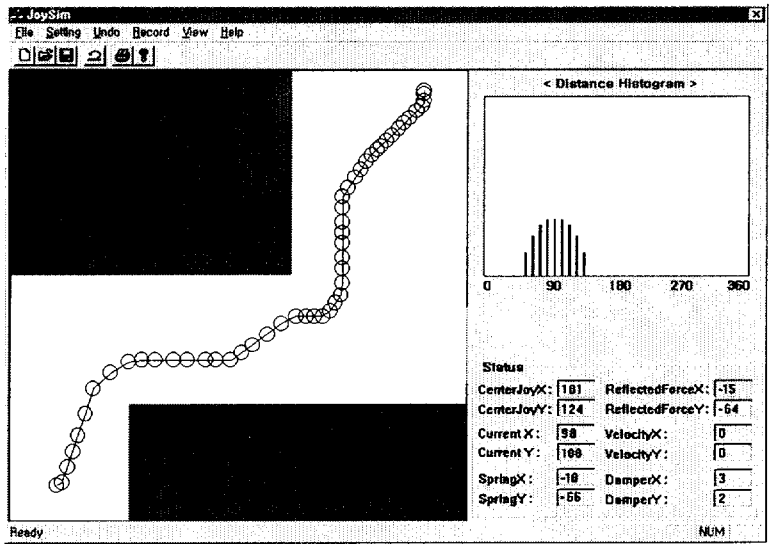

그림 11. 힘반향 조이스틱의 가상 환경 실험 프로그램 >

제 4 절. 매니퓰레이터 제어를 위한 OpenGL 프로그램

1. OpenGL의 개요

OpenGL이란 3차원 그래픽 라이브러리이다. 컴퓨터 그래픽 분야에서 유명한 미국의 실리콘 그래픽스(Silicon Graphics, 이하 SGI)가 워크스테이션에서 사용하던 IRIS GL을 플랫폼이나 운영 체제와 관계 없이 사용할 수 있도록 여러 회사들과 컨소시엄을 구성하여 만든 그래픽 라이브러리이다. 처음에는 그래픽 기능을 중시한 워크스테이션을 제작할 때 IRIS GL이라는 획기적이고 막강한 라이브러리를 만들었다. 그러나 실리콘 그래픽스는 여기서 멈추지 않고 사용하기 쉬운 여러 가지 그래픽 툴들을 만들어 나갔다. 현재까지도 고수준의 그래픽 라이브러리로는 Inventor, Performer를 따라갈 제품이 없다. SGI는 OpenGL 기반으로 예전에 개발된 여러 가지 툴들을 사용할 수 있도록 변형하여 OpenInventor 등이 상용 제품으로 내놓게 되었다.

OpenGL은 표준이 결정되면서 많은 업체들에 의해 여러 가지 기종으로 포팅되기 시작하였다. 현재는 SUN, RS6000, Windows NT, Windows 95 등에서 사용할 수 있다. OpenGL의 각 기종별 포팅은 Windows NT의 경우는 Microsoft사에서, 나머지는 Template Graphics Software(TGS)에서 진행 중이다.

OpenGL은 우수한 성능과 가능에도 불구하고 PC에서는 성능이 미약했기 때문에 많이 알려지지는 않았다. Windows NT에 내장되어 개발자의 관심을 끄는 데는 성공했지만, 워크스테이션과는 달리 PC와 Windows NT의 성능 미비로 제 성능을 충분히 발휘하지 못했던 것이다. 따라서 Windows 95에 OpenGL이 포함될 것인가에 많은 논란이 있었고, Windows 95가 발표된 지 3~4개월만에 별도로 제공되었다. 이제 Windows 95에도 포함되었

고 OpenGL을 지원하는 Accelerator 보드가 속속 발표되고 있으므로, 보다 대중적인 기반이 마련될 것이다.

그 뿐만 아니라 OpenGL은 대중적인 기반을 넘어 3-D 그래픽의 표준 API로 자리잡게 될 것이다. 그 이유는 다음과 같다.

(1) 많은 회사에서 OpenGL을 선택하고 있다.

Microsoft, INTEL, IBM, Mitsubishi, Sony, SGI, Sun Micro System, Digital Equipment, HP, Konix Graphics, TGS, Apple 등이 고성능 3-D 그래픽스를 위한 개방형 표준으로 OpenGL을 선택하거나 개발하고 있다.

(2) 다양한 플랫폼에서 개발이 가능하다.

마이크로소프트에서 윈도우 환경에서 개발한 DirectX, Direct3D와는 다르게 OpenGL은 여러 가지 플랫폼에서 개발이 가능하다. 보통 쓰고 있는 PC급에서 Onyx, Indigo, Indigo 2, 그리고 RS/6000과 같은 고성능 워크스테이션에 이르기까지 다양한 플랫폼에서 작동한다.

(3) C 라이브러리이다.

OpenGL은 객체지향적 구조가 아니다. 즉 C++ 라이브러리가 아니라 C 라이브러리이다. 이것은 단점이라기 보다는 장점으로 더욱 부각된다. 따라서 OpenGL은 호환성이 좋고 이해하기가 쉽다.

(4) 클라이언트-서버 모델에 유용하다.

OpenGL은 클라이언트-서버 모델에 사용하며, 클라이언트가 명령어를 보내면 OpenGL 서버는 명령어를 해석하여 처리한다. 물론 서로 다른 기종에서 동작할 수 있고, 계산이 많이 필요한 작업은 분산 처리로 수행할 수 있다.

(5) 사용자 인터페이스 환경과 독립적이다.

3-D 그래픽 자체를 위한 함수는 X-Windows 혹은 MS-Windows에서 똑같은 이름의 함수를 사용하게 되어 있다. 때문에 다양한 플랫폼으로 포팅하는데에 아주 편리하다. 그러나 개별적인 운영체제와 그래픽 인터페이스 환경을 위해서 서로 다른 확장 라이브러리가 조금씩 필요하다.

2. MFC에서의 OpenGL 라이브러리 사용

먼저 OpenGL 코드를 MFC 응용 프로그램에 삽입하기 전에 View 클래스의 헤드 파일 혹은 StdAfx.h 파일에 다음과 같이 헤더 파일을 포함시킨다. 이 헤더 파일들은 OpenGL 라이브러리와 유틸리티를 포함시키는 역할을 하게 된다.

```
#include "gl/gl.h"
#include "gl/glu.h"
#include "gl/glaux.h"      // Auxiliary Library를 사용할 경우에 포함시킴
```

(1) 윈도우 스타일 설정

PreCreateWindow 메소드의 코드를 입력한다. 클래스 위저드를 이용해서 View 클래스를 선택하면 PreCreateWindow 메소드가 존재한다. 이 멤버 함수에 윈도우 스타일 WS_CLIPCHILDREN과 WS_CLIPSIBLING을 추가한다.

OpenGL은 두 가지 윈도우 스타일이 설정된 상태에서만 동작한다. 그 이유는 디바이스 컨텍스트(device context)에서 픽셀 포맷을 설정할 수 있는 윈도우 스타일이기 때문이다. OpenGL은 부모 윈도우를 가지는 윈도우(Sibling) 혹은 자신 윈도우를 가지는 윈도우(Child)에 대해서는 픽셀 포맷을 설정할 수 없다. 그래서 이 두 가지의 경우의 클라이언트 영역 속성을 배제시키는 것이다. 만약 픽셀 포맷이 설정되지 않으면 OpenGL은 랜더링을 할 수 없게 된다. 반대로 픽셀 포맷이 설정되면 응용 프로그램이 OpenGL을 사용할 수 있다는 뜻이 된다.

```
BOOL CTestGLView::PreCreateWindow(CREATESTRUCT& cs) {
    // TODO: Modify the Window class or styles here by modifying
    //          the CREATESTRUCT cs
    cs.style |= WS_CLIPSIBLINGS | WS_CLIPCHILDREN;

    reutrn CView::PreCreateWindow(cs);
}
```

(2) 픽셀 포맷 설정하기

픽셀 포맷을 설정하기 위해 SetupPixelFormat() 함수를 만든다. 이 함수는 PIXELFORMATDESCRIPTOR 구조체를 사용해서 픽셀 포맷을 초기화한다. PIXELFORMATDESCRIPTOR 구조체에서는 구조체의 크기, 버전, 속성 플래그, 컬러 비트 수, 버퍼링 지정, 메인 레이어 타입 등의 정보를 가지고 있게 된다. 구조체의 첫 번째 필드인 nSize 필드는 이 구조체의 사이즈를 바

이트로 표시하고, 버전 필드인 nVersion 필드는 무조건 '1'로 설정한다.

속성필드인 dwFlags 필드는 ChoosePixelFormat()이 사용되었을 때 픽셀 포맷의 속성을 정의하게 된다. OpenGL을 사용하려면 PFD_SUPPORT_OPENGL, 윈도우 클라이언트 영역을 랜더링하려는 경우 PFD_DRAW_TO_WINDOW, 더블 버퍼링을 사용하는 경우에는 PFD_DOUBLEBUFFER를 속성값에 넣어주면 된다. 더블 버퍼링은 화면 버퍼(Front Buffer)와 내부 버퍼(Hidden Back Buffer)를 사용해서 보다 빠르게 이미지를 구현할 수 있다.

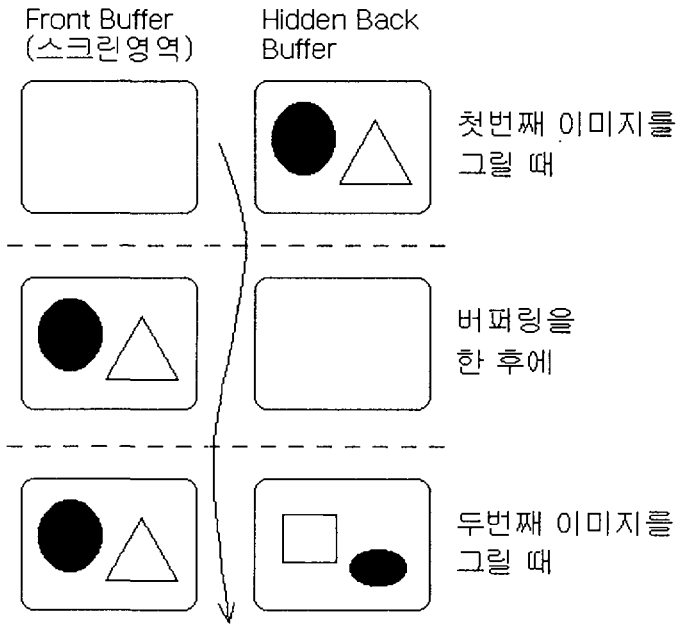

그림 12. 더블 버퍼링

iPixelType 필드에 PFD_TYPE_RGBA가 설정되면 RGB 형태의 픽셀 타입으로 색상이 구성된다는 것을 의미하며, PFD_TYPE_COLORINDEX는 RGB 대신에 팔레트의 인덱스를 설정해서 색상을 나타낸다. 물론 OpenGL은 두

방식의 타입을 모두 지원하지만 전자를 많이 사용한다. 그 이유는 RGB 색상 설정이 보다 다루기 쉽고 평범하기 때문이다. nColorBits 필드 다음부터 cDepthBits 필드 전까지는 여러 가지 색상의 픽셀당 비트 수를 표시한다. cDepthBits 필드는 깊이 정보를 나타내는 것으로, 깊이 버퍼의 단위 픽셀당 비트 수를 의미한다. 그리고 깊이 버퍼는 위에서 언급한 내부 버퍼(Hidden Back Buffer)에 이용되는 버퍼이다. iLayerType 필드는 레이어 타입을 설정하며, 거의 대부분 PFD_MAIN_PLANE으로 설정되어야 한다.

픽셀 포맷이 지정되면 ChoosePixelFormat() 함수를 이용해 제공된 픽셀 포맷과 가장 가까운 디바이스 컨텍스트의 픽셀 포맷을 얻어 낸다. 그런 다음 SetPixelFormat() 함수로 지정된 픽셀 포맷 인덱스로 현재 픽셀 포맷을 설정한다.

```
typedef struct tagPIXELFORMATDESCRIPTOR { // pfd
    WORD  nSize;            // PIXELFORMATDESCRIPTOR의 사이즈
    WORD  nVersion;         // PIXELFORMATDESCRIPTOR의 버전
    DWORD dwFlags;          // 속성 플래그
    BYTE  iPixelType;       // 픽셀 타입
    BYTE  cColorBits;       // 픽셀당 색상 비트 수
    BYTE  cRedBits;         // 픽셀당 빨간색 비트 수
    BYTE  cRedShift;        // 빨간색 시프트 카운트
    BYTE  cGreenBits;       // 픽셀당 녹색 비트 수
    BYTE  cGreenShift;      // 녹색 시프트 카운트
    BYTE  cBlueBits;        // 픽셀당 파란색 비트 수
    BYTE  cBlueShift;       // 파란색 시프트 카운트
    BYTE  cAlphaBits;       // 픽셀당 알파 비트 수
    BYTE  cAlphaShift;      // 알파 시프트 카운트
    BYTE  cAccumBits;       // 앤티앨리어싱 등과 같은 특수 효과를 나타낼 때 사용
    BYTE  cAccumRedBits;    // 빨간색의 축적 버퍼에서 픽셀당 비트 수
    BYTE  cAccumGreenBits;  // 녹색의 축적 버퍼에서 픽셀당 비트 수
    BYTE  cAccumBlueBits;   // 파란색의 축적 버퍼에서 픽셀당 비트 수
    BYTE  cAccumAlphaBits;  // 알파의 축적 버퍼에서 픽셀당 비트 수
    BYTE  cDepthBits;       // 깊이 버퍼의 픽셀당 비트 수
    BYTE  cStencilBits;     // 스텐실 버퍼의 픽셀당 비트 수
    BYTE  cAuxBuffers;      // 보조 버퍼의 수
    BYTE  iLayerType;       // PFD_MAIN_PLANE 레이어 타입 설정
    BYTE  bReserved;        // 예약 필드
    DWORD dwLayerMask;      // 일반적으로 지원하지 않음
    DWORD dwVisibleMask;    //        "
    DWORD dwDamageMask;     //        "
} PIXELFORMATDESCRIPTOR;
```

그림 13. PIXELFORMATDESCRIPTOR 구조체

(3) 렌더링 컨텍스트 생성하기

디바이스 컨텍스트에 대한 픽셀 포맷을 설정한 후에는 이것과 연결되는 렌더링 컨텍스트를 생성해야 한다. 만약 렌더링 컨텍스트가 설정되지 않는다면 OpenGL 명령어는 아무런 소용이 없게 된다.

렌더링 컨텍스트를 생성하는 방법은 두 가지가 있다. 첫 번째 방법은 OpenGL이 오브젝트를 그릴 때 렌더링 컨텍스트를 설정하고, 사용이 끝난 다음 즉시 해제하는 것이다. 즉, WM_CREATE 메시지가 발생할 때 렌더링 컨텍스트를 생성만 하고 있다가, WM_PAINT 메시지가 발생하면 사용할 렌더링 컨텍스트를 만들어 사용하고 그 후에 사용했던 렌더링 컨텍스트를 해제시키는 것이다.

두 번째 방법은 WM_CREATE 메시지가 발생되고 나서 렌더링 컨텍스트를 생성하고 사용할 렌더링 컨텍스트까지 설정하는 것이다. 즉, 오브젝트를 그릴 때마다 사용할 렌더링 컨텍스트를 매번 얻어오지 않고 오직 한 번만 만들어 주는 것이다.

이 두 가지 방법은 서로 장단점을 갖고 있다. 두 번째 방법처럼 코드가 간편해지면 불필요한 리소스의 사용이 많아진다는 것이다. 반면에 첫 번째 방법은 사용이 끝난 렌더링 컨텍스트는 바로 해제를 시켜주어야만 한다는 다소 귀찮은 점이 있다. 코딩에서는 두 번째 방법을 이용하였으며, 간단하게 설명하면 다음과 같다. 디바이스 컨텍스트를 메모리 할당 연산자 new를 통해 할당받는다. 그리고 OpenGL에 적합한 픽셀 포맷을 설정한 후 렌더링 컨텍스트를 생성한다. 그리고 이것을 디바이스 컨텍스트에 연결하면 된다. 한 가지 고려해야 할 점은 픽셀 포맷이 꼭 OpenGL 명령어가 수행되기 전에 설정되어야 하기 때문에 OnCreate() 함수에서 지정해야 한다는 것이다. 픽셀 포맷은 위에서 설명한 SetupPixelFormat() 함수를 사용해서 설정했다. 렌더링 컨텍스트를 생성하기 위해서는

wglCreateContext() 함수를 사용해야 한다. 'wgl'로 시작하는 함수는 "위글 함수"라고 부르며, 디바이스 컨텍스트와 같이 윈도우에 대한 일련의 정보를 렌더링 컨텍스트와 함께 사용하는데 이용된다. 아래 소스에서 wglCreateContext() 함수는 렌더링 컨텍스트를 나타내는 뷰 클래스의 멤버 변수 m_hRC를 리턴하는데, 바로 렌터링 컨텍스트를 생성하는 것이다. 다음 wglMakeCurrent() 함수는 리턴된 컨텍스트 핸들을 가지고 사용할 컨텍스트를 설정한다. 다음은 위글 함수의 종류와 그에 대한 설명이다.

표 1. 위글 함수의 종류

WGL 함수 이름	내 용
wglCreateContext()	새로운 렌더링 컨텍스트를 생성한다.
wglDeleteContext()	렌더링 컨텍스트를 제거한다.
wglGetCurrentContext()	현재 사용하는 랜더링 컨텍스트 핸들을 얻는다.
wglGetCurrentDC()	현재 사용하는 렌더링 컨텍스트와 관련된 디바이스 컨텍스트 핸들을 얻는다.
wglMakeCurrent()	현재 사용할 렌더링 컨텍스트를 만든다.

(4) 3-D 광경 설정하기

3-D 광경을 디스플레이에서 보려면 복잡한 절차가 따르게 된다. 우리가 보는 3-D 화면은 화면 전체가 아니라 응용 프로그램의 윈도우 클라이언트 영역이다. 그렇다면 윈도우 클라이언트 영역의 크기를 먼저 알아야만 할 것이다. 그러기 위해 클라이언트 영역이 새로 만들어지거나 크기가 변경되면 발생하는 WM_SIZE 메시지와 그 메시지를 처리하는 OnSize() 메

시지 핸들러 함수를 사용해야 한다. 이 함수를 통해서 윈도우 크기가 주어진 경우에 3-D 좌표계를 어떻게 구성하는지 살펴볼 수 있을 것이다. 먼저 OpenGL은 오브젝트의 변환을 위해서 행렬을 이용한다. 정확히 오브젝트의 변환과 오브젝트의 좌표계 변환을 위해서 두 개의 행렬 변환 스택을 가지고 사용하게 된다. 이 두 가지 모드는 glMatrixMode() 함수를 통해서 이루어지게 된다. 그리고 변환 행렬은 기존의 스택에 쌓여 있는 것들과 연관되어 계산되므로 단위 행렬을 불러냄으로써 초기화될 수있다. 이 때 사용하는 함수가 glLoadIdentity() 함수이다. 다음에 3-D 좌표계를 설정할 때 가장 먼저 생각할 점은 직교 좌표계를 사용할 것인지, 원근 좌표계를 사용할 것인지를 결정하는 것이다. 직교 좌표계는 특이한 경우를 제외하고는 원근감이 나타나지 않으므로 주로 원근 좌표계를 이용한다. glPerspective() 함수가 바로 원근 좌표계를 설정하는 역할을 한다. 또한 시야의 각도와 렌더링할 Z 값을 설정해준다.

(5) OnEraseBkgnd 메소드 처리 (오버로드 함수)

윈도우는 다시 그려야 될 응용 프로그램이 있을 경우 배경을 매번 다시 그리게 된다. 이렇게 되면 속도가 느려진다. 그래서 OpenGL은 스크린의 배경을 다시 그리는 것을 막아서 속도의 효율성을 높인다. 즉, 스크린 내에 렌더링을 하기 위해 배경을 다시 그리는 일이 없도록 해준다. 그렇게 하기 위해서는 리턴 값을 TRUE로 하면 된다.

(6) 오브젝트 그리기

오브젝트를 그리기 위해서는 WM_PAINT 메시지가 발생되고, 이것을 처리하는 메시지 핸들러에서 OpenGL 명령어를 써서 광경을 그리면 된다.

즉, 실제로 렌더링되는 OnDraw() 멤버 함수에 디스플레이 루틴을 첨가하면 된다. DrawScene() 함수가 표현할 오브젝트나 광경을 가지고 있다.

(7) 응용 프로그램 종료시 리소스 제거하기

OnCreate에서 new 연산자를 통해 디바이스 컨텍스트를 생성했다면 delete 연산자를 사용해서 제거해주어야 한다. 또 렌더링 컨텍스트를 생성했다면 마찬가지로 delete 연산자를 사용해서 제거해주어야 한다. 이 때 유의해야 할 점은 렌더링 컨텍스트를 제거하고 나서, 나중에 디바이스 컨텍스트를 제거한다는 것이다. 렌더링 컨텍스트를 제거하려면 wglDeleteContext() 함수를 쓰면 된다.

다음은 CView 클래스에서 계승받아 만든 COpenGLView 클래스이다. Visual C++를 사용하여 OpenGL 프로그램을 작성하기 위해 위에서 언급한 절차들을 코딩해놓은 것이다. 만약 새로운 프로그램을 작성하려면 이 뷰 클래스에서 계승받아 자신의 뷰 클래스를 만들던지, 아니면 수정하여 사용하면 된다.

```
// COpenGLView.h : interface of the COpenGLView class
//
/////////////////////////////////////////////////////////////////////////////

// Include the OpenGL headers
#include "gl\gl.h"
#include "gl\glu.h"
#include "gl\glaux.h"

class COpenGLView : public CView
{
protected: // create from serialization only
        COpenGLView();
```

```
        DECLARE_DYNCREATE(COpenGLView)

// Attributes
public:
        COpenGLViewClassDoc* GetDocument();

// Operations
public:

// Overrides
        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(COpenGLView)
        public:
        virtual void OnDraw(CDC* pDC);  // overridden to draw this view
        virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
        //}}AFX_VIRTUAL

// Implementation
public:
        virtual ~COpenGLView();
#ifdef _DEBUG
        virtual void AssertValid() const;
        virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Generated message map functions
protected:
        //{{AFX_MSG(COpenGLView)
        afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
        afx_msg void OnDestroy();
        afx_msg BOOL OnEraseBkgnd(CDC* pDC);
        afx_msg void OnSize(UINT nType, int cx, int cy);
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()

        // The following was added

        virtual BOOL SetupPixelFormat( void );
        virtual BOOL SetupViewport( int cx, int cy );
        virtual BOOL SetupViewingFrustum( GLdouble aspect_ratio );
        virtual BOOL SetupViewingTransform( void );
        virtual BOOL PreRenderScene( void ) { return TRUE; }
        virtual void RenderStockScene( void );
        virtual BOOL RenderScene( void );

private:
        BOOL InitializeOpenGL();
        void SetError( int e );


        HGLRC     m_hRC;
        CDC*      m_pDC;
```

```
        static const char* const __ErrorStrings[];
        const char* m__ErrorString;

};

#ifndef __DEBUG  // debug version in COpenGLView.cpp
inline COpenGLViewClassDoc* COpenGLView::GetDocument()
   { return (COpenGLViewClassDoc*)m__pDocument; }
#endif

/////////////////////////////////////////////////////////////////////////////


// COpenGLView.cpp : implementation of the COpenGLView class
//

#include "stdafx.h"
#include "OpenGLViewClass.h"

#include "OpenGLViewClassDoc.h"
#include "COpenGLView.h"

#ifdef __DEBUG
#define new DEBUG__NEW
#undef THIS__FILE
static char THIS__FILE[] = ____FILE____;
#endif



const char* const COpenGLView::__ErrorStrings[]= {
        {"No Error"},                   // 0
        {"Unable to get a DC"},         // 1
        {"ChoosePixelFormat failed"},   // 2
        {"SelectPixelFormat failed"},   // 3
        {"wglCreateContext failed"},    // 4
        {"wglMakeCurrent failed"},      // 5
        {"wglDeleteContext failed"},    // 6
        {"SwapBuffers failed"},         // 7
};

/////////////////////////////////////////////////////////////////////////////
// COpenGLView

IMPLEMENT__DYNCREATE(COpenGLView, CView)

BEGIN__MESSAGE__MAP(COpenGLView, CView)
        //{{AFX__MSG__MAP(COpenGLView)
        ON__WM__CREATE()
        ON__WM__DESTROY()
        ON__WM__ERASEBKGND()
        ON__WM__SIZE()
        //}}AFX__MSG__MAP
```

```
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// COpenGLView construction/destruction

COpenGLView::COpenGLView() :
            m_hRC(0), m_pDC(0), m_ErrorString(_ErrorStrings[0])

{
            // TODO: add construction code here

}

COpenGLView::~COpenGLView()
{
}

BOOL COpenGLView::PreCreateWindow(CREATESTRUCT& cs)
{
            // TODO: Add your specialized code here and/or call the base class

            // An OpenGL window must be created with the following flags and must not
            // include CS_PARENTDC for the class style.
            cs.style |= WS_CLIPSIBLINGS | WS_CLIPCHILDREN;

            return CView::PreCreateWindow(cs);
}


/////////////////////////////////////////////////////////////////////////////
// COpenGLView drawing


void COpenGLView::OnDraw(CDC* pDC)
{
            COpenGLViewClassDoc* pDoc = GetDocument();
            ASSERT_VALID(pDoc);

            // TODO: add draw code for native data here

            ::glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );

            PreRenderScene();

            ::glPushMatrix();
            RenderStockScene();
            ::glPopMatrix();

            ::glPushMatrix();
            RenderScene();
            ::glPopMatrix();

            ::glFinish();
```

```
		if ( FALSE == ::SwapBuffers( m_pDC->GetSafeHdc() ) ) {
			SetError(7);
		}
}


/////////////////////////////////////////////////////////////////////////////
// COpenGLView diagnostics

#ifdef _DEBUG
void COpenGLView::AssertValid() const
{
		CView::AssertValid();
}

void COpenGLView::Dump(CDumpContext& dc) const
{
		CView::Dump(dc);
}

COpenGLViewClassDoc* COpenGLView::GetDocument() // non-debug version is inline
{
		ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(COpenGLViewClassDoc)));
		return (COpenGLViewClassDoc*)m_pDocument;
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// COpenGLView message handlers

int COpenGLView::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
		if (CView::OnCreate(lpCreateStruct) == -1)
			return -1;

		// TODO: Add your specialized creation code here

		InitializeOpenGL();

		return 0;
}


/////////////////////////////////////////////////////////////////////////////
// GL helper functions

void COpenGLView::SetError( int e )
{
		// if there was no previous error,
		// then save this one
		if ( _ErrorStrings[0] == m_ErrorString ) {
			m_ErrorString = _ErrorStrings[e];
		}
}
```

```
BOOL COpenGLView::InitializeOpenGL()
{
	// Can we put this in the constructor?
	m_pDC = new CClientDC(this);

	if ( NULL == m_pDC ) { // failure to get DC
		SetError(1);
		return FALSE;
	}

	if (!SetupPixelFormat()) {
		return FALSE;
	}

	//n = ::GetPixelFormat(m_pDC->GetSafeHdc());
	//::DescribePixelFormat(m_pDC->GetSafeHdc(), n, sizeof(pfd), &pfd);

	//  CreateRGBPalette();

	if ( 0 == (m_hRC = ::wglCreateContext( m_pDC->GetSafeHdc() ) ) ) {
		SetError(4);
		return FALSE;
	}

	if ( FALSE == ::wglMakeCurrent( m_pDC->GetSafeHdc(), m_hRC ) ) {
		SetError(5);
		return FALSE;
	}

	// specify black as clear color
	::glClearColor( 0.0f, 0.0f, 0.0f, 0.0f );
	// specify the back of the buffer as clear depth
	::glClearDepth( 1.0f );
	// enable depth testing
	::glEnable( GL_DEPTH_TEST );

	return TRUE;
}


BOOL COpenGLView::SetupPixelFormat()
{
	static PIXELFORMATDESCRIPTOR pfd = {
		sizeof(PIXELFORMATDESCRIPTOR),		// size of this pfd
		1,					// version number
		PFD_DRAW_TO_WINDOW |			// support window
		PFD_SUPPORT_OPENGL |			// support OpenGL
		PFD_DOUBLEBUFFER,			// double buffered
		PFD_TYPE_RGBA,				// RGBA type
		24,					// 24-bit color depth
		0, 0, 0, 0, 0, 0,			// color bits ignored
		0,					// no alpha buffer
```

```
                0,                                      // shift bit ignored
                0,                                      // no accumulation buffer
                0, 0, 0, 0,                             // accum bits ignored
                16,                     // NOTE: better performance with 16-bit z-buffer
                0,                                      // no stencil buffer
                0,                                      // no auxiliary buffer
                PFD_MAIN_PLANE,                         // main layer
                0,                                      // reserved
                0, 0, 0                                 // layer masks ignored
        };
        int pixelformat;

        if ( 0 == (pixelformat = ::ChoosePixelFormat(m_pDC->GetSafeHdc(), &pfd)) ) {
                SetError(2);
                return FALSE;
        }

        if ( FALSE == ::SetPixelFormat(m_pDC->GetSafeHdc(), pixelformat, &pfd) ) {
                SetError(3);
                return FALSE;
        }

        return TRUE;
}

void COpenGLView::OnDestroy()
{
        CView::OnDestroy();

        // TODO: Add your message handler code here

        if ( FALSE == ::wglDeleteContext( m_hRC ) ) {
                SetError(6);
        }

        if ( m_pDC ) {
                delete m_pDC;
        }
}

BOOL COpenGLView::OnEraseBkgnd( CDC* pDC )
{
        // TODO: Add your message handler code here and/or call default

        //      return CView::OnEraseBkgnd(pDC);
        return TRUE; // tell Windows not to erase the background
}

void COpenGLView::OnSize(UINT nType, int cx, int cy)
{
        CView::OnSize(nType, cx, cy);

        // TODO: Add your message handler code here
```

```
        GLdouble aspect_ratio; // width/height ratio

        if ( 0 >= cx || 0 >= cy ) {
                return;
        }


        SetupViewport( cx, cy );

        // select the projection matrix and clear it
        ::glMatrixMode( GL_PROJECTION );
        ::glLoadIdentity();

        // compute the aspect ratio
        // this will keep all dimension scales equal
        aspect_ratio = (GLdouble)cx/(GLdouble)cy;

        // select the viewing volumn
        SetupViewingFrustum( aspect_ratio );

        // switch back to the modelview matrix
        ::glMatrixMode( GL_MODELVIEW );
        ::glLoadIdentity();

        // now perform any viewing transformations
        SetupViewingTransform();
}

/////////////////////////////////////////////////////////////////////////////
// COpenGLView helper functions

BOOL COpenGLView::SetupViewport( int cx, int cy )
{
        // select the full client area
        ::glViewport(0, 0, cx, cy);

        return TRUE;
}

BOOL COpenGLView::SetupViewingFrustum( GLdouble aspect_ratio )
{
        // select a default viewing volumn
        ::gluPerspective( 40.0f, aspect_ratio, .1f, 20.0f );
        return TRUE;
}

BOOL COpenGLView::SetupViewingTransform()
{
        // select a default viewing transformation
        // of a 20 degree rotation about the X axis
        // then a -5 unit transformation along Z
        ::glTranslatef( 0.0f, 0.0f, -5.0f );
        ::glRotatef( 20.0f, 1.0f, 0.0f, 0.0f );
```

```
        return TRUE;
}


BOOL COpenGLView::RenderScene()
{
        // draw a red wire sphere inside a
        // light blue cube

        // rotate the wire sphere so it's vertically
        // oriented
        ::glRotatef( 90.0f, 1.0f, 0.0f, 0.0f );
        ::glColor3f( 1.0f, 0.0f, 0.0f );
        ::auxWireSphere( .5 );
        ::glColor3f( 0.5f, 0.5f, 1.0f );
        ::auxWireCube( 1.0 );

        return TRUE;
}

// Draw a square surface that looks like a
// black and white checkerboard
void COpenGLView::RenderStockScene()
{
        // define all vertices   X      Y      Z
        GLfloat v0[3], v1[3], v2[3], v3[3], delta;
        int color = 0;

        delta = 0.5f;

        // define the two colors
        GLfloat color1[3] = { 0.9f, 0.9f, 0.9f };
        GLfloat color2[3] = { 0.05f, 0.05f, 0.05f };

        v0[1] = v1[1] = v2[1] = v3[1] = 0.0f;

        ::glBegin( GL_QUADS );

        for ( int x = -5 ; x <= 5 ; x++ ) {
                for ( int z = -5 ; z <= 5 ; z++ ) {
                        ::glColor3fv( (color++)%2 ? color1 : color2 );

                        v0[0] = 0.0f+delta*z;
                        v0[2] = 0.0f+delta*x;

                        v1[0] = v0[0]+delta;
                        v1[2] = v0[2];

                        v2[0] = v0[0]+delta;
                        v2[2] = v0[2]+delta;

                        v3[0] = v0[0];
                        v3[2] = v0[2]+delta;
```

```
                    ::glVertex3fv( v0 );
                    ::glVertex3fv( v1 );
                    ::glVertex3fv( v2 );
                    ::glVertex3fv( v3 );
                }
            }
            ::glEnd();
}
```

상기 개발된 프로그램의 실행시 화면을 그림 14에 나타내었다.

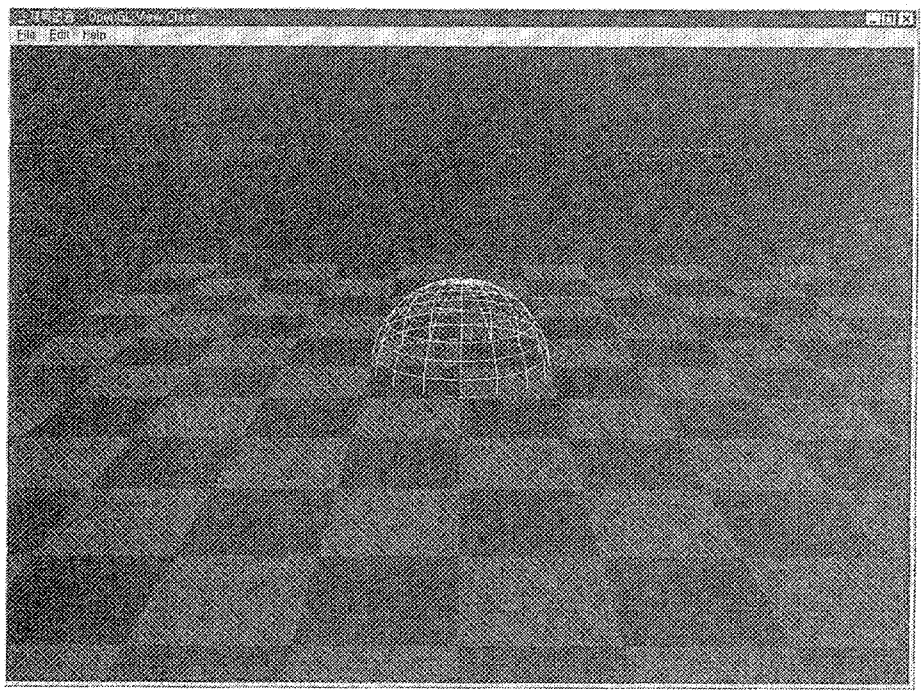

그림 14. 프로그램의 실행 화면

제 3 장. 결 론

원자력시설과 같은 극한환경에서 작업하는 이동 로보트의 관리제어시스템은 이동로보트 작업 성능을 좌우하는 매우 중요한 요소이다. 본 연구에서는 가상환경에서의 시뮬레이션과 온-라인 제어가 가능한 관리제어 시스템을 PC 윈도우즈 환경에서 개발하였다. 개발된 관리 제어시스템은 이동로보트와 소켓을 통하여 통신하도록 되어 있고, 조이스틱 제어는 힘반향 알고리즘을 사용하여 개발하였다.

개발된 시스템은 향후 계획하고 있는 원격 이동로보트 및 메니퓰레이터의 성능을 크게 향상시키리라 예상된다.

참고문헌

[1] Inside Visual C++ 5, David J. Kruglinski, Microsoft Press 1997

[2] Microsoft Visual C++ - The Four-Volume Documentation Collection for Microsoft Visual C++ Version 5 for Win32 Vol.1-6

[3] Advanced Programming in the UNIX Environment, W.Richard Stevens, Addison Wesley 1992

[4] VxWorks Training Workshop - Version 5.1, WIND RIVER SYSTEMS 1993

[5] VxWorks User's Guide - Version 5.1, WIND RIVER SYSTEMS 1993

[6] VxWorks Programmer's Guide - Version 5.1, WIND RIVER SYSTEMS 1993

[7] VxWorks Reference Manual - Version 5.1, WIND RIVER SYSTEMS 1993

[8] Louis B. Rosenberg, A Force Feedback Programming Primer, Immersion Corporation, 1997

[9] OpenGL Programming Guide, OpenGL ARB, Addison Wesley 1993

[10] OpenGL Reference Guide, OpenGL ARB, Addison Wesley, 1993

[11] OpenGL 프로그래밍, 우상수, 사이버출판사, 1997

부 록. (힘 반향 조이스틱의 모의 환경 실험 프로그램 소스 코드)

```
// JoySimView.h : interface of the CJoySimView class
//
/////////////////////////////////////////////////////////////////////////////

#include "DrawObject.h"
#include "fstream.h"
#include "stdio.h"

extern "C" {
#include "i-force.h"
#include "iforceld.h"
}

struct myPOINT { float x, y; };
struct myLINE { myPOINT p1, p2; };
struct myRECT { myPOINT p1, p2; };

class CJoySimView : public CView
{
protected: // create from serialization only
    CJoySimView();
    DECLARE_DYNCREATE(CJoySimView)

// Attributes
public:
    myLINE Robot;
    myPOINT prev2Robot;
    myPOINT prev1Robot;
    myPOINT CenterJoy;
    int VelocityX;
    int VelocityY;
    BOOL out_flag;
    CPoint out_max;
    BOOL m_bRobotTrace;
    int minimum_distance;
    float SpringX;
    float SpringY;
    float DamperX;
    float DamperY;
```

```
    float ReflectedForceX;
    float ReflectedForceY;
    int distance[100];
    CDrawObject *m_pCurDrawObject;
    CJoySimDoc* GetDocument();
    myPOINT get_reaching_point(myPOINT robot, int angle);
    myPOINT find_intersect_point(struct myLINE l1, struct myLINE l2);
    int get_distance(myLINE *robot, myRECT obstacle);
    void draw_histogram();
    void draw_robot_status();
    // Dialog variables
    int     m_nAngleStart;
    int     m_nAngleEnd;
    int     m_nAngleStep;
    int     m_nRobotRadius;
    int     m_nDetectDistance;
    int     m_nSpring;
    int     m_nDamper;
    float   m_fSpring_Kcd;
    float   m_fDamper_Kb;

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CJoySimView)
    public:
    virtual void OnDraw(CDC* pDC);  // overridden to draw this view
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    virtual void OnInitialUpdate();
    protected:
    virtual BOOL OnPreparePrinting(CPrintInfo* pInfo);
    virtual void OnBeginPrinting(CDC* pDC, CPrintInfo* pInfo);
    virtual void OnEndPrinting(CDC* pDC, CPrintInfo* pInfo);
    //}}AFX_VIRTUAL

// Implementation
public:
    BOOL recordFlag;
    FILE* outFile;
    virtual ~CJoySimView();
#ifdef _DEBUG
    virtual void AssertValid() const;
```

```
    virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Generated message map functions
protected:
    //{{AFX_MSG(CJoySimView)
    afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
    afx_msg void OnMouseMove(UINT nFlags, CPoint point);
    afx_msg void OnLButtonUp(UINT nFlags, CPoint point);
    afx_msg void OnTimer(UINT nIDEvent);
    afx_msg void OnSetting();
    afx_msg void OnEditUndo();
    afx_msg void OnRecordStart();
    afx_msg void OnRecordEnd();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

#ifndef _DEBUG  // debug version in JoySimView.cpp
inline CJoySimDoc* CJoySimView::GetDocument()
   { return (CJoySimDoc*)m_pDocument; }
#endif

/////////////////////////////////////////////////////////////////////////////

// JoySimView.cpp : implementation of the CJoySimView class
//

#include "stdafx.h"
#include "JoySim.h"

#include "JoySimDoc.h"
#include "JoySimView.h"
#include "ForceFx.h"

#include "math.h"
#include "stdio.h"

#include "Setting.h"
#include "sizes.h"

#ifdef _DEBUG
```

```
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

#define JOY_OFFSET 15
#define PI (3.1415925)
#define toradian(x) ((PI/180)*x)
#define CRASH 1
#define FREE   0

#define TOP        0
#define BOTTOM   1
#define LEFT   2
#define RIGHT  3

float INF = (float) 30000;

/////////////////////////////////////////////////////////////////////////////
// CJoySimView

IMPLEMENT_DYNCREATE(CJoySimView, CView)

BEGIN_MESSAGE_MAP(CJoySimView, CView)
    //{{AFX_MSG_MAP(CJoySimView)
    ON_WM_LBUTTONDOWN()
    ON_WM_MOUSEMOVE()
    ON_WM_LBUTTONUP()
    ON_WM_TIMER()
    ON_COMMAND(ID_SETTING, OnSetting)
    ON_COMMAND(ID_EDIT_UNDO, OnEditUndo)
    ON_COMMAND(ID_RECORD_START, OnRecordStart)
    ON_COMMAND(ID_RECORD_END, OnRecordEnd)
    //}}AFX_MSG_MAP
    // Standard printing commands
    ON_COMMAND(ID_FILE_PRINT, CView::OnFilePrint)
    ON_COMMAND(ID_FILE_PRINT_DIRECT, CView::OnFilePrint)
    ON_COMMAND(ID_FILE_PRINT_PREVIEW, CView::OnFilePrintPreview)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CJoySimView construction/destruction

CJoySimView::CJoySimView()
```

```
{
    // TODO: add construction code here
    m_pCurDrawObject = NULL;
    out_flag = 0;
    out_max.x = 0;
    out_max.y = 0;
    Robot.p1.x = (float)(MAP_X / 2);
    Robot.p1.y = (float)(MAP_Y / 2);
    prev1Robot = Robot.p1;
    VelocityX = 0;
    VelocityY = 0;
    SpringX = (float)0;
    SpringY = (float)0;
    DamperX = (float)0;
    DamperY = (float)0;
    ReflectedForceX = (float)0;
    ReflectedForceY = (float)0;
    // Dialog variables
    m_nAngleStart = 0;
    m_nAngleEnd = 360;
    m_nAngleStep = 10;
    m_nRobotRadius = 10;
    m_nDetectDistance = 100;
    m_bRobotTrace = FALSE;
    m_nSpring = 100;
    m_nDamper = 100;
    m_fSpring_Kcd = 0;
    m_fDamper_Kb = 0;

    recordFlag = FALSE;
    // initialize distance array
    for (int i = 0; i < 100; i++)
        distance[i] = 0;
}

CJoySimView::~CJoySimView()
{
    if (recordFlag)
        fclose(outFile);
}

BOOL CJoySimView::PreCreateWindow(CREATESTRUCT& cs)
{
    // TODO: Modify the Window class or styles here by modifying
```

```
    //  the CREATESTRUCT cs

    return CView::PreCreateWindow(cs);
}

/////////////////////////////////////////////////////////////////////////////
// CJoySimView drawing

void CJoySimView::OnDraw(CDC* pDC)
{
    CJoySimDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    pDC->Rectangle(0, 0, 500, 500);        // map size
    pDC->Rectangle(MAP_X+20, 0, MAP_X+11+HISTOGRAM_X, HISTOGRAM_Y);   // histogram size

    // TODO: add draw code for native data here
    CRect rectClip, rectObject;
    pDC->GetClipBox(&rectClip);

    for (POSITION pos = pDoc->m_objectList.GetHeadPosition(); pos != NULL;) {
        CDrawObject *pObject = (CDrawObject *)pDoc->m_objectList.GetNext(pos);
        pObject->GetBoundingRect(&rectObject);
        if (!rectObject.IntersectRect(&rectObject, &rectClip))
            continue;
        pObject->DrawObject(pDC);
    }

    draw_robot_status();

    // draw_robot
    pDC->Rectangle((int)Robot.p1.x-m_nRobotRadius, (int)Robot.p1.y-m_nRobotRadius,
        (int)Robot.p1.x+m_nRobotRadius, (int)Robot.p1.y+m_nRobotRadius);
    //pDC->Ellipse((int)Robot.p1.x-m_nRobotRadius, (int)Robot.p1.y-m_nRobotRadius,
        (int)Robot.p1.x+m_nRobotRadius, (int)Robot.p1.y+m_nRobotRadius);
}

/////////////////////////////////////////////////////////////////////////////
// CJoySimView printing

BOOL CJoySimView::OnPreparePrinting(CPrintInfo* pInfo)
{
    // default preparation
    return DoPreparePrinting(pInfo);
```

```
}

void CJoySimView::OnBeginPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
    // TODO: add extra initialization before printing
}

void CJoySimView::OnEndPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
    // TODO: add cleanup after printing
}

/////////////////////////////////////////////////////////////////////////////
// CJoySimView diagnostics

#ifdef __DEBUG
void CJoySimView::AssertValid() const
{
    CView::AssertValid();
}

void CJoySimView::Dump(CDumpContext& dc) const
{
    CView::Dump(dc);
}

CJoySimDoc* CJoySimView::GetDocument() // non-debug version is inline
{
    ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CJoySimDoc)));
    return (CJoySimDoc*)m_pDocument;
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CJoySimView message handlers

void CJoySimView::OnLButtonDown(UINT nFlags, CPoint point)
{
    // TODO: Add your message handler code here and/or call default
    if (point.x < MAP_X && point.y < MAP_Y) {
        m_pCurDrawObject = new CRectangle(RGB(64,64,128), PS_SOLID, 1);
        m_pCurDrawObject->SetStartPoint(point);
        m_pCurDrawObject->SetEndPoint(point);
```

```
        SetCapture();
    }

    CView::OnLButtonDown(nFlags, point);
}

void CJoySimView::OnMouseMove(UINT nFlags, CPoint point)
{
    // TODO: Add your message handler code here and/or call default
    if (GetCapture() == this) {
        if (point.x >= MAP__X) {  // range check
            out__max.x = point.x = MAP__X;
            out__max.y = point.y;
            out__flag = TRUE;
        }
        if (point.y >= MAP__Y) {  // range check
            out__max.x = point.x;
            out__max.y = point.y = MAP__Y;
            out__flag = TRUE;;
        }

        CClientDC dc(this);

        dc.SetROP2(R2__NOTXORPEN);
        m__pCurDrawObject->DrawObject(&dc);
        m__pCurDrawObject->SetEndPoint(point);
        m__pCurDrawObject->DrawObject(&dc);
    }

    CView::OnMouseMove(nFlags, point);
}

void CJoySimView::OnLButtonUp(UINT nFlags, CPoint point)
{
    // TODO: Add your message handler code here and/or call default
    if (GetCapture() == this) {
        if (out__flag) {
            point = out__max;
            out__flag = FALSE;
        }

        CClientDC dc(this);

        m__pCurDrawObject->SetEndPoint(point);
```

```
        m_pCurDrawObject->DrawObject(&dc);

        ReleaseCapture();

        CJoySimDoc* pDoc = GetDocument();
        ASSERT_VALID(pDoc);

        pDoc->m_objectList.AddTail(m_pCurDrawObject);
        pDoc->SetModifiedFlag(TRUE);
    }

    CView::OnLButtonUp(nFlags, point);
}

void CJoySimView::OnInitialUpdate()
{
    CView::OnInitialUpdate();

    // TODO: Add your specialized code here and/or call the base class

    // get center postion of joystick
    if (pGetStickPosition(&JP)) {
        CenterJoy.x = (float)JP.x;
        CenterJoy.y = (float)JP.y;
    }

    // run a timer
    SetTimer(1, 50, NULL);
    TRACE("Timer running...\n");
}

void CJoySimView::OnTimer(UINT nIDEvent)
{
    // TODO: Add your message handler code here and/or call default
    BOOL joy_move_flag = FALSE;

    if (pGetStickPosition(&JP)) {
        int add_x, add_y;

        add_x = JP.x - (int)CenterJoy.x;
        add_y = JP.y - (int)CenterJoy.y;

        if (abs(add_x) <= JOY_OFFSET)
            add_x = 0;
```

```
    else
        joy__move__flag = TRUE;

    if (abs(add__y) <= JOY__OFFSET)
        add__y = 0;
    else
        joy__move__flag = TRUE;

    VelocityX = (joy__move__flag) ? int(add__x * 2.5 / 20) : 0;   // 127 / 50 = 2.5
    VelocityY = (joy__move__flag) ? int(add__y * 2.5 / 20) : 0;

    prev2Robot = prev1Robot;
    prev1Robot = Robot.p1; // save previous robot position
    Robot.p1.x += VelocityX;
    Robot.p1.y += VelocityY;

    // Range check (1) whether robot position is over the walls
    if (Robot.p1.x <= m__nRobotRadius)
        Robot.p1.x = float(m__nRobotRadius) + 1;
    if (Robot.p1.x >= MAP__X-m__nRobotRadius)
        Robot.p1.x = float(MAP__X-m__nRobotRadius) - 1;
    if (Robot.p1.y <= m__nRobotRadius)
        Robot.p1.y = float(m__nRobotRadius) + 1;
    if (Robot.p1.y >= MAP__Y-m__nRobotRadius)
        Robot.p1.y = float(MAP__Y-m__nRobotRadius) - 1;

    // Range check (2) whether robot position is over the obstacles
    CJoySimDoc* pDoc = GetDocument();
    ASSERT__VALID(pDoc);
    for (POSITION pos = pDoc->m__objectList.GetHeadPosition(); pos != NULL;) {
        CDrawObject *pObject = (CDrawObject *)pDoc->m__objectList.GetNext(pos);
        CRect robot(int(Robot.p1.x-m__nRobotRadius), int(Robot.p1.y-m__nRobotRadius),
            int(Robot.p1.x+m__nRobotRadius), int(Robot.p1.y+m__nRobotRadius));
        CRect obstacle;
        pObject->GetBoundingRect(&obstacle);
        if (robot.IntersectRect(robot, obstacle)) { // if intersect
/*
            if (Robot.p1.x >= obstacle.left && Robot.p1.x <= obstacle.right) {
                if (prev1Robot.y < obstacle.top - m__nRobotRadius &&
                    Robot.p1.y >= obstacle.top - m__nRobotRadius)
                    Robot.p1.y = float(obstacle.top - m__nRobotRadius - 1);
                else if (prev1Robot.y > obstacle.top + m__nRobotRadius &&
                    Robot.p1.y <= obstacle.bottom + m__nRobotRadius)
                    Robot.p1.y = float(obstacle.bottom + m__nRobotRadius + 1);
```

```
            }
            else if (Robot.p1.y >= obstacle.top && Robot.p1.y <= obstacle.bottom) {
                if (prev1Robot.x < obstacle.left - m__nRobotRadius &&
                    Robot.p1.x >= obstacle.left - m__nRobotRadius)
                    Robot.p1.x = float(obstacle.left - m__nRobotRadius - 1);
                else if (prev1Robot.x > obstacle.right - m__nRobotRadius &&
                    Robot.p1.x <= obstacle.right + m__nRobotRadius)
                    Robot.p1.x = float(obstacle.right + m__nRobotRadius + 1);
            }
*/
            Robot.p1 = prev1Robot;
        }
    }

    if (joy__move__flag) {  // if joystick was moved
        CClientDC dc(this);
        if (!m__bRobotTrace) { // if user want to draw new robot position only
            CBrush newBrush(dc.GetBkColor());
            CRect rect((int)prev1Robot.x-m__nRobotRadius, (int)prev1Robot.y-m__nRobotRadius,
                (int)prev1Robot.x+m__nRobotRadius, (int)prev1Robot.y+m__nRobotRadius);
            dc.SelectObject(&newBrush);             // select new pen
            dc.FillRect(rect, &newBrush);
            dc.SelectStockObject(WHITE__BRUSH);  // release newBrush
        }

        // draw robot
        dc.Rectangle((int)Robot.p1.x-m__nRobotRadius, (int)Robot.p1.y-m__nRobotRadius,
            (int)Robot.p1.x+m__nRobotRadius, (int)Robot.p1.y+m__nRobotRadius);

        // calculate SPRING term
        CJoySimDoc* pDoc = GetDocument();
        ASSERT__VALID(pDoc);

        int theta, i, nearest__index = (int)INF;
        int nearest__distance = (int)INF;
        myPOINT nearest__point;

        SpringX = SpringY = (float)0;
        nearest__point.x = (float)INF;
        nearest__point.y = (float)INF;

        for (theta = m__nAngleStart, i = 0; theta < m__nAngleEnd; theta += m__nAngleStep, i++) {
            minimum__distance = (int)INF;
            Robot.p2 = get__reaching__point(Robot.p1, theta);
```

```
    int temp__distance;
    for (POSITION pos = pDoc->m__objectList.GetHeadPosition(); pos != NULL;) {
        CDrawObject *pObject = (CDrawObject *)pDoc->m__objectList.GetNext(pos);
        CRect rect;
        pObject->GetBoundingRect(&rect);
        myRECT obstacle;
        obstacle.p1.x = (float)rect.left;
        obstacle.p1.y = (float)rect.top;
        obstacle.p2.x = (float)rect.right;
        obstacle.p2.y = (float)rect.bottom;
        temp__distance = get__distance(&Robot, obstacle);
        minimum__distance = (temp__distance <= minimum__distance) ?
            temp__distance : minimum__distance;
    }
    distance[i] = minimum__distance;

    // new version : START
    // save shortest length betweer robot and obstacle within
    // sensor data
    if (distance[i] < nearest__distance) {
        nearest__distance = distance[i];
        nearest__point.x = Robot.p2.x;
        nearest__point.y = Robot.p2.y;
        nearest__index = i;
    }
    // new version : END

    /*
    // previous version : START
    // calculate reflected force (range : -100 ~ +100)
    if (distance[i] <= m__nDetectDistance) {
        //SpringX += (((Robot.p2.x - Robot.p1.x) / m__nDetectDistance) * float(10.3));
        //SpringY += (((Robot.p2.y - Robot.p1.y) / m__nDetectDistance) * float(10.3));
        SpringX += (Robot.p2.x - Robot.p1.x);
        SpringY += (Robot.p2.y - Robot.p1.y);
    }
    // previous version : END
    */
}
// previous version : START
//SpringX *= m__fSpring__Kcd;
//SpringY *= m__fSpring__Kcd;
// previous version : END
```

```
// new version : START
SpringX = SpringY = 0;
if (nearest__distance <= m__nDetectDistance) {
    // Reflected force to the X-axis
    // measured distance of X-axis
    SpringX = (float)fabs(Robot.p1.x - nearest__point.x);
    SpringX *= ((float)m__nDetectDistance -
        (float)nearest__distance);
    SpringX *= (100 / (float)m__nDetectDistance);
    SpringX /= nearest__distance;
    SpringX += (m__nRobotRadius *
        ((float)(m__nDetectDistance - nearest__distance)
        / (float)m__nDetectDistance) + 1);
    SpringX = (Robot.p1.x - nearest__point.x >= 0) ?
        SpringX : SpringX * -1;
    SpringX *= m__fSpring__Kcd;

    // Reflected force to the Y-axis

    // measured distance of X-axis
    SpringY = (float)fabs(Robot.p1.y - nearest__point.y);
    SpringY *= ((float)m__nDetectDistance -
        (float)nearest__distance);
    SpringY *= (100 / (float)m__nDetectDistance);
    SpringY /= nearest__distance;
    SpringY += (m__nRobotRadius * ((float)nearest__distance
        / (float)m__nDetectDistance) + 1);
    SpringY = (Robot.p1.y - nearest__point.y >= 0) ?
        SpringY : SpringY * -1;
    SpringY *= m__fSpring__Kcd;
}
// new version : END

// calculate DAMPER term
float MagnitudeX, MagnitudeY;
MagnitudeX = (prev1Robot.x != prev2Robot.x) ?
      (Robot.p1.x - prev1Robot.x) / (prev1Robot.x - prev2Robot.x) : 0;
MagnitudeY = (prev1Robot.y != prev2Robot.y) ?
      (Robot.p1.y - prev1Robot.y) / (prev1Robot.y - prev2Robot.y) : 0;

DamperX = (Robot.p1.x - prev1Robot.x) * MagnitudeX;
DamperY = (Robot.p1.y - prev1Robot.y) * MagnitudeY;

DamperX *= m__fDamper__Kb;
```

```
            DamperY *= m__fDamper__Kb;

/*
            ////////////////////////////////////////////////////
            // FOR FORCE TEST AGAINST WALLS (FOR DEMO)
            ReflectedForceX = int((Robot.p1.x - (MAP__X)/2) / 2.5);
            ReflectedForceY = int((Robot.p1.y - (MAP__Y)/2) / 2.5);
            pXYVectorForce(-VectorForceX, VectorForceY);
            ////////////////////////////////////////////////////
*/
            ReflectedForceX = SpringX;
            ReflectedForceX += -DamperX;
            ReflectedForceY = SpringY;
            ReflectedForceY += -DamperY;

            pXYVectorForce((int)ReflectedForceX, (int)-ReflectedForceY);
            draw__histogram();
            draw__robot__status();
        }
    }
    //draw__robot__status();

    if (recordFlag) {
        fprintf(outFile, "Joystick: %d,%d Robot: %03.0f,%03.0f  ReflectedForce: %03.0f,%03.0f  Spring:
%03.0f, %03.0f  Damper: %03.0f, %03.0f\n",
            JP.x, JP.y,
            Robot.p1.x, Robot.p1.y,
            ReflectedForceX, ReflectedForceY,
            SpringX, SpringY,
            DamperX, DamperY
        );
    }

    //CView::OnTimer(nIDEvent);
}

void CJoySimView::OnSetting()
{
    // TODO: Add your command handler code here
    CSetting dlg;

    dlg.m__nAngleStart = m__nAngleStart;
    dlg.m__nAngleEnd = m__nAngleEnd;
    dlg.m__nAngleStep = m__nAngleStep;
```

```
    dlg.m_nRobotRadius = m_nRobotRadius;
    dlg.m_nDetectDistance = m_nDetectDistance;
    dlg.m_bRobotTrace = m_bRobotTrace;
    dlg.m_fSpring_Kcd = m_fSpring_Kcd;
    dlg.m_fDamper_Kb = m_fDamper_Kb;


    if (dlg.DoModal() == IDOK) {
        m_nAngleStart = dlg.m_nAngleStart;
        m_nAngleEnd = dlg.m_nAngleEnd;
        m_nAngleStep = dlg.m_nAngleStep;
        m_nRobotRadius = dlg.m_nRobotRadius;
        m_nDetectDistance = dlg.m_nDetectDistance;
        m_bRobotTrace = dlg.m_bRobotTrace;
        m_nSpring = dlg.m_nSpring;
        m_nDamper = dlg.m_nDamper;
        m_fSpring_Kcd = dlg.m_fSpring_Kcd;
        m_fDamper_Kb = dlg.m_fDamper_Kb;
    }
}

void CJoySimView::OnEditUndo()
{
    // TODO: Add your command handler code here
    CJoySimDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    if (!pDoc->m_objectList.IsEmpty())
        pDoc->m_objectList.RemoveTail();

    CClientDC dc(this);
    CRect rect(0, 0, MAP_X, MAP_Y);
    CBrush newBrush(dc.GetBkColor());

    dc.SelectObject(&newBrush);
    dc.FillRect(rect, &newBrush);
    dc.SelectStockObject(WHITE_BRUSH);  // release newBrush

    dc.Rectangle(0, 0, 500, 500);        // map size
    // histogram size
    dc.Rectangle(MAP_X+20, 0, MAP_X+20+HISTOGRAM_X, HISTOGRAM_Y);

    for (POSITION pos = pDoc->m_objectList.GetHeadPosition(); pos != NULL;) {
        CDrawObject *pObject = (CDrawObject *)pDoc->m_objectList.GetNext(pos);
```

```
        pObject->DrawObject(&dc);
    }

    // draw_robot
    dc.Rectangle((int)Robot.p1.x-m_nRobotRadius,
        (int)Robot.p1.y-m_nRobotRadius,
        (int)Robot.p1.x+m_nRobotRadius,
        (int)Robot.p1.y+m_nRobotRadius
    );
}

void CJoySimView::draw_histogram() {
    int i, total_array = (m_nAngleEnd-m_nAngleStart)/m_nAngleStep;

    CClientDC dc(this);
    CRect rect(MAP_X+21, 1, MAP_X+10+HISTOGRAM_X, HISTOGRAM_Y-1);
    CBrush eraseBrush(RGB(0xff,0xff,0xff));
    dc.SelectObject(&eraseBrush);
    dc.FillRect(rect, &eraseBrush);

    CBrush newBrush(RGB(0,0,128));
    dc.SelectObject(&newBrush);

    for (i = 0; i < total_array; i++) {
        rect.left = (MAP_X+20) + (HISTOGRAM_X / total_array) * i + 2;
        if (distance[i] == INF)
            rect.top = HISTOGRAM_Y;
        else
            rect.top = distance[i] * (HISTOGRAM_Y / m_nDetectDistance);
            // rect.top = long((float(distance[i]) / (float)m_nDetectDistance) *
            // (HISTOGRAM_Y/*-10*/));
        rect.right = rect.left + 3;
        rect.bottom = HISTOGRAM_Y;
        dc.FillRect(rect, &newBrush);
    }

    dc.SelectStockObject(WHITE_BRUSH);  // release newBrush
}

void CJoySimView::draw_robot_status()
{
    // print robot info
    CClientDC dc(this);
    CString str;
```

```
    // center point of joystick
    str.Format("CenterJoyX = %.0f      ", CenterJoy.x);
    dc.TextOut(MAP_X+20, HISTOGRAM_Y+10, str);
    str.Format("CenterJoyY = %.0f      ", CenterJoy.y);
    dc.TextOut(MAP_X+20, HISTOGRAM_Y+30, str);

    // current point of joystick
    str.Format("Current X = %03d       ", JP.x);
    dc.TextOut(MAP_X+20, HISTOGRAM_Y+60, str);
    str.Format("Current Y = %03d       ", JP.y);
    dc.TextOut(MAP_X+20, HISTOGRAM_Y+80, str);

    // current velocity of robot
    str.Format("VelocityX = %03d       ", VelocityX);
    dc.TextOut(MAP_X+20, HISTOGRAM_Y+110, str);
    str.Format("VelocityY = %03d       ", VelocityY);
    dc.TextOut(MAP_X+20, HISTOGRAM_Y+130, str);

    // reflected force information
    // SPRING
    str.Format("SpringX = %3.0f       ", SpringX);
    dc.TextOut(MAP_X+20, HISTOGRAM_Y+160, str);
    str.Format("SpringY = %3.0f       ", SpringY);
    dc.TextOut(MAP_X+20, HISTOGRAM_Y+180, str);
    // DAMPER
    str.Format("DamperX = %3.0f       ", DamperX);
    dc.TextOut(MAP_X+20, HISTOGRAM_Y+210, str);
    str.Format("DamperY = %3.0f       ", DamperY);
    dc.TextOut(MAP_X+20, HISTOGRAM_Y+230, str);
    // TOTAL
    str.Format("ReflectedForceX = %3.0f       ", ReflectedForceX);
    dc.TextOut(MAP_X+20, HISTOGRAM_Y+260, str);
    str.Format("ReflectedForceY = %3.0f       ", ReflectedForceY);
    dc.TextOut(MAP_X+20, HISTOGRAM_Y+280, str);
}

myPOINT CJoySimView::get_reaching_point(myPOINT robot, int angle) {
    myPOINT pt;

    pt.x = float(robot.x + m_nDetectDistance * sin(toradian(angle)));
    pt.y = float(robot.y - m_nDetectDistance * cos(toradian(angle)));

    return pt;
```

```
}

myPOINT CJoySimView::find_intersect_point(struct myLINE l1, struct myLINE l2) {
    myPOINT intersect_point, min[2], max[2];
    float a, b;  // y1=ax1+b or x=b type
    float c, d;  // y2=cx1+d or x=d type

    intersect_point.x = INF;
    intersect_point.y = INF;

    // for check later, save the max-min position
    min[0].x = (l1.p1.x < l1.p2.x) ? l1.p1.x : l1.p2.x;
    max[0].x = (l1.p1.x > l1.p2.x) ? l1.p1.x : l1.p2.x;
    min[0].y = (l1.p1.y < l1.p2.y) ? l1.p1.y : l1.p2.y;
    max[0].y = (l1.p1.y > l1.p2.y) ? l1.p1.y : l1.p2.y;
    min[1].x = (l2.p1.x < l2.p2.x) ? l2.p1.x : l2.p2.x;
    max[1].x = (l2.p1.x > l2.p2.x) ? l2.p1.x : l2.p2.x;
    min[1].y = (l2.p1.y < l2.p2.y) ? l2.p1.y : l2.p2.y;
    max[1].y = (l2.p1.y > l2.p2.y) ? l2.p1.y : l2.p2.y;

    // get LINE-l1 type
    if (l1.p2.x == l1.p1.x)   {   // if vertical line (x=b type)
        a = INF;
        intersect_point.x = l1.p1.x;
    }
    else {
        a = (l1.p2.y - l1.p1.y) / (l1.p2.x - l1.p1.x);
        b = l1.p1.y - a * l1.p1.x;
    }

    // get LINE-l2 type
    if (l2.p2.x == l2.p1.x) { // if vertical line (x=d type)
        c = INF;
        intersect_point.x = l2.p1.x;
    }
    else {
        c = (l2.p2.y - l2.p1.y) / (l2.p2.x - l2.p1.x);
        d = l2.p1.y - c * l2.p1.x;
    }

    if (l1.p2.x == l1.p1.x && l2.p2.x != l2.p1.x) {
        intersect_point.y = c * intersect_point.x + d;
        if (
            intersect_point.x < min[0].x || intersect_point.x > max[0].x ||
```

```
            intersect_point.y < min[0].y || intersect_point.y > max[0].y ||
            intersect_point.x < min[1].x || intersect_point.x > max[1].x ||
            intersect_point.y < min[1].y || intersect_point.y > max[1].y
        )
            intersect_point.x = intersect_point.y = INF;
    }
    else if (l1.p2.x != l1.p1.x && l2.p2.x == l2.p1.x) {
        intersect_point.y = a * intersect_point.x + b;
        if (
            intersect_point.x < min[0].x || intersect_point.x > max[0].x ||
            intersect_point.y < min[0].y || intersect_point.y > max[0].y ||
            intersect_point.x < min[1].x || intersect_point.x > max[1].x ||
            intersect_point.y < min[1].y || intersect_point.y > max[1].y
        )
            intersect_point.x = intersect_point.y = INF;
    }
    else if (a != c) {  // get intersect point
        intersect_point.x = (d - b) / (a - c);
        intersect_point.y = a * ((d - b) / (a - c)) + b;
        if (
            intersect_point.x < min[0].x || intersect_point.x > max[0].x ||
            intersect_point.y < min[0].y || intersect_point.y > max[0].y ||
            intersect_point.x < min[1].x || intersect_point.x > max[1].x ||
            intersect_point.y < min[1].y || intersect_point.y > max[1].y
        )
            intersect_point.x = intersect_point.y = INF;
    }
    else
        intersect_point.x = intersect_point.y = INF;

    return intersect_point;
}

int CJoySimView::get_distance(myLINE *robot, myRECT obstacle) {
    myLINE robot_line = *robot, obstacle_line;
    myPOINT intersect_point = { INF, INF };
    int side;
    unsigned int distance;
    // unsigned int minimum_distance = m_nDetectDistance;  // previous version
    unsigned int minimum_distance = (unsigned int)INF;  // new version

    for (side = TOP; side <= RIGHT; side++) {
        if (side == TOP) { // (0)
            obstacle_line.p1.x = obstacle.p1.x;
```

```
            obstacle_line.p1.y = obstacle.p1.y;
            obstacle_line.p2.x = obstacle.p2.x;
            obstacle_line.p2.y = obstacle.p1.y;
        }
        else if (side == BOTTOM) {   // (1)
            obstacle_line.p1.x = obstacle.p1.x;
            obstacle_line.p1.y = obstacle.p2.y;
            obstacle_line.p2.x = obstacle.p2.x;
            obstacle_line.p2.y = obstacle.p2.y;
        }
        else if (side == LEFT) {    // (2)
            obstacle_line.p1.x = obstacle.p1.x;
            obstacle_line.p1.y = obstacle.p1.y;
            obstacle_line.p2.x = obstacle.p1.x;
            obstacle_line.p2.y = obstacle.p2.y;
        }
        else {  // RIGHT (3)
            obstacle_line.p1.x = obstacle.p2.x;
            obstacle_line.p1.y = obstacle.p1.y;
            obstacle_line.p2.x = obstacle.p2.x;
            obstacle_line.p2.y = obstacle.p2.y;
        }

        intersect_point = find_intersect_point(robot_line, obstacle_line);

        if (intersect_point.x != INF && intersect_point.y != INF) {
            distance = (int) sqrt(
                pow(robot->p1.x - intersect_point.x, 2) +
                pow(robot->p1.y - intersect_point.y, 2)
            );

            if (distance < minimum_distance) {
                minimum_distance = distance;
                robot->p2 = intersect_point;
            }
        }

    }

    return minimum_distance;
}

void CJoySimView::OnRecordStart()
{
```

```
    // TODO: Add your command handler code here
    CMenu* pMenu;
    pMenu = AfxGetApp()->m_pMainWnd->GetMenu();
    pMenu->EnableMenuItem(ID_RECORD_START, MF_BYCOMMAND | MF_GRAYED);
    pMenu->EnableMenuItem(ID_RECORD_END, MF_BYCOMMAND | MF_ENABLED);

    static char BASED_CODE szFilter[] = "Trace Files (*.txt)|*.txt|All Files (*.*)|*.*||";
    CFileDialog dlg(FALSE, "txt", "trace.txt", OFN_PATHMUSTEXIST, szFilter);
    if (dlg.DoModal() == IDOK) {
        outFile = fopen(dlg.GetPathName(), "wt");
        fprintf(outFile, "Start of Test\n");
    }

    recordFlag = TRUE;
}

void CJoySimView::OnRecordEnd()
{
    // TODO: Add your command handler code here
    CMenu* pMenu;
    pMenu = AfxGetApp()->m_pMainWnd->GetMenu();
    pMenu->EnableMenuItem(ID_RECORD_START, MF_BYCOMMAND | MF_ENABLED);
    pMenu->EnableMenuItem(ID_RECORD_END, MF_BYCOMMAND | MF_GRAYED);

    fprintf(outFile, "End of Test");
    fclose(outFile);

    recordFlag = FALSE;
}
```

서 지 정 보 양 식			
수행기관보고서번호	위탁기관보고서번호	표준보고서번호	INIS주제코드
KAERI/TR-980/98			

제목 / 부제	윈도우즈 환경에서의 원격로보트 제어시스템 개발
연구책임자 및 부서명 (AR, TR의 경우 주저자)	김 병 수 (원자력 로보트 Lab)
연구자 및 부서명	김승호('), 서용칠('), 김창회('), 황석용('), 김기호('), 정승호('), 이영광(')

출 판 지	대 전	발행기관	한국원자력연소	발행년	1998. 2. 27
페 이 지	87	도 표	유(o) , 무( )	크 기	19 x 26 cm
참고사항					

비밀여부	공개(o), 대외비( ), 급비밀	보고서종류	기 술 보 고 서
연 구 위 탁 기 관		계 약 번 호	

초록

중수로형 원자력발전소내 고방사능 지역과 같이 인간의 접근이 불가능한 지역에서 인간을 대신하여 감시작업을 수행할 수 있는 이동로보트의 효율적인 제어를 위하여 윈도우즈 환경 관리제어시스템을 개발하였다. 개발된 시스템은 크게 윈도우즈 메뉴 프로그램, 소켓을 이용한 통신 프로그램, 힘반향 조이스틱 프로그램 및 OpenGL을 이용한 3차원 그래픽 프로그램으로 구성하였다. 또한 조이스틱의 조작성을 향상시키기 위하여 이동로보트를 위한 힘반향 제어 알고리즘을 개발하였다.

설계된 윈도우즈형 관리제어시스템과 힘반향 제어 알고리즘의 효용성을 평가하기 위하여 가상환경에서 조이스틱 동작 테스트를 수행하였다. 테스트 결과 설계된 윈도우즈형 관리제어시스템의 효용성이 매우 크고 사용하기가 용이함을 알수 있었다. 또한 힘반향 제어알고리즘 실험결과 사람에 따라 다소 차이는 있으나 일반 원격제어 방식에 비하여 매우 효율적임을 확인할 수 있었다.

향후 본 연구에서 개발된 결과는 KAEROT의 차기 제품에 적용될 예정이다.

주제명키워드

원격이동로보트, 관리제어시스템, 윈도우즈 프로그래밍

BIOBLIGRAPHIC INFORMATION SHEET

Performing Org. Report No.	Sponsoring Org. Reoprt No.	Standard Report No.	INIS Subject Code
KAERI/TR-980/98			

Title / Subtitle	The Development of Windows based Control System for the Telerobotics
Project Manager and Dept. (or Main Author)	Byung-Soo Kim (Nuclear Robotics Lab)
Researcher and Dept.	Seungho Kim('), Yong Chil Seo('), Kiho Kim('), Suk Yeoung Hwang('), Chang Hoi Kim('), Seungho Jung('), Young Kwang Lee(')

Pub. Place	Taejon	Pub. Org.	KAERI	Pub. Date	1998. 3. 12
Page	87	Fig. and Tab.	Yes(o) , No ( )	Size	19 x 26 cm
Note					

Classified	Open(o), Outside ( ), Class	Report Type	Technical Report
Sponsoring Org.		Contack No.	

Abstract

The WSCS (Windows-based Supervisory Control System) has been developed for the efficient control of the mobile robot in the harzardous area, such as reactor surroundings of HPWR (Heavy Pressurized Water Reactor). The WSCS is basically computer program which consists windows menu-program, socket-based communication program, force reflection joystick program, and OpenGL-based 3D graphic program. Also, the WSCS includes the force reflection control algorithm of a master control device (in this case, joystick) for the enhanced operability.

To evaluate the effectiveness of the designed WSCS and the force reflection control algorithm, a series of experiments has been made in such a way that human operators command the desired motion of robot by manipulating the joystick in the virtual environment. As a result, it was proven that the designed WSCS is very easy-to-use and effective. Also, the developed force reflection algorithm is more efficient than that of general teleoperation, even though there are some differences in human dexterity.

In near future, the WSCS will be applied in the next version of KAEROT.

Subject Keyword(About 10 Words)

Tele-mobile Robot, Supervisory Control System, Windows Programming

HPWR