2007년 정보보호 졸업작품 보고서

# Security Guard (보안 요원)


팀명 :Black Chamber

팀원 :김일원

김학천

안용헌

정구혁

한상호



2007. 11

중부대학교 정보보호학과

# 요 약

화상카메라의 활용도를 생각해 보다가 ‘카메라로 CCTV처럼 감시를 할 수 있지 않을까’ 하고 생각을 해보았다.
그래서 여러 가지 조사를 해본 결과 웹캠으로 보안, 감시 기능이 있다는 것을 알게 되었다.
일반 적으로 웹캠으로 쓰이는 용도는 화상통신 및 채팅을 할 때 선명한 화질을 지원해주고 빠른 화면전송으로 생생한 화상 통신 기능, 동영상 촬영이나 디지털 이미지를 캡처 할 수 있는 기능 등의 여러 가지 사용자들의 편의를 위해서 기능이 추가되고 프로그램들이 향상되고 있는 실정이다.
저희 조는 학교 실습실에서 사용하고 있는 웹캠을 사용하였고, 캠의 사진, 동영상을 찍는 기본 기능을 이용하여 감시기능을 추가한 프로그램 만들었다.
기본바탕은 윈도우환경이고 프로그램 환경은 C++로 만든 알고리즘이다.
인터페이스 환경과 사용자가 편리하게 사용할 수 있도록 환경을 설정했으며, 빛의 3요소인 R(레드), G(그린), B(블루)를 적용하여 미리 찍어놓은 화면에 빛의 밝기를 비교 분석 하여 가장 작은 크기의 1픽셀의 오차를 확인하여 미리 저장해놓은 화면과 이상변동이 있을 때에 화면을 찍어서 사용자에게 이상 징후가 있는지를 판단할 수 있게 보여주도록 설정을 했다.
이전영상비교, 누적영상비교, 기본영상비교의 3가지로 구분해서 알고리즘을 작성하였으며, 각각 장단점이 있고 사용자가 골라서 사용할 수 있도록 해놓았다.
또한, 사진파일 뿐만 아니라 동영상 파일로도 저장하여 확인할 수 있고, 사용자가 부재중일 때에는 사용자의 혹은 다른 타인에게 메일을 보낼 수 있는 메일링 서비스 기능을 추가 하였다.

# 목 차

## 3. 구축 과정

# 그림 목차

# 1. 서　　론

## 1.1 연구의 필요성

감시카메라는 몇 년간 매우 빠르게 발전해왔고, 새로운 시스템, 새로운 경쟁자들, 새로운 제품, 그리고 새로운 응용분야 등을 통해 매일 그 속도가 증가하고 있다. 이러한 속도라면 전반적인 흐름과 미래에 대해 예측하기가 어렵게 될 것이다.

감시카메라는 쇼핑센터, 학교지역, 주차시설, 대중교통역사, 군대, 그리고 수많은 크고 작은 회사, 가정에서와 같이 우리의 생활 속에 깊게 자리 잡고 있다. 감시의 눈이 되어주고 치안과 도난에 안심할 수 있게끔 해주는 역할을 해 주고 있다. 이 모든 시설에 있어 경험상 드러난 법칙은 동일하다. 감시카메라를 모니터하는 한사람이 순찰을 하는 열 사람보다 더 많은 것을 효율성 있게 일을 처리 할 수 있다. 감시카메라가 없는 경우 감시구역을 시찰하는 사람이 필요하고 감시구역이 늘어나면 감시요원 또한 늘어난다. 감시카메라가 있어서 인력의 효율성을 높일 수 있는 것이다.

감시카메라의 사용 목적은 사람들의 존재를 탐지하는 것으로, 쇼핑센터와 같은 곳은 범죄의 예방, 허가받지 않은 사람이나 출입 불가지역 같은 곳에 있는 불법 침입자를 찾는 것이다.

## 1.2 과제의 개요, 목표, 내용

기존의 웹캠과 CCTV의 장점을 모아서 웹캠의 활용성을 높이고 CCTV의 감시기능을 더하여 기존의 CCTV로 찍은 수많은 영상자료의 방대함을 줄이고 캠을 이용해서 물체의 움직임을 감지하여 필요 할 때만 찍을 수 있게 하여 영상 또는 이미지 자료들의 양을 줄이는 효과를 얻기 위함이다.

현재 캠의 활용부분에서 대부분의 사람들이 화상 캠을 쓸 때 메신저로 사용을 많이 하고 있다. 그러므로 제품을 만드는 유명 많은 회사들은 차별화를 두지 않고 사용자들이 원하는 메신저로서의 기능과 이미지 사진 찍기 기능의 중요부분인 해상도와 카메라의 화소를 높이는 등의 제품들이 주를 이루고 있다. 차별화된 제품들이 종종 나오고 있는 상황이지만 아직 대부분의 사용자들은 사용을 하지 않고 있다.

캠으로 어느 한 지역을 기존영상, 누적영상, 이전영상을 비교 분석하여 오차가 있을 때 캠으로 동영상이나 이미지영상을 찍는 것이다. 자동메일 전송프로그램을 도입해서 일반보안요원이 24시간 지키고 있을 필요가 없도록 기존영상에 이상이 있을시 경보음과 동영상이나 이미지를 메일로 자동으로 보내지면 보안요원은 그것을 확인하고 신속히 상황을 처리할 수 있는 방식이다. C++기반으로 VFW로 연동하고, MFC함수를 사용하여 CallBack 함수의 영상비교 알고리즘을 통해서 화면을 비교하여 저장하는 방식이다.

# 2. 기반 기술(MFC)

## 2.1. 제품비교

**웹 캠** : PC기반의 USB 카메라를 말하며 주로 화상 채팅용으로 많이 사용된다. PC 의존 형이기 때문에 보안 감시용으로는 적합하지 않다.
PC와 함께 필요한 소프트웨어가 항시 동작되어야 하고 USB 특성상 거리에 제약이 있다.

**웹서버 카메라** : 일반적인 PC카메라와 달리 별도의 PC와 부가장비 없이 카메라에 랜 선만 꽂으면 바로 인터넷이나 로컬 상에서 화상을 받아 볼 수가 있으며, 원격지의 사용자들 역시 별도의 프로그램 없이 인터넷에 접속만 하면 WebEye가 제공하는 고화질의 화상을 볼 수 있다.

## 2.2. MFC란?

MFC (Microsoft Foundation Class Library)는 마이크로소프트에서 만든 윈도우 API를 C++로 둘러 싼 라이브러리이다. 클래스들은 윈도의 공용 컨트롤과 스마트 포인터를 사용하는 창 개체이다.

## 2.3. MFC의 역사

MFC 1.0 - 1992년 4월 (MS-C/C++ 7.0)
[윈도우와 관련된 클래스]
Window management : 윈도우 관리
Graphic Device Interface (GDI) : 그래픽 장치 인터페이스
Multiple Document Interface (MDI) : 다중 문서 인터페이스
Menus : 메뉴
Dialog boxes : 다이얼로그 박스
Windows controls : 윈도우 컨트롤
Windows common dialogs : 윈도우용 일반 다이얼로그
OLE (Object Linking & Embedding) 1.0 : 객체 연결과 삽입
Application services : 애플리케이션 서비스
[일반적인 목적의 클래스]
Run-Time Type Information : 실행 타입 정보
Object Persistence : 객체 보존성
Collection Classes : 집합체 클래스
Strings : 스트링
Files : 파일
Time and Date : 시간과 날짜
Exception Handling : 예외처리

MFC 2.0 - 1993년 8월 (VC++ 1.0 - 93년 2월)
[구조적 클래스]
Commands : 명령 관리 클래스들
Documents and Views : 도큐먼트와 뷰 (일명 "닥뷰")
Printing and Print Preview : 프린트와 프린트 미리보기
Dialog Data Exchange and Validation(DDX/DDV) : 다이얼로그 데이터 교환, 확인
Eontext-Sensitive Help : 도움말
[좀 더 편리하게 만들어진 비주얼 클래스]
Form View : 형태 뷰
Edit View : 편집 뷰
Scrolling View : 스크롤 뷰
Splitter Window : 분할 뷰
Toolbars and Status bar : 툴바와 상태 바
Dialog Bar and other Contorl Bars : 다이얼로그 바와 다른 컨트롤 바
VBX 1.0 control(16bit) : VBX 컨트롤

MFC 2.5 - 1993년 12월 (VC++ 1.5)
[데이터베이스 클래스]
Database Engine classes : 데이터베이스 엔진 클래스
Record Field Exchange (RFX) : 레코드 필드 교환
Record View : 레코드 뷰
[OLE 2.0 클래스]
Visual Editing servers : 비주얼 편집 서버
Visual Editing containers : 비주얼 편집 컨테이너
Drag and Drop Structured Stroage : 드래그 앤 드롭
OLE Automation servers : OLE 자동화 서버
OLE Automation clients : OLE 자동화 클라이언트

MFC 3.0 - 1994년 10월 (VC++ 2.0)
[User Interface 클래스]
Enhanced toolbars : 기능이 추가된 툴바
Miniframe windows : 미니프레임 윈도
Tabbed dialogs : 탭 다이얼로그 (탭에 의해 포커스 이동)
[Win32 지원]
New Win32 APIs : Win32 API 함수들 지원
Multithreading : 멀티 Thread 관련된 클래스
Unicode support : 유니코드 지원
Shared 32-bit DLLs : 공유된 32비트 DLL
[MFC 3.1 & 3.2에 추가된 클래스]

Windows95 Common Controls : Win95 컨트롤
Simple MAPI : 단순한 MAPI
Windows Sockets : 윈도우 소켓
Swap-tuned DLL versions : 최적화된 DLL 지원 (바이트 스와핑)

MFC 4.0 - 1995년 9월 (VC++ 4.0)
Containment of OLE Controls : OLE 컨트롤
DAO (Data Access Objects) : MFC 자체 데이터베이스 엔진
Simplified Windows 95 Common Controls : 간단한 Win95 일반 컨트롤
Windows 95 Common Dialogs : Win95 일반 다이얼로그
Thread synchronization Objects : 스레드 동기화 객체

## 2.4. App Wizard

프로그램의 틀을 자동적으로 생성해 주는 도구이다. 여기서 말하는 프로그램의 틀이란 흔히 스켈렉톤 코드(skeleton code)라고 말하기도 한다. VC++ 사용자는 App Wizard에서 알맞게 옵션을 지정하면 나머지는 VC++가 자동적으로 뼈대(skeleton)가 되는 코드를 포함하는 파일을 생성해 준다.
프로그래머들은 아무 일도 하지 않고 App Wizard를 이용하여 마우스 클릭만으로 새로운 애플리케이션의 뼈대를 구축할 수 있으므로 과거 SDK를 이용한 프로그래밍 보다 무척 편리하다.

MFC App Wizard-단계(1)
App Wizard-단계(1)에서는 어플리케이션의 유형과 리소스(resourse)에서 사용할 언어를 설정한다. 어플리케이션 유형에는 SDI어플리케이션, MDI어플리케이션, 다이얼로그 기반 어플리케이션이 있다.

| 구 분 | 내 용 |
|---|---|
| Single Document | 단일 문서를 작업할 수 있는 어플리케이션을 생성한다.<br>이를 SDI(SingleDocument Interface)어플리케이션이라고 한다.<br>예를 들어 메모장(Notepad.exe)이 여기에 해당된다. |
| Multiple Document | 다중문서를 작업할 수 있는 어플리케이션을 생성한다.<br>이를 MDI(Multiple Document Interface)어플리케이션이라고 한다.<br>예를 들어 시스템 편집기(Svsedit.exe)가 여기에 해당된다. |
| Dialog based | 다이얼로그 기반의 어플리케이션을 생성한다.<br>예를 들어 윈도우 기본 오락인 지뢰 찾기가 여기에 해당된다. |
| 리소스에 사용할 언어 | 콤보 박스에서 사용할 리소스 언어를 선택한다.<br>자신의 시스템에 설치되어 있는 DLL에 따라서 지원하는 언어 항목이 다르게 나타난다. |

MFC App Wizard-단계(2)

App Wizard-단계(2)는 데이터베이스 지원과 관련된 사양을 선택할 수 있다.

| 구 분 | 내 용 |
|---|---|
| None | Database를 지원하지 않는 (Database를 지원하는 클래스를 포함하지 않는다.) 어플리케이션을 생성한다. |
| Header files only | Database를 지원하는 클래스를 이용할 수 있도록 헤더파일(AFXDB.H)만 추가되므로 원하는 클래스를 직접 추가해서 사용해야 한다. |
| Database view without file support | Database를 지원하는 클래스(CRecordView나 CDaoRecordView에서 파생된 뷰와 CRecordSet이나 CDatRecordSet에서 파생된 레코드셋 클래스)를 포함하는 어플리케이션을 생성한다. 단 파일 지원이 되지 않는다. |
| Database view with file support | Database를 지원하는 클래스(CDRecordView나 CDaoRecordView에서 파생된 뷰와 CRecordSet에서 파생된 레코드 셋 클래스)를 포함하는 어플리케이션을 생성한다. |
| Data Sourse 버튼 | 데이터 소스이름(DSN) 또는 MDB의 위치를 설정하는 대화상자가 출력된다. 동시에 도큐먼트 클래스에 파일 지원을 위한 코드가 추가된다. |

MFC App Wizard-단계(3)

App Wizard-단계(3)에서는 OLE(Object linking & Embedding)지원과 관련된 옵션을 설정한다.

| 구 분 | | 내 용 |
|---|---|---|
| OLE compound document (OLE 복합문서) | None | OLE를 지원하지 않는 어플리케이션을 생성한다. |
| | Container | OLE를 컨테이너 어플리케이션을 생성한다. |
| | Mini-server | OLE복합 문서를 생성하고 관리하는 OLE서버 어플리케이션을 생성한다. 단, 독립적으로 실행이 되지 않는다.(컨테이너 프로그램 내에서만 수행된다.) |
| | Full-server | OLE복합 문서를 생성하고 관리하는 OLE서버 어플리케이션을 생성한다. Mini-server와 다른 점은 독립적으로 실행된다. |
| | Both container and server | OLE 컨테이너와 서버 역할을 동시에 하는 어플리케이션을 생성한다. **ActiveX document server** ActiveX문서를 생성하는 서버 어플리케이션을 생성한다. 이 어플리케이션에서 생성된 문서는 인터넷 익스플로러 3.0이상 버전에서 In-place 활성화가 된다. |
| OLE compound file | yes, please | 생성된 OLE컨테이너 어플리케이션이 OLE복합파일 형식으로 데이터를 저장할 수 있게 한다. |
| | no, Thank You | 생성된 OLE컨테이너 어플리케이션이 OLE복합 파일 형식으로 데이터를 저장할 수 없게 한다. |
| 다른 기능 지원 | automation | 자동화기능을 지원하는 어플리케이션을 생성한다. |
| | Active x | Active X 컨트롤을 사용 할 수 있는 어플리케이션을 생성한다. |

MFC App Wizard-단계(4)

App Wizard-단계(4)에서 사용자 인터페이스와 WOSA(Windows Open Services Architecture), 파일 확장자, 윈도우 스타일 등을 설정한다.

| 구 분 | | 내 용 |
|---|---|---|
| 사용자 인터페이스 옵션 | Docking toolbar | 도킹 툴바를 생성한다. |
| | Initial status bar | 상태 바를 생성한다. |
| | Printing and Printing Preview | 파일 메뉴에 인쇄와 미리보기 메뉴가 추가되고 관련 코드가 생성된다. |
| | Context-sensitive Help | 문맥 감지형 도움말을 지원하는 도움말 파일에 생성된다. |
| | 3D controls | 대화상자가 3차원으로 출력된다. |

MAPI

Message API를 지원하기 위한 헤더 파일이 추가되고 파일 메뉴에 Send(편지 보내기)메뉴와 관련 코드가 생성된다.

Windows Socket

Win socket API를 지원하기 위한 헤더 파일이 추가 된다.

MRU file list

최근에 사용한 파일 리스트 개수

MRU(Most Recently Used)는 가장 최근에 사용한 파일을 말하며 파일 메뉴에 지정한 개수만큼 파일 리스트가 추가된다.

Advanced 버튼

Advanced Options 다이얼로그가 출력된다.

| 구 분 | 내 용 |
|---|---|
| File extension | 파일 확장자를 지정한다. |
| File type ID | 레지스트리에서 사용되는 문서 타입 ID를 지정한다. |
| Main frame caption | 어플리케이션의 캡션 바에 출력될 제목을 지정한다. |
| Main frame caption | 새로운 도큐먼트 템플릿이 추가되면 File에서 New 메뉴 항목을 선택했을 때 다이얼로그가 출력되는데 이때 사용되는 타입 이름을 지정한다. |
| Filter name | 파일열기(open)나 다른 이름으로 저장(save as) 다이얼로그의 파일 형식 콤보 박스에 출력될 문자열을 지정한다. |
| File new name | OLE Object의 짧은 이름으로 사용된다. |
| File type name | 개체삽입 다이얼로그의 object type리스트 박스에서 사용되는 타입 이름을 지정한다. |

| 구 분 | | 내 용 |
|---|---|---|
| Use spliter window | | 분할 윈도우의 사용여부를 설정한다. |
| Main frame styles | Thick frame | 메인 프레임 윈도우의 크기를 조절할 수 있는지의 여부를 설정한다. |
| | system menu | 메인 윈도우의 시스템 메뉴 사용여부를 설정한다. |
| | Minimize box | 메인 윈도우에서 아이콘화 버튼의 사용여부를 설정한다. |
| | Minimized | 메인 윈도우를 아이콘화로 활성화할 것인지의 여부를 설정한다.(윈도우 95에서는 사용하지 않음) |
| | Maximize box | 메인 윈도우에서 최대화 버튼의 사용여부를 설정한다. |
| | Maximized | 메인 윈도우를 최대화로 활성화할 것인지의 여부를 설정한다.(윈도우 95에서는 사용하지 않음) |
| MDI child frame style | Thick frame | 차일드 프레임 윈도우의 크기를 조절할 수 있는지의 여부를 설정한다. |
| | Minimize box | 차일드 윈도우에서 아이콘화 버튼의 사용여부를 설정한다. |
| | Minimized | 차일드 윈도우를 아이콘화로 활성화할 것인지의 여부를 설정한다. |
| | Maximize box | 차일드 윈도우에서 최대화 버튼의 사용 여부를 설정한다. |
| | Maximized | 차일드 윈도우를 최대화로 활성화할 것인지의 여부를 설정한다. |

MFC App Wizard-단계(5)

App Wizard-단계(5)에서는 App Wizard에 의해 생성된 소스 파일의 형식을 설정한다.

| 구분 | | 내용 |
|---|---|---|
| Comments | Yes, please | App Wizard에 의해 생성된 소스 코드에 주석 문을 추가한다. |
| | No thank you | App Wizard에 의해 생성된 소스 코드에 주석 문을 추가 하지 않는다. |
| MFC library | As a shared DLL | 어플리케이션에 링크할 MFC라이브러리를 지정하는데 있어 공유 DLL을 사용하면 EXE의 크기가 작아지는 대신 별도의 DLL을 EXE와 함께 배포해야한다.(공유DLL) |
| | As a statically linked library | 라이브러리가 EXE에 포함되므로 EXE가 커지는 대신 별도의 DLL없이 실행 가능하다.(정적 링크 라이브러리) |

MFC App Wizard-단계(6)

마지막 단계인 App Wizard가 생성할 클래스명과 소스파일명을 변경할 수 있다. 특히 뷰 클래스의 경우는 Base class란에서 기초 클래스를 변경할 수 있다.

## 2.5. App Studio

사용자 인터페이스 요소들(메뉴, 다이얼로그, 핫키(단축키) 등)을 시각적으로 디자인할 수 있도록 도와준다. 보통 자원(resource)들을 편집하는 도구라고 생각하면 된다.

## 2.6. Class Wizard

App Studio에서 시각적으로 디자인된 요소들을 실제 프로그램 코드와 연결시키는 작업을 할 수 있다. 또한 새로운 클래스를 정의하는 데도 쓰인다.

윈도우 프로그램은 모든 것을 메시지 이벤트에 의존한다고 해도 과언은 아니다. 달리 말해 윈도우 프로그래밍은 어떤 메시지와 조건에 어떻게 작동하라는 정의를 내리는 것이라고 볼 수도 있다. 요는 메시지에 의해 프로그램이 구동된다.

FILE메뉴에서 PRINT SETUP을 선택했을 경우 WM_COMMAND라는 메시지가 발생하여 움직이게 되는 것이다.

MFC에서는 Class Wizard라는 것을 사용하여 메시지에 처리를 위한 함수를 쉽게 만들 수 있다. BEGIN MASSAGE MAP 메시지와 함수를 연결시키는 역할을 한다. Class Wizard가 없다면 사용자가 직접 입력을 해야 한다. Ctrl+W를 누르면 Class Wizard가 실행된다.

[Class Wizard 다이얼로그]

Object IDs 리스트 상자

Object IDs는 클래스명과 리소스에 등록된 ID를 말한다. App Studio를 실행시켜서, 즉 rc파일을 열어서 보면 모든 리소스들이 각각의 ID를 가지고 있다. 메뉴에서 FILE의 PRINT SETUP이란 메뉴도 그에 해당하는 ID가 있다.

Message 리스트 상자

ID들의 메시지를 말한다. ID_FILE_PRINT_SETUP에는 COMMAND와 UPDATE_COMMAND_UI이 두 가지 메시지 밖에 없다. 디폴트는 COMMAND이고 UPDATE_COMMAND_UI은 메뉴의 활성화, 비활성화를 설정할 수 있다.

Member Functions 리스트 상자

원하는 ID를 가지고 함수를 만들면 Member Functions에도 그 리스트가 만들어진다.

Class Name combo 상자

생성된 클래스 리스트가 등록되어 Combo 상자에 나타난다.

Add Class 버튼

새로운 클래스를 생성할 때 사용된다. 흔히 자신만의 새로운 다이얼로그를 만들어 사용하는 경우 이 버튼을 사용한다.

Delete Function
불필요하거나 잘못 함수를 넣었다가 지울 때 사용된다. 그러나 이것만으로 완전히 지워진 것은 아니다. View 코드의 Create 함수도 지워야 한다. 지우지 않으면 컴파일 에러 메시지와 함께 컴파일 끝난다.

```
BOOL CSafarisView::Create(LPCTSTR lpszClassName, LPCTSTR lpszWindowName, DWORD dwStyle, const RECT& rect, CWnd* pParentWnd, UINT nID, CCreateContext* pContext)
{
// TODO: Add your specialized code here and/or  call the base class
return CWnd::Create(lpszClassName, lpszWindowName, dwStyle, rect, pParentWnd, nID, pContext);
}
```

Member Variable
각 클래스에 사용자가 만든 멤버 변수나 객체를 보여주고 이것을 삭제, 추가, 편집을 할 수 있는 탭이다.

## 2.7. C Win App(자동생성 클래스 분석)

App Wizard 단계를 거치고 나면 Lyra프로젝트를 만든 후 자동적으로
C Win App -> CLyra App
CMDIFrameWnd -> CMainFrame
CDocument -> CLyraDoc
C View -> CLyraView인4개의 클래스가 만들어진다.
한 가지 덧붙이면 Work Space 창에서 클래스 탭을 클릭하면 현재 생성된 클래스들을 볼 수 있다.
클래스는 *.CPP, *.h 두 파일을 모두 보여준다. 클래스 이름을 클릭하면 *.h 파일을 볼 수 있고, 해당 클래스의 함수를 클릭하면 *.cpp 파일이 보이며 클래스를 수정해도 *.cpp, *.h 파일이 함께 자동 변경된다.
CLyraApp클래스
CWinApp 클래스이다. 애플리케이션 클래스로 CLyraApp란 클래스가 만들어지고 그것은 Lyra.cpp 파일에 기록된다. 이 CLyraApp 클래스는 CWinApp클래스를 상속받아서 사용한다.
즉, CWinApp 클래스의 기능을 물려받고 거기다가 개발자가 하고자하는 기능을 넣으면 된다.
CWinApp라는 것은 애플리케이션을 만들 때 필요한 클래스이다.
CLyraApp파일을 열어보면,

```
#include "stdafx.h"
#include "Lyra.h"
```

```
#include "mainfrm.h"
#include "Lyradoc.h"
#include "Lyraview.h"
#indef _DEBUG
#UNDEF THIS_FILE
ststic char BASED_CODE THIS_FILE[] = _FILE_;
#endif
BEGIN_MESSAGE_MAP(CLyraApp, CwinApp)
   //{{AFX_MSG_MAP(CLyraApp)
ON_COMAND(ID_APP_ABOUT, OnAppAbout)
   // NOTE-the Classwizard will add and remove
            mapping  macros here.
   // DO NOT EDIT what you see in these block of
            generated  code!
   //}}AFX_MSG_MAP
        // Standard file based document commands
        ON_COMMAND(ID_FILE_NEW,  CwinApp::OnFileNew)
        ON_COMMAND(ID_FILE_OPEN,  CwinApp::OnFileOpen)
        // Standard print setup command
        ON_COMMAND(ID_FILE_PRINT_SETUP,  CwinApp::OnFilePrintSetup)
END_MESSAGE_MAP()
BEGIN_MESSAGE_MAP(CLyraApp, CwinApp),
END_MESSAGE_MAP()
```

BEGIN_MESAGE_MAP 부분에서 이상한 점은 세미콜론(;)이 없다는 것이다.

Visual C++는 윈도우 프로그래밍이다. 즉 모든 실행을 메시지로 처리한다는 뜻이다.

```
BEGIN_MESSAGE_MAP(CLyraApp, CwinApp)
```

//CLyraApp 클래스는 CwinApp 클래스를 상속받았으므로 메시지 발생 시 다음 줄의 함수를 실행하라는 의미이다.

```
ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
```

//ID_APP_ABOUT란 일종의 이름이고, 1개의 프로젝트에 같은 이름이 존재해서는 안 되며 유일해야만 한다. 즉, 프로젝트에 사용되는 개체와 연결고리인 것이다. AboutDEMO1란 팝업 메뉴를 선택하면 OnAppAbout 함수를 실행하라는 의미이다.

```
ON_COMMAND(ID_FILE_NEW, CwinApp::OnFileNew)
```

//팝업 메뉴에서 File -> New를  선택하면 CwinApp 클래스의 OnFileNew 함수를 실행하라는 의미이다.

ON_COMMAND(ID_FILE_OPEN, CwinApp::OnFileOpen)
//팝업 메뉴에서 File->Open을 선택하면 CwinApp 클래스의 OnFileOpen 함수를 실행하라는 뜻이다.
ON_COMMAND(ID_FILE_PRINT_SETUP, CwinApp::OnFilePrintSetup)
//팝업 메뉴에서 File->PrintSetup을 선택하면 CwinApp 클래스의 OnFilePrintSetup 함수를 실행하라는 뜻이다.
END_MESSAGE_MAP()
위에서 알 수 있듯이 기본적으로 생성된 프로그램에는 보통 File메뉴와 Edit메뉴, Window메뉴, Help메뉴가 들어간다. 또한 위의 4개의 함수는 CWinApp에서 제공되는 기본 함수이다. 원하는 다른 함수를 동작시키려면 위의 메시지 매핑에 사용자가 직접 써넣어야 한다.
다른 함수는 CLyraApp의 생성자이다. 이것은 이 클래스가 생성됨과 동시에 자동적으로 실행되는 함수이다. 이 클래스는 프로그램이 시작됨과 동시에 실행된다. 그런데 안에는 아무것도 없다. 즉 현재는 생성 시 아무것도 하지 않겠지만 후에 어플리케이션 클래스가 생성될 때 어떤 일을 해야 한다면 이 함수 안에 기록하면 된다.
CLyraApp::CLyraApp()
{
// TODO add construction code here,
// place all significant initialization in InitInstance
}
위의 함수는 CLyraApp의 생성자이다.
CLyraApp 클래스가 실행됨과 동시에 자동적으로 실행되는 함수이며 프로그램이 시작과 동시에 실행된다. 생성자와 같이 CLyraApp 클래스의 실행과 함께 CWinApp::InitInstance() 함수가 작동한다.
그리고 상속받은 것을 무시하려면 헤더에 다음과 같이 등록한다.
virtual BOOL InitInstance();
아래는 사용자 인터페이스 옵션 항목에서 3D Controls를 Enable했을 때,
BOOL CLyraApp::InitInstance()
{
#ifdef _AFXDLL
Enable3dControls(); // Call this when using MFC in a shared DLL
#else // Call this when linking to MFC statically
Enable3dControlsStatic();
#endif
SetRegistryKey(_T("Local AppWizard-Generated Applications"));
// 위의 값은 윈도우 레지스트리에 저장된다.

LoadStdProfileSettings(); //Load standard INI file options (including MRU)
// 위 또한 AppWizard 4단계에서 사용자 인터페이스 옵션 항목에서 MRU File List 항목에서 개수를 넣었을 것이다. 최근 열어본 파일을 List화하여 사용하기 편하게 팝업 메뉴에 추가해 놓은 것이다.
}

```
CSingleDocTemplate* pDocTemplate;
pDocTemplate = new CSingleDocTemplate(
        IDR_MAINFRAME,
        RUNTIME_CLASS(CLyraDoc),
        RUNTIME_CLASS(CMainFrame),  // main SDI frame window
        RUNTIME_CLASS(CLyraView));
AddDocTemplate(pDocTemplate);
```

// Parse command line for standard shell commands, DDE, file open
이 프로젝트를 SDI로 선택했었다. 그렇다면 MDI처럼 사용될 데이터를 저장할 공간이 있어야 하는 것은 아니다. 단, SDI에 필요한 하나의 공간을 설정해 주는 클래스가 CSingleDocTemplate이다.
pDocTemplate라는 가상적인 CSingleDocTemplate 형의 변수를 선언하고 이것이 생성될 때는 CLyraDoc 클래스인 Document와 CMainFrame 형태인 틀과 CDemo1View인 화면이 링크되면서 RC(리소스) 파일의 String Table 안에 설정된 IDR_MAINFRAME형으로 연결된다.
도스 창 에서 프로그램을 실행시켰을 경우나 또는 DDE나 파일 설정 등으로 실행했을 때 넘겨주는 인자를 받는 내용이다.

```
CCommandLineInfo cmdInfo;
ParseCommandLine(cmdInfo);
```

위의 내용은 CCommandLineInfo라는 클래스 변수 cmdInfo를 만들고 현재 프로그램이 실행되면서 어떤 인자가 들어왔나 조사하기 위한 ParseCommandLine라는 함수를 실행하는 것이다. 또한 AddDocTemplate 함수를 이용하여 여러 개의 Doc와 View와 Frame이 연결된 템플리트를 만들어 등록하였을 경우에도 cmdInfo안에 이 정보가 들어간다. 아래는 정보를 실행시키는 것이다.

```
// Dispatch commands specified on the command  line
   if  (!ProcessShellCommand(cmdinfo))
            return  FALSE;
```

ProcessShellCommand 함수가 실행되면서 OnFileNew()라는 함수가 실행된다. 이 함수가 실행되면 프레임과 도큐먼트 그리고 View가 만들어진다.

```
// The main window has been initialized, so show and update it.
pMainFrame->ShowWindow(SW_SHOW);
pMainFrame->UpdateWindow();
```

아래의 소스는 About 팝업 메뉴를 클릭했을 때의 실행코드이다.
CAboutDlg도 하나의 클래스이며 C Dialog 클래스를 상속받았다.

```
// CAboutDlg dialog used for App About
class C About Dlg : public C Dialog
{
public:
CAboutDlg();
// Dialog Data
//{{AFX_DATA(CAboutDlg)
enum { IDD = IDD_ABOUTBOX };
//}}AFX_DATA
// ClassWizard generated virtual function  overrides
//{{AFX_VIRTUAL(CAboutDlg)
protected:
virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV support
//}}AFX_VIRTUAL
// Implementation
protected:
//{{AFX_MSG(CAboutDlg)
// No message handlers
//}}AFX_MSG
DECLARE_MESSAGE_MAP()
};
CAboutDlg::CAboutDlg() :  CDialog(CAboutDlg::IDD)
{
//{{AFX_DATA_INIT(CAboutDlg)
//}}AFX_DATA_INIT
}
void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
CDialog::DoDataExchange(pDX);
//{{AFX_DATA_MAP(CAboutDlg)
//}}AFX_DATA_MAP
}
BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
//{{AFX_MSG_MAP(CAboutDlg)
// No message handlers
```

```
//}}AFX_MSG_MAP
END_MESSAGE_MAP()
// App command to run the dialog
void CLyrasApp::OnAppAbout()
{
CAboutDlg aboutDlg;
aboutDlg.DoModal();
}
```

바로 위의 OnAppAbout() 함수에 의해 대화상자가 사용자에게 보인다.
aboutDlg.DoModal();의 의미는 프로그램에서 About 대화상자를 열었을 경우 이 대화상자를 종료하기 전까지는 About 대화상자를 벗어나 이 프로그램내의 다른 명령을 내릴 수 없다는 뜻이다.

## 2.8. CMainFrame

```
BEGIN_MESSAGE_MAP(CMainFrame. CFrameWnd)
//{{AFX_MSG_MAP(CMainFrame)
// NOTE - the Classwizard will add and remove
mapping macros here.
// DO NOT EDIT what you see in these blocks
of generated code !
ON_WM_CREATE()
//}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

아래의 indicators는 상태 창에 나타나는 string의 ID값이다.
ID_SEPARATOR는 라인을 그어서 분리시키고 ID_INDICATOR_CAPS는 CapsLock키를, ID_INDICATOR_NUM은 NumLock키를, ID_INDICATOR_SCRLS는 SrollLock키를 각각 ON/OFF 상태를 확인하는 ID이다.
프로그램 실행 시 키들을 눌러보면 상태 창 오른쪽에 나타난다.

```
static UNIT indicators[] =
{
ID_SEPARATOR, //status line indicator
ID_INDICATOR_CAPS,
ID_INDICATOR_NUM,
ID_INDICATOR_SCRL,
};
```

메인 프레임의 생성자입니다. 현재는 비어 있는 상태입니다.

```
CMainFrame::CMainFrame()
{
// TODO: add member initialization code here
}
```

메인프레임의 소멸자다.

```
CMainFrame::~CMainFrame()
{
}
```

OnCreate 함수는 메인 프레임이 만들어질 때 실행된다. 아버지 클래스인 CFrameWnd것을 사용하지 않고 자기 것, 즉 virtual로 사용한다는 것이다.

```
int CMainFrame::OnCreate(LPCREATESTRUCT 1pCreateStruct)
{
//아버지 클래스의 OnCreate를 먼저 실행하고
if (CFrameWnd:: OnCreate(1pCreateStruct) == -1)
return -1;
//도구 바를 만들고
if (!m_WndToolBar.Create(this) ||
!m_WndToolBar.LoadToolBar(IDR_MAINFRAME))
{
TRACE0("Failed to create toolbar₩n");
return -1; // fail to create
}
//상태 창을 만들고
if (!m_wndStatusBar.Create(this) ||
!m_wndStatusBar.SetIndicators(indicators,
sizeof(indicators)/sizeof(UNIT)))
{
TRACE0("Failed to create status  bar₩n");
return -1; // fail to create
}
```

아래의 코드는 메뉴 바를 도킹 가능한 메뉴 바로 만드는 코드이다.

```
m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
EnableDocking(CBRS_ALIGN_ANY);
```

```
DockControlBar(&m_wndToolBar);
m_wndToolBar.SetBarStyle(m_wndToolBar.GetBarStyle() |
CBRS_TOOLTIPS | CBRS_FLYBY);
return 0;
}
```

CWinApp클래스에서 CMainFrame 프로그램이 실행될 수 있도록 프레임을 로드하였다.

CMainFrame의 OnCreate 함수는 CWinApp의 Initinstance 함수의 if(!pMainFrame->LoadFrame(IDR_MAINFRAME))가 실행되면 후에 수행되는 함수다. OnCreate 함수는 실행될 때  필요한 여러 가지 일을 수행 하는 함수이다. 위의 내용은 메인 프레임에 있는 메뉴와 도구 바 그리고 상태 창 등을 메인 프레임에 등록하는 것이다. 지금까지 내용으로 CWinApp 클래스에서 메인 프레임을 만들고 CMainFrame에 도구 바와 메뉴와 상태 바를 등록했다.

## 2.9. CDocument 클래스

CLyraDoc 클래스는 CDocument 클래스를 상속받은 것이다.

CDocument 클래스는 일종의 데이터 임시 저장소이다. CLyraDoc와 CView는 연결되어 있다. 이것을 확인하려면 LyraView.cpp에서 찾아볼 수 있을 것이다. 즉, VIEW 창에 나타내기를 원하는 것을 CDocument 클래스에 넣으면 된다.

```
BEGIN_MESSAGE_MAP(CLyraDoc, CDocument)
//{{AFX_MSG_MAP(CLyraDoc)
// NOTE - the Classwizard will add and remove mapping macros here.
// DO NOT EDIT what you see in these blocks of generated code!
//}}AFX_MSG_MAP
END_MESSAGE_MAP()
CLyraDoc::CLyraDoc()
{
// TODO: add one-time construction code here
생성자
}
CLyraDoc::CLyraDoc()
{
소멸자
}
```

OnNewDocument 함수는 팝업메뉴 new나 OnFileNew() 함수를 실행하여서 새로운 파일이 생성될 때 실행되는 함수다.

지금 현재는 아무것도 없으므로 아버지 클래스인 CDocument만 실행된다.

```
BOOL CLyraDoc::OnNewDocument()
{
if (!CDocument::OnNewDocument())
return FALSE;
// TODO: add reinitialization code here
// (SDI documents will reuse this document)
return TRUE;
}
```

Serialize 함수는 메모리에 있는 데이터를 저장장치와 연결시키거나 열어주는 함수다.

```
void CLyraDoc::Serialize(CArchive& ar)
{
if (ar.IsStoring())
      {
// 파일로 저장할 때
}
else
{
// 파일에서 불러올 때
      }
}
```

## 2.10. CView

CLyraView 클래스는 CView 클래스를 상속받은 것이다.
우리가 잘 쓰는 메모장에서 실제 우리가 텍스트를 쓰는 하얀 영역을 말한다.
메시지 매핑

```
BEGIN_MESSAGE_MAP(CDEMO1View, CView)
//{{AFX_MSG_MAP(CDEMO1View)
ON_WM_LBUTTONDOWN()
//}}AFX_MSG_MAP
// Standard printing commands
ON_COMMAND(ID_FILE_PRINT, CView::OnFilePrint)
ON_COMMAND(ID_FILE_PRINT_PREVIEW,
CView::OnFilePrintPreview)
END_MESSAGE_MAP()
```

생성자 함수

```
CDEMO1View::CDEMO1View()
{
// TODO: add construction code here
}
```
소멸자 함수
```
CDEMO1View::~CDEMO1View()
{
}
```
함수 OnDraw안에 화면에 뿌려줄 기능을 추가하면 된다. 화면에 출력할 여러 정보는 CDocument 클래스에 추가한다고 앞 페이지에서 설명했다.
```
void CDEMO1View::OnDraw(CDC* pDC)
{
```
일단 도큐먼트와 연결을 한다.
```
CDEMO1Dov* pDoc = GetDocument();
```
ASSERT_VALID라는 함수는 pDoc가 활성화되어 있는 것을 보증하는 함수다. 프로그램이 실행되다가 오류가 발생하여 링크가 되지 않아 도큐먼트를 받지 못하면 pDoc가 null값이 되므로 '응용프로그램 에러'라는 메시지를 출력하고 프로그램을 종료하라는 뜻이다.
```
ASSERT_VALID(pDoc);
// TODO: add draw code for native data here
}
```
프린팅 미리보기 기능을 수행하는 함수이다. 이 기능은 프로젝트 시작 시 AppWizard 4단계에서  Printing and Print Preview 옵션을 선택했기 때문이다.
```
BOOL CDEMO1View::OnPreparePrinting(CPrintInfo* pInfo)
{
//default preparation
return DOPreparePrinting(pInfo);
}
//프린팅 하는 함수
void CDEMO1View::OnBeginPrinting(CDC* /*pDC*/,
CPrintInfo* /*pInfo*/)
{
//TODO: add extra initialization before  printing
}
//프린팅이 끝났을 때 사용하는 함수
void CDEMO1View::OnEndPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
//TODO: add cleanup after printing
}
```

프로젝트를 만들면 생성되는 4개의 클래스에 대해서 간단히 설명했다. CWinApp 클래스가 먼저 실행되어 메인 프레임을 만들고 CMainFrame 클래스가 수행되면서 메뉴와 도구 바, 상태 바를 연결하고 CWinApp 클래스가 다시 View와 Document를 생성한다. 이렇게 하여 CView와 CDocument 클래스인 CLyraView와 CLyraDoc가 수행되는 것이다.

## 2.11. MFC 기본 프로그래밍

1. CString

텍스트 데이터들을 저장하고 컨트롤하는 클래스이다. 보통 텍스트 String을 저장하는 변수 중 우리가 가장 쉽게 알 수 있는 변수는 char* text, char text[80] 등으로 표현하고  저장을 할 수 있는 반면 CString 함수는 더 많은 기능을 제공한다.

CString m_strText;
char temp[80];
lstrcpy((LPSTR)temp, (LPSTR)"여기는 SAFARI 사이트입니다");
m_strText = " CString("여기는")" + "SAFARI" => "여기는  SAFARI"
m_strText = m_strText + "사이트입니다"  // "여기는 SAFARI 사이트입니다"
m_strText.GetLength()// 위의 텍스트 길이 값 26을 리턴 한다.
m_strText.Empty;=>  m_strText =  "" 가 된다.
int CString::GetLength()
현재 설정된 문자열의  길이를 리턴 한다.
ex)CString test = CString("안녕");
int len=test.GetLength();
BOOL CString::IsEmpty()
//현재 문자열을 검사하여 없으면 TRUE 한 개의  문자라도 있으면 FALSE를 리턴 한다.
void CString::Empty()
//현재 저장된 문자열을 모두 지운다.

2. CPoint

CPoint 함수는 x,y 좌료를  저장하고 응용하는 클래스이다.
ex) CPoint po=CPoint(30,20);

3. CRect

사각형 좌표를 저장하고  응용하는 클래스 입니다.
즉 사각형의 왼쪽 좌표(left), 상단좌표(top), 오른쪽 좌표(right), 하단좌표(bottom)값을 가지고 기능을 하는 클래스이다.
CRect m_pRectText;
m_pRectText=CRect(0,0,640,480);

//left=0, top=0, right=640, bottom=480이란 뜻이다.
int CRect::Width()  //현재 사각형의 가로 길이를 리턴.
int CRect::Height()  //현재 사각형의 세로 길이를 리턴.
CSize CRect::Size()
현재 사각형의 가로 길이와 세로 길이를 CSize  형태로 함께 리턴.
CPoint& CRect::TopLeft()
현재 사각형의 좌측 상단 좌표를 CPoint형으로 리턴.
CPoint& CRect::BottomRight()
현재 사각형의 우측 하단 좌표를 CPoint형으로 리턴.
CPoint& CRect::CenterPoint()
현재 사각형의 중심 좌표를 CPoint형으로 리턴.

4. CView::OnDraw(CDC* pDC)
화면이나 프린트 DEVICE에  출력할 때 사용하는 함수이다.
즉, Device Context 클래스(CDC)를 받아서 화면 출력을 설정하면 바로 화면에 뿌려진다.

5. CDC::TextOut()
화면에 글자를 출력 함수. 두 가지 형태가 있다.
Virtual BOOL TextOut( int x, int y, LPCTSTR lpszString, int nCount );
//LPSTR형의 스트링을 화면에 뿌림.
BOOL TextOut( int x, int y, const CString& str );
// CSring형의 스트링을 화면에 뿌림.

6. CDC::Rectangle()
화면에 상자를 그리는 함수이며 두 가지 종류의 함수로 되어 있다.
BOOL Rectangle( int x1, int y1, int x2, int y2 );  //int형 좌표 값
BOOL Rectangle( LPCRECT lprect );  //CRect형 좌표 값

7. CDC::Ellipse()
화면에 원을 그리는 함수다. 이것도 두 가지 종류로 되어 있다.
BOOL Ellipse( int x1, int y1, int x2, int y2 );  //int형 좌표 값
BOOL Ellipse( LPCRECT lpRect );  // CRect형 좌표 값

8. OnLButtonDown(UINT nFlags, CPoint point)
좌측 마우스 버튼을 누르면 작동하는 함수이며, 리턴 되는 인자 중 nFlags는 좌측 마우스 버튼과 함께 어떤 키가 눌려졌는지를 확인한다.
MK_CONTROL  //Ctrl키가  눌려졌다.
MK_LBUTTON  //LButton만 눌려졌다.
MK_MBUTTON  //중간마우스 버튼이  눌려졌다.
MK_RBUTTON  //우측 마우스 버튼이  눌려졌다.
MK_SHIFT  //Shift키가  눌려졌다.
Point 안에는 현재 마우스의 좌표 값이 저장된다.

9. InvalidateRect()
OnDraw() 함수를 호출하며, 리턴 값이 TRUE이면 배경화면까지 다시 출력하고 FALSE이면 나머지 부분만 출력한다.

10. Invalidate()
이 함수는 전체를 재 표시 해주고 인자가 FALSE이면 배경화면색을 제외한 나머지 부분을  재 출력 한다.
(OnDraw(), OnPaint())
Invalidate(FALSE);// 화면의 배경색은  그대로 놔두고 재출력
Invalidate() 또는 Invalidate(TRUE);// 화면을  배경색부터 재출력

11. GDI 클래스
GDI를 지원하는 클래스는 CDC이다. 그래픽 장치를 조절하는 클래스이므로 이것을 잘 이용하면 멋진 화면을 만들 수 있다는 것이다. 또한 사용자가 그래픽 장치를 더 세밀하고 전문적으로 조절할 수 있도록 CDC의 파생 클래스가 있다.
CDC *pDC = GetDC();// 생성
ReleaseDC(pDC);// 소멸
소멸시키지 않을 경우 에러가 발생한다. 그러므로 반드시 소멸시켜야하지만 OnDraw() 함수 내에서는 자동으로 소멸된다. 그러나 WM_LBUTTONDBLCLK과 같은 메시지 함수를 사용할 경우에는 반드시 소멸시켜주어야 한다.
그러나 CClientDC를 사용하면 소멸시키지 않아도 된다.
ClientDC
윈도우의 캡션 바, 외곽 툴, 상태 창 등을 제외한 실질 출력되는 화면을 클라이언트 영역이라고 한다. 이 영역 외의 부분은 그래픽 출력을 할 수 없다.
CPaintDC
WM_PAINT메시지가 발생되면 실행되는 OnPaint함수에서 사용된다.
CWindowDC
클라이언트 영역과 윈도우의 캡션 바, 외곽 툴, 상태 창 등을 모두 포함한다. 그러므로 윈도우 캡션 바에도 그림을 그릴 수 있다. EX) CWindowDC
*pDC=GetWindowDC();
//DC 생성
GDI안에 내장 객체
점, 선, 면, 원, 글자, 비트맵 등을 자유롭게 사용할 수 있는 객체를 제공한다.
CPen선이나 도형의 테두리를 그릴 때 사용한다. 색과 굵기, 선의 종류도 지정할 수 있다.

CBrush
면을 색칠하는 데 사용된다.
CPalette
윈도우에서 색상은 RGB값 즉 Red, Green, Blue색의 256단계를 서로 혼합해서 사용한다. 이 클래스는 비디오 메모리가 부족하거나 색상의 질에 의존하지 않는 프로그램을 위한 클래스라고 보아도 좋다. 즉, 필요한 256색을 정하여 팔레트를 만들어 사용할 수 있게 해주는 클래스다.
CBitmap
이미지를 출력에 있어 한 픽셀 당 해당하는 컬러가 따로따로 있는데 픽셀 대 비트의 배열을 조절하는 클래스가 바로 CBitmap 클래스다.
CFont
윈도우에는 많은 폰트들이 등록되어 있다. 폰트를 로드하고 출력할 때 사용하는 클래스다.
CRgn
다각형이나 원 등은 어떤 범위를 가지므로 이 범위를 설정하고 그 안에 색칠하고 클리핑 하는데 이용하는 클래스.
BOOL CPalette::CreatePalette( LPLOGPALETTE 1pLogPalette);
팔레트를 만드는 함수.
int CPalette::GetEntryCount();
현재 팔레트 클래스 안에 설정되어 있는 색의 수를 리턴.
CDC::Framerect(CRect ,CBrush *);
외곽선만 있는 상자를 그리는 함수다.

12. 매핑 모드
매핑 모드를 하려면 OnPrepareDC 또는 OnDraw에서 매핑 모드를 설정해 주면 된다. 만일 MM_ANISOTROPIC으로 설정한다면
pDC->SetMapMode(MM_ANISOTROPIC);
MM_TEXT//(poxel) y축 아래로 증가
MM_LOMETRIC // y축 아래로 감소(-)
pDC->SetWindowExt(1000,1000);
pDC->SetViewportExt(1000,1000);
설정하고 화면에 출력할 때는 LPtoDp->DPtoLp해야 합니다.

13. 화면 스케일
화면 스케일 변환 모드
MM_ANISOTROPIC //비율이 변해도 됨
MM_ISOTROPIC// 비율을 1:1로 유지해야 됨
- 스케일 변경
pDC->SetMapMode(MM_ANISOTROPIC);
pDC->SetWindowsExt(1000,1000);
pDC->SetViewportExt(1000, -1000);// (증가, 감소)
- 원점의 위치 변경
CDC::SetWindowOrg();// 윈도우 자체이동
CDC::SetViewportOrg();// 화면의 보이는 좌표가 이동

14. 좌표

윈도우 프로그래밍에서의 좌표는 장치좌표와 로직좌표가 있다. 장치좌표는 윈도우의 그래픽 좌표를 말하며 로직 좌표는 다른 장치(마우스)가 갖는 좌표이다.

- 좌표계 전환 함수

CDC::DPtoLP(CRect m_rect);

CDC::LPtoDP(CRect m_rect);

15. SetTimer 함수

SetTimer(시계번호, 시간(msec), NULL)와 Ontimer() 함수를 함께 사용할 수 있다. SetTimer의 정의된 시간이 되면 OnTimer() 함수가 실행된다.

시간 해제 함수는 Killtimer(시계번호)이다.

16. 커서를 만들기

WorkSpace의 ResourceView 탭을 열고 Tree 구조에서 오른쪽 마우스 클릭을 하여 Insert 메뉴를 선택한다. 그러면 다이얼로그 대화상자가 나타나고 여기서 Cursor를 선택하면 편집창이 열리는데 여기서 원하는 아이콘을 그린다. 다 그린 후 ID값을 원하는 값으로 바꾸면 된다.

17.Document 클래스 얻기

현재 View와 연결된 Document 클래스를 얻고자 할 때는 아래와 같이 입력하면 된다. CMyDoc* pDoc = GetDocument();

## 2.12. 평면 툴바 클래스

평면 툴바는 MS-OFFICE에서 처음 채용된 툴바의 형태이다. 기존의 튀어나와 있는 툴바와는 달리 평면 상태에서 마우스가 버튼 위에 위치하면 토글 되는 툴바다.

이 클래스의 사용법은 비교적 간단하다. 툴바의 생성은 CMainFrame 클래스에서 이루어진다.

먼저 헤더파일에 CToolBar m_wndToolBar;와 같이 선언되는 툴바의 객체를 CToolBarEx 로 고친다.

CToolBarEx m_wndToolBar;

그리고, 구현 파일의 OnCreate 함수에 다음과 같은 생성코드를 집어넣으면 간단히 평면 툴바를 구현할 수 있다.

```
if (!m_wndToolBar.Create(this) || !m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
{
TRACE0("Failed to create toolbar₩n");
return -1; // fail to create
}
```

m_wndToolBar.SetBarStyle(m_wndToolBar.GetBarStyle() | CBRS_TOOLTIPS | CBRS_FLYBY | CBRS_SIZE_DYNAMIC);
m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
EnableDocking(CBRS_ALIGN_ANY);
DockControlBar(&m_wndToolBar);

## 2.13. 평면 버튼 클래스

일반적인 버튼은 돌출되어 있는 입체형의 형태를 보여준다. 그러나 이 클래스는 일반적인 버튼과는 달리 평면 형태이고 마우스가 버튼 위로 가면 토글 되는 버튼을 생성하는 클래스다.
클래스 명은 CHotButton 이고 Hotbtn.h와 Hotbtn.cpp 파일로 구성되어 있다.
<사용법>
리소스 뷰에서 다이얼로그에 버튼 컨트롤을 올려놓는다. 주의할 점은 버튼 속성에 styles탭에 Owner Draw 속성을 체크해야 한다.
클래스 위저드를 열고 버튼의 객체변수를 만든다.(Member Variables 탭) 변수 이름은 적당히 주고 목록은 Control을 선택, 변수 Type은 CButton 으로 설정한다.
클래스 위저드를 닫고 다이얼로그 클래스의 헤더파일을 열고 2번에서 설정한 변수의 Type을 CHotButton으로 바꾼다.
Ex) CButton m_FlatBtn CHotButton m_FlatBtn;
버튼에 아이콘을 넣고  싶으면 OnInitDialog 함수에서 다음과 같이 설정한다.
Ex) m_FlatBtn.SetIcon(IDR_MAINFRAME);
void CHotButton::SetIcon(UINT nID, int nWidth, int nHeight)
nID : 삽입할 아이콘의 리소스 아이디.
nWidth,nHeight : 아이콘의 크기
※ 주의할 점은 버튼의  동작을 맵핑해야 완전한 동작을 볼 수 있다.

## 2.14. 컬러 선택 콤보박스

이 클래스는 MS-OFFICE에서 채용된 확장형 컬러 선택 다이얼로그이다.
깔끔한 모습이고 마우스가 위로가면 토글 되는 형태이다. 또한 특이한 점은 콤보박스 컨트롤을 이용한 것이 아니고 버튼 컨트롤을 이용해서  생성된다.
클래스는 CColourPopup과 CColourPicker 두개이고 파일은 ColourPicker.h, ColourPicker.cpp, ColourPopup.h, ColourPopup.h이다.
<사용법>
리소스 뷰에서 다이얼로그에 버튼 컨트롤을 올려놓는다. 그리고 속성에서 styles탭에 Owner Draw속성을 체크한다.
클래스 위저드를 열고 버튼의 컨트롤 변수를 만든다. 그리고 클래스 위저드를 닫고 버튼 컨트롤 변수의 객체를 변경한다.

Ex) Cbutton m_ColourBox; CColourPicker  m_ColourBox;
DoDataExchange함수에 다음과  같이 코딩한다.
CDialog::DoDataExchange(pDX);
//{{AFX_DATA_MAP(CColourPickDemoDlg)
DDX_Control(pDX, IDC_COLOURPICKER,  m_ColourBox);
//}}AFX_DATA_MAP
DDX_ColourPicker(pDX, IDC_COLOURPICKER,  m_crColour);
생성자에 위에 코딩된 메시지 변수를 초기화한다. 콤보 박스에 처음 나타나는 색깔이다.
m_crColour = RGB(0,0,255);
메시지 맵에 다음과 같이 코딩한다.
BEGIN_MESSAGE_MAP(CColourPickDemoDlg, CDialog)
//{{AFX_MSG_MAP(CColourPickDemoDlg)
ON_BN_CLICKED(IDC_COLOURPICKER,  OnColourPickerClicked)
//}}AFX_MSG_MAP
ON_MESSAGE(CPN_SELENDOK, OnSelEndOK)
ON_MESSAGE(CPN_SELENDCANCEL, OnSelEndCancel)
ON_MESSAGE(CPN_SELCHANGE, OnSelChange)
ON_MESSAGE(CPN_CLOSEUP, OnCloseUp)
ON_MESSAGE(CPN_DROPDOWN, OnDropDown)
END_MESSAGE_MAP()
다음과 같이 메시지 handler를 코딩한다.
LONG CColourPickDemoDlg::OnSelEndOK(UINT  lParam, LONG wParam)
{
TRACE("Selection ended OK₩n");
return TRUE;
}
LONG CColourPickDemoDlg::OnSelEndCancel(UINT  lParam, LONG wParam)
{
TRACE("Selection cancelled₩n");
return TRUE;
}
LONG CColourPickDemoDlg::OnSelChange(UINT  lParam, LONG wParam)
{
TRACE("Selection changed₩n");
return TRUE;
}

```
LONG CColourPickDemoDlg::OnCloseUp(UINT lParam,  LONG wParam)
{
TRACE("Colour picker close up₩n");
return TRUE;
}
LONG CColourPickDemoDlg::OnDropDown(UINT  lParam, LONG wParam)
{
TRACE("Colour picker drop down₩n");
return TRUE;
}
```

여기서 콤보박스에서 선택한 색에 대한 정보는 변수 m_crColour에 담겨져 있다. 그러므로 이 변수를 이용해서 데이터를 받거나 전달할 수 있다.

## 2.15. 이미지 삽입된 메뉴

여기서는 메뉴 바에 이미지가 들어가는 클래스로서, 쿨 메뉴라는 클래스이다. 메뉴에 이미지가 삽입되고 MS-OFFICE 제품군들과 같은 메뉴 바를 간단히 생성할 수 있다.

여기서 사용되는 클래스는CCoolMenuManager와 CSubClass이고 파일은 CoolMenu.h, CoolMenu.cpp와 SubClass.h, SubClass.cpp이다.

<사용법>

메인 프레임 헤더 파일에 다음과 같은 멤버 변수를 지정한다.

```
CCoolMenuManager m_MainMenu;
```

메인 프레임 구현 파일, OnCreate 함수 내에 다음과 같이 코딩하면 간단히 이미지가 메뉴 바에 삽입된다.

```
m_MainMenu.Install(this);
m_MainMenu.LoadToolbar(IDR_MAINFRAME);
m_MainMenu.m_bAutoAccel = FALSE;
```

## 2.16. Static 컨트롤

MFC를 통해서 static 컨트롤에 텍스트가 자동으로 스크롤 되는 클래스다.

이 클래스는 CCreditStatic클래스이고 CreditStatic.h와 CreditStatic.cpp파일로 구성되어 있다.

<사용법>

1. 먼저 다이얼로그나 폼 뷰에 Static컨트롤을 올려놓는다. 스크롤 될 텍스트가 들어갈 정도의 충분한 크기가 되게 한다.
2. 클래스 위저드를 열고 Static 컨트롤에 대한 객체 변수를 생성한다. 변수 이름을 m_TextScroll이라 정한 후 위저드를 닫고 생성된 다이얼로그 클래스 헤더 파일에서 다음과 같이 변수의 형을 바꾼다. CStatic m_TextScroll; -> CCreditStatic m_TextScroll;

하이퍼링크 Static 컨트롤

이 클래스는 CStatic 컨트롤에 HyperLink 개체를 연결해 주는 클래스다. 간단히 자신의 프로그램에서 웹 문서처럼 글씨를 마우스로 클릭하면 HyperLink 문서나 메일을 연결해 줄 수 있다.

이 클래스는 HyperLink.h와 HyperLink.cpp로 구성되어있고, 클래스 이름은 CHyperLink다.

<사용법>

1. 다이얼로그 리소스에 Static 컨트롤을 올려놓고 적당한 이름을 쓰고, 컨트롤의 아이디를 지정한다. 아이디를 지정하지 않으면 클래스 위저드로 객체 변수를 만들 수 없다.
2. 클래스 위저드를 열고 Member Variables 탭을 클릭한 후 Static 컨트롤에 대한 객체 변수를 만든다.

name : m_Url , Category : Control, Type : CStatic

3. 클래스 위저드로 생성된 객체변수를 다이얼로그 클래스 헤더에서 다음과 같이 고쳐준다.

CStatic m_Url; -> CHyperLink m_Url;

4. 다이얼로그 클래스의 구현 파일에서 OnInitDialog 함수 내에 다음과 같이 코딩하면 완성.

```
m_Url.SetURL("mailto:jungyc@hanmail.net");
m_Url.SetColours(RGB(0,0,255),RGB(255,0,0));
m_Url.SetUnderline(TRUE);
```

## 2.17. 원형 버튼 만들기

이 클래스는 일반 사각형 버튼을 둥근 원형 버튼으로 만들어주는 클래스이다. 클래스에 의해서 생성된 버튼의 모양은 다음과 같다.

이 클래스는 CMyBtn 클래스이고 이 클래스 내에 CMemDC 클래스가 쓰인다. 사용법은 아래와 같이 간단히 구현할 수 있다.

<사용법>

1. 다이얼로그에 적당한 크기로 버튼 컨트롤을 올려놓는다. 그런 다음 버튼의 아이디를 정하고 버튼 표면에 나타날 글자를 지정한다. 그리고 style 탭에서 Owner draw속성을 지정한다.
2. 다이얼로그 헤더 파일에서 다음과 같이 CMyBtn의 객체 변수를 설정한다.

CMyBtn m_cirBtn;

3. 다이얼로그 구현 파일의 OnInitDialog 함수에서 다음과 같이 코딩해주면 된다.

m_cirBtn.SubclassDlgItem(IDC_OWN,true);

3D 텍스트 버튼

여기서는 3D 형 텍스트로 만들어지는 버튼을 생성해 주는 클래스다.

여기서 사용되는 클래스는CMyTextButton이고 헤더 파일은 MyTextButton.h이고 구현 파일은 MyTextButton이다.

<사용법>

먼저 다이얼로그에 버튼 컨트롤을 올려놓는다. 그리고 아이디와 이름은 임의로 주시고 속성 다이얼로그의 style 탭에서 owner draw 속성을 체크한다. 그런 다음 클래스 위저드를 열고 member variables 탭에서 버튼의 컨트롤  객체를 생성한다.

클래스 위저드를 닫고 다이얼로그의 헤더 파일로 가서 클래스 위저드가 생성한 컨트롤 객체의 type을 다음과 같이 변경한다.

CButton  m_ctlText; -> CMyTextButton m_ctlText;

## 2.18. 확장 Static 컨트롤(Label)

이 클래스는 Static 컨트롤을 비주얼 베이직의 Label 보다도 더 확장된 인터페이스를 제공해 준다. 배경색 바꾸기, 글자색 바꾸기, 폰트의 형태 바꾸기, 하이퍼링크 텍스트 만들기 등 여러 가지 유용한 기능을 제공해 준다. 이것도 www.codeguru.com 에서 제공된 클래스이다. 이 클래스는 CLabel 클래스이고 Label.h와 Label.cpp로  이루어져 있다.

<사용법>

1. 다이얼로그에 Static 컨트롤을 올려놓는다. 주의할 점은 ID 값을 변경해야 클래스 위저드에서 인식을 할 수 있다는 점이다.
2. 클래스 위저드를 열고 Member Variables 탭에서 각 컨트롤의 객체를 생성한다.
3. 다이얼로그 헤더 파일에서 클래스 위저드에서 생성된 컨트롤 객체의 Type을 CLabel 로 변경한다. 그러면 레이블을 만들 수 있는 객체가 생성된다.

<주요 함수들>

SetBkColor(COLORREF crBkgnd): 레이블의 배경색을 지정한다.

SetText(const CString& strText): 레이블에 들어갈 글자를 지정한다.

SetTextColor(COLORREF crText): 글자 색을 지정한다.

SetFontBold(BOOL bBold): 굵은 글씨체인지를 지정한다.

SetFontName(const CString& strFont): 폰트를 지정합니다.

SetFontUnderline(BOOL bSet): 밑줄이 들어간 글자인지를 지정한다.

SetFontItalic(BOOL bSet): 이탤릭체인지를 지정한다.

SetFontSize(int nSize): 글자 크기를 지정한다.

SetSunken(BOOL bSet): 레이블의 모양이 입체적으로 들어간 모양인지 지정한다.

SetBorder(BOOL bSet): 레이블의 테두리가 있는지를 지정한다.
FlashText(BOOL bSet): 글자가 반짝거리는지를 지정한다.
FlashBackground(BOOL bSet): 레이블의 배경이 반짝거리는지를 지정한다.
SetLink(BOOL bLink): 글자가 하이퍼링크 글자인지를 지정(글자는 반드시 도메인 네임이어야 한다.)
SetLinkCursor(HCURSOR hCursor): 하이퍼링크의 커서를 지정한다.

# 3. 구축과정

## 3.1. VFW lib 사용방법

VFW, Vidio For Window는 Microsoft사에서 기본적으로 제공하는 영상녹화 라이브러리이다.
영상녹화에는 DirectX의 Direct Show나 MPEG를 사용하는 방법도 있다.
우선 vfw32.dll에서 기본으로 제공하고 있는 함수 중 중요한 함수 2가지는 캡처 윈도우를 생성해주는 capCreateCaptureWindow()와 cpature Driver의 버전 정보를 가져다주는 capGetDriveDescription()이다.

**capGetDriverDescription()**
**capCreateCaptureWindow()**
**capDriverConnect()**
**capPreviewRate()**
**capSetVideoFormat()**
**capDriverDisconnect()**
**capSetCallbackOnFrame()**

- capGetDriverDescription()함수

  BOOLcapGetDriverDescription(index, name, name_size, version, version_size);
  이 함수는 캡처 드라이버의 이름 및 버전 정보를 검색한다. 첫 번째 매개변수인 index는 검색하고자 하는 드라이버의 번호를 나타내는데, 0부터 9까지의 값을 가질 수 있다.
  즉, 한 컴퓨터에서 9대의 캡처 장치가 사용될 수 있다고 가정하고 있다. 검색하고자 하는 번호의 드라이버가 존재하면 이 함수는 name에 드라이버의 이름을 저장하고 version에 드라이버 버전을 저장한 다음에 함수 결과 값으로 TRUE 값을 반환한다.

- capCreateCaptureWindow()함수

  capCreateCaptureWindow(name, style, x, y, width, height, hWnd, id);
  이 함수는 캡처 윈도우를 생성한다. name 에는 윈도우위 이름을 지정한다. style 윈도우위 스타일을 지정한다. (x, y)에는 캡처 윈도우의 좌측 상단의 좌표를 지정한다. width height에는 캡처 윈도우의 크기를 지정한다. hWnd에는 부모윈도우의 핸들 값을 입력한다. id에는 윈도우의 식별 번호를 입력한다. 캡처 윈도우가 정상적으로 생성되면 캡처 윈도우의 핸들의 함수 결과 값으로 반환되고 그렇지 않으면 NULL값이 반환된다.

- capoDriverConnect()함수

  capDriverConnect(hWnd, index);
  이 함수는 캡처 윈도우를 캡처 드라이버에 연결한다. hWnd는 캡처 윈도우의 핸들을 나타내고 index는 캡처 드라이버의 번호를 나타낸다. 이 함수는 캡처 장치가 정상적으로 작동하여 연결이 성공되면 TRUE 값을 반환하고 그렇지 않으면 FALSE 값을 반환한다.

- capPreviewRate()함수

  capPreviewRate(hWnd, rate);

  이 함수는 미리보기 (preview)모드에서의 프레임 재생 속도를 설정한다. 여기에서 미리보기란 카메라에서 입력도니 비디오를 파일에 저장하는 것이 아니라 화면에 보여준다는 것을 의미한다. hWnd는 캡처 윈도우의 핸들 값으로 설정하고 rate는 밀리 초(ms) 단 위의 시간으로 설정한다. 예를 들어, rate 값을 66으로 설정하면 0.066초마다 새로운 비디오 프레임을 캡처해서 디스플레이 하게 된다. 이와 같은 속도로 재생을 하면 1초에 15개의 비디오 프레임이 디스 플레이된다.

- carSetVideFormat()함수

  carSetVideoFormat(hWnd, videoFormat, videoFormat_size);

  이 함수는 캡처된 비디오 데이터 형식을 설정한다. 사용자가 원하는 비디오 데이터형식이 캡처 장치에서 지원이 되면 이 함수는 TRUE 값을 반환하고 그렇지 않으면 FALSE 값을 반환하므로 반드시 이 함수의 결과 값이 TRUE 인지 검사한 다음에 다음단계로 넘어가야 한다.

- capDriverDisconnect()함수;

  capDriverDisconnect(hWnd);

  이 함수는 carDriverConnect() 함수에 의하여 연결한 캡처 윈도우와 캡처 장치를 분리하는 함수이다. hWnd에는 분리하고자 하는 캡처 윈도우의 핸들 값을 설정한다.

- capSetCallbackOnFrame()함수

  BOOL capSetCallbackOnFrame(hWnd, func);

  VFW 라이브러리에서는 캡처된 비디오 프레임을 화면에 보여주는 작업을 callback함수를 사용해서 처리하도록 하고 있다. capSetCallbackOnFrame() 함수는 캡처 장치로부터 비디오 프레임이 캡처되었을 때에 이를 화면에 보여주기 위해서 호출되는 callback 함수를 설정한다. hWnd는 캡처 윈도우의 핸들 값으로 설정하고 func는 호출될 함수 이름으로 설정한다.

**이미지 및 동영상 저장**

VFW방법을 사용

```
capFileSaveDIB(m_hWndCap,(LPCTSTR)str);
//경로명과 스크린윈도우만 넣어주면 됨
```

파일구조를 이용한 방법

```
CFile file;
//파일을 만들 클래스
BYTE * pDib
//이미지 버퍼를 가져올 포인터
pDib = new BYTE[g_BmInfo.bmiHeader.biWidth*g_BmInfo.bmiHeader.biHeight*3];
//공백의 이미지버퍼 만들기
```

```
memcpy(pDib, g_Video_MemBuffer,
g_BmInfo.bmiHeader.biWidth*g_BmInfo.bmiHeader.biHeight*3);
//영상 버퍼를 카피
CClientDC dc(this);
//현재 화면 DC가져오기
dc.SetStretchBltMode(COLORONCOLOR);
//컬러이미지를 깨짐 없이 줄이기 위해서
BITMAPFILEHEADER bmFileHeader
//파일 헤드 구조
bmFileHeader.bfSize=sizeof(BITMAPFILEHEADER);
//bmiHeader의 구조체 크기 설정
bmFileHeader.bfType=0x4D42;
//0x4D42;
bmFileHeader.bfOffBits=sizeof(BITMAPFILEHEADER)+sizeof(BITMAPINFOHEADER);
```

**파일구조를 이용한 방법**

```
BITMAPINFO szBitmapInfo
// bitmap 정보
szBitmapInfo.bmiHeader.biSize=sizeof(BITMAPINFOHEADER);
szBitmapInfo.bmiHeader.biWidth=g_BmInfo.bmiHeader.biWidth
szBitmapInfo.bmiHeader.biHeight=g_BmInfo.bmiHeader.biHeight
szBitmapInfo.bmiHeader.biPlanes=1;
szBitmapInfo.bmiHeader.biBitCount=24;
//컬러비트수
szBitmapInfo.bmiHeader.biCompression=BI_RGB;
szBitmapInfo.bmiHeader.biSizeImage=
g_BmInfo.bmiHeader.biWidth*g_BmInfo.bmiHeader.biHeight * 3;
//버퍼 크기
szBitmapInfo.bmiHeader.biXPelsPerMeter=0;
szBitmapInfo.bmiHeader.biYPelsPerMeter=0;
szBitmapInfo.bmiHeader.biClrUsed=0;
szBitmapInfo.bmiHeader.biClrImportant=0;
file.Write(&bmFileHeader, sizeof(BITMAPFILEHEADER));
//파일 정보를 넣기
file.Write(&szBitmapInfo, sizeof(BITMAPINFOHEADER));
//이미지 정보 넣기
file.Write(pDib, szBitmapInfo.bmiHeader.biSizeImage);
//이미지 관련 버퍼 넣기
file.Close();     //파일 닫기
delete [] pDib  //생성한 메모리 해제
```

**저장 시작**

```
capFileSetCaptureFile(m_hWndCap, (LPCSTR)str);
//녹화될 파일 이름 설정
capFileAlloc( m_hWndCap, (1024L * 1024L * 5));
//저장할 메모리 버퍼 설정
capCaptureSequence(m_hWndCap);
// 녹화시작
```

**저장 종료**

```
capCaptureStop(m_hWndCap);
// 녹화 중지.
```

## 3.2. GUI에 의한 Layout

GUI는 Graphical user interface의 약자로 그림 사용자 환경이라는 뜻이다.
Visual C++에서 GUI를 구현하는 것은 API와 MFC가 있다.
Windows 프로그래밍이라고도 하는데, 윈도우의 거의 모든 프로그램은 GUI이다.
API의 계념을 설명하면 어떤 Massage를 받아서 그 Massage에 대한 처리를 해주는 식으로 하는데 Massage중에는 마우스 클릭, 마우스 Move, 키보드 누르고, 키보드 때는 것과 같은 것들이 있고, Timer도 있다.
GUI는 화면상에 출력하는 부분으로 텍스트 출력, 선 출력, 사각형 출력 등 여러 가지가 있다. 도스 프로그램을 하던 사람이 윈도우 프로그램을 할 경우 이 부분에서 버그를 많이 만들고 또한 이 경우 디버깅도 어렵다. 따라서 GUI용 클래스를 만들어 제공한다. 작성한 클래스는 두 가지로 텍스트 출력용 클래스(CText) 및 그래픽 출력용 클래스(CDrawTools)이다.
MFC에서 GUI 함수는 CDC를 기반으로 화면뿐만 아니라 프린터에 출력한다.
CDC는 OnDraw함수의 인자로 OS가 넘겨준다.

**CText**

사용가능 함수

```
CSize DrawText(CDC *pDC, int x, int y, LPCTSTR strText, COLORREF
Color, LOGFONT *pLogFont = NULL);
CSize DrawText(CDC *pDC, LPRECT pRect, LPCTSTR strText, COLORREF
Color, LOGFONT* pLogFont = NULL, int nFormat = -1);
```

대부분 첫 번째 함수를 사용하고, 가운데 정렬로 여러 텍스트를 출력할 경우 nFormat을 DT_CENTER 또는 DT_CENTER | DT_VCENTER | DT_SINGLELINE로 설정하여 주로 사용한다.

CDrawTools

사용가능 함수

```
//선 그리기 함수
void DrawLine(CDC *pDC,int nSx,int nSy,int nEx,int nEy,COLORREF nColor);
void DrawLine(CDC *pDC,RECT Rect,COLORREF Color);
//사각형 그리기 함수
void DrawRect(CDC *pDC,int nSx,int nSy,int nEx,int nEy,long Color);
void DrawRect(CDC *pDC,RECT Rect,long Color);
//원 그리기 함수
void DrawEllipse(CDC *pDC,int nSx,int nSy,int nEx,int nEy,long Color);
void DrawEllipse(CDC *pDC,RECT Rect,long Color);
//둥근 사각형 그리기 함수
void DrawRoundRect(CDC *pDC,int nSx,int nSy,int nEx,int nEy,long Color);
void DrawRoundRect(CDC *pDC,RECT Rect,long Color);
//다각형 그리기 함수
void DrawPolygon(CDC *pDC, POINT *pPoints, int nCount, long Color1,long Color2);
//삼각형 그리기 함수
void DrawTriangle(CDC *pDC, RECT Rect, int nType, long Color1, long Color2);
//nType = 삼각형 중 뾰족한 부분의 위치를 설정한다. 설정 가능한 nType으로는
TYPE_TOP_POINT(위), TYPE_BOTTOM_POINT(아래), TYPE_LEFT_POINT(왼쪽),
TYPE_RIGHT_POINT(오른쪽)등이 있다.
//버튼 모양 그리기 함수
void DrawPopBox(CDC *pDC, int x1, int y1, int x2, int y2, long color);
void DrawPopBox(CDC *pDC, RECT Rect, long Color);
//들어간 버튼 모양 그리기 함수
void DrawPushBox(CDC *pDC, int x1, int y1, int x2, int y2, long color);
void DrawPushBox(CDC *pDC, RECT Rect, long Color);
```

제공하는 클래스만 사용하여도 대부분 그래픽은 모두 표현할 수 있으나 함수를 추가할 필요가 있을 경우를 대비하여 DrawRect함수를 예로 내부 루틴을 설명한다.

기본적으로 MFC에서 사각형 그리는 함수로 Rectangle를 제공한다. 그러나 이 함수만을 사용하여 그림을 그리면 색상을 마음대로 표현할 수 없다. 따라서 색상, 글자의 폰트 등도 원리는 같다. CDC에 설정하는 루틴을 함수 내에 추가한 것뿐이다. 작업 순서를 보면 다음과 같다.

1. Pen, Brush 등 필요한 Object를 만든다.
2. CDC 변수에 SelectObject를 이용하여 설정한다.
3. 사각형을 그린다.
4. CDC 변수에 설정된 속성을 원래대로 되돌린다.
5. Pen, Brush 등 만든 Object를 삭제한다.

```
void CDrawTools::DrawRect(CDC *pDC, int nSx, int nSy, int nEx, int nEy, long Color)
{
CPen Pen, *pOldPen;
CBrush Brush, *pOldBrush;
1. Pen, Brush 등 필요한 Object를 만든다.
Pen.CreatePen(PS_SOLID,m_LineThick,Color);
Brush.CreateStockObject(NULL_BRUSH);
2. CDC 변수에 SelectObject를 이용하여 설정한다.
pOldBrush = (CBrush*)pDC->SelectObject(&Brush);
pOldPen = (CPen*)pDC->SelectObject(&Pen);
3. 사각형을 그린다.
pDC->Rectangle(nSx,nSy,nEx,nEy);
4. CDC 변수에 설정된 속성을 원래대로 되돌린다.
pDC->SelectObject(pOldBrush);
pDC->SelectObject(pOldPen);
5. Pen, Brush 등 만든 Object를 삭제한다.
Brush.DeleteObject();
Pen.DeleteObject();
}
```

### 3.3. 사운드 추가

```
if(pMain->m_bSilen)      // 소리 발생 체크 되었으면
  {
        if(!bSounding)
        //경보음이 현재 울리고 있지 않다면 경보음 울리고 경보음이
        울리고 있으면 지나가게 하기 위해서
        {
              bSounding = true;
              //소리 발생하면서 경보음이 울리고 있다고 플래그 변경
              PlaySound("경찰.wav",NULL,SND_ASYNC|SND_LOOP);
              //소리발생, SND_ASYNC(소리 발생 중에 다른 일을 할 수
                있는 옵션), SND_LOOP(반복연주)
        }
        else     //감지 값 이하일 때,
              {
              if(bSounding)      //경보음이 울리면 정지
                    {
```

```
                    PlaySound(NULL,NULL,0);
                    bSounding = FALSE;
                }
            }
            return TRUE;
}
```

## 3.4. 영상처리

RGB방식은 인간이 인식하는 빛의 본질에 가까운 방식이다. 사람은 빨강(R), 녹색(G), 파랑(B)색 파장에 민감한데 이 3가지 색을 조합하면 모든 가시광선의 색을 나타낼 수 있기 때문이다. 수많은 표시장치(CRT, PDP, LCD) 모두 근본적으로는 RGB에 기반을 두고 있다.

YUV 방식은 밝기(Luminance)인 Y와 색상(Chrominance)인 UV로 나뉜다. 그리고 YUV는 TV에서 사용되는 YCbCr방식으로 바로 변환이 된다. 이렇게 나누는 이유는 과거 흑백 TV와의 호환성 때문이다. 흑백TV는 Y성분만 있어서 그것만 전송하고, 컬러 TV는 UV 성분을 전송하고 합성해서 RGB로 표현하기 때문이다.

그리고 HSI방식은 색상(Hue), 채도(Saturation), 명도(Intensity)로 나뉜다. 포토샵을 비롯한 대부분의 영상처리 엔진들은 HSI방식에 기반을 두고 있다. 이 방식의 경우 영상의 밝기변화에 대해 강한 내성이 있어서 설계한 엔진이 주변 빛의 밝기에 민감하지 않고도 처리를 할 수 있도록 해준다.

### 3.4.1. 기본영상비교

```
//기준 영상과 오차밝기가 60이상 차이나면. RGB각각 비교
// 일반적인 경우에 사용하는 로직이다.
double dbRate = 0.0;
//검사 옵션
for (int i=0; i<g_BmInfo.bmiHeader.biWidth*g_BmInfo.bmiHeader.biHeight i++)
//화면 모든 픽셀만큼 검사
{
long nOffset = 60;
// 차이 오차값
long nIndex = i*3;
// 영상처리 속도를 높이기 위해서 미리 인덱스 한번만 계산하도록 변경
//설정 시 지정된 이미지 와 현재 이미지의 R G B의 밝기 값을 비교 하여 차이
  가 60 이상이면 에러 픽셀로 간주
BOOL bR = (g_Video_TeachBuffer[nIndex] - nOffset) < g_Video_MemBuffer[nIndex] &&
(g_Video_TeachBuffer[nIndex] + nOffset) > g_Video_MemBuffer[nIndex];
nIndex++;
```

```
BOOL bG = (g_Video_TeachBuffer[nIndex] - nOffset) < g_Video_MemBuffer[nIndex] &&
(g_Video_TeachBuffer[nIndex] + nOffset) > g_Video_MemBuffer[nIndex];
nIndex++;
BOOL bB = (g_Video_TeachBuffer[nIndex] - nOffset) < g_Video_MemBuffer[nIndex] &&
(g_Video_TeachBuffer[nIndex] + nOffset) > g_Video_MemBuffer[nIndex];
if(!bR || !bG || !bB)
// 3색상중 하나라도 틀리면
dbRate = dbRate+1.0;
//틀린 픽셀로 간주
}
//오차 값을 1000을 기준으로 해서 값을 표현 RGB오차 값이 난 픽셀을 수의 계산함
dbRate = dbRate / (g_BmInfo.bmiHeader.biWidth*g_BmInfo.bmiHeader.biHeight);
dbRate = dbRate * 1000.0;
```

### 3.4.2. 누적영상비교

기본 영상 처리 로직은 같으나 기본 영상 처리 하기 전 원본이미지를 현 이미지를 적용하여 만든다.

특정한 부분이 서서히 바뀌는 현상이 발생 할 때는 이 로직을 사용하면 침입탐지에 더 효과적이다.

예를 들어 전체적으로 서서히 밝아지는 현상과 같은 것이다.

```
// 누적 영상 알고리즘 기준 영상 + 현재 영상의 차이 값에 0.1을 곱해서 누적영상에 더해준다.
for (int i=0; i<g_BmInfo.bmiHeader.biWidth*g_BmInfo.bmiHeader.biHeight i++)
{
long nIndex = i*3;
g_Video_TeachBuffer[nIndex]= g_Video_TeachBuffer[nIndex] + (BYTE)(0.1
* (g_Video_MemBuffer[nIndex] - g_Video_TeachBuffer[nIndex]) + 0.5);
nIndex++;
g_Video_TeachBuffer[nIndex]= g_Video_TeachBuffer[nIndex] + (BYTE)(0.1
* (g_Video_MemBuffer[nIndex] - g_Video_TeachBuffer[nIndex]) + 0.5);
nIndex++;
g_Video_TeachBuffer[nIndex]= g_Video_TeachBuffer[nIndex] + (BYTE)(0.1
* (g_Video_MemBuffer[nIndex] - g_Video_TeachBuffer[nIndex]) + 0.5);
}
// 기준 영상에 이미지의 현재 영상 이미지를 빼고 그 차이 값을 0.1을 곱한 후 기준영상에 더해준다. 이로써, 현재 영상이 기준 영상과 차이 영상의 10프로가 반영되어진다.
```

### 3.4.3. 이전영상비교

기본 영상 처리 로직은 같으나 영상 처리를 하고 나서, 현재의 영상을 기준영상으로 복사를 해주어서 전 프레임의 영상을 기준영상으로 한다. 이 로직은 물체가 움직임이 있을 때만 감지를 할 때 사용한다.

```
memcpy(g_Video_TeachBuffer,g_Video_MemBuffer,g_BmInfo.bmiHeader.biWidth
*g_BmInfo.bmiHeader.biHeight*3);
```

## 3.5. E-mail

SMTP프로토콜이란?

SMTP [simple mail transfer protocol] :인터넷에서 전자우편(E-mail)을 보낼 때 이용하게 되는 표준 통신 규약을 말한다.

컴퓨터로 E-Mail을 보낼 때, sendmail이나 qmail 등의 메일서버프로그램들이 사용된다. 이 프로그램은 사용자 또는 Outlook Express와 같은 메일클라이언트와 미리 정해진 규칙들을 사용하여 메일을 발송하게 된다. 정해진 규칙들을 SMTP 프로토콜이라고 한다.

예를 들면, 윈도우 Outlook Express를 사용하여 메일을 보내기 위해서는 계정설정을 미리 해두어야 하고, 그 계정정보를 보면 "보내는 메일 서버(SMTP 서버)"라는 곳이 있다. 아웃룩 익스프레스와 이 SMTP서버가 통신을 하는 데에 쓰이는 프로토콜(통신규약)이 바로 SMTP프로토콜이다.

SMTP프로토콜은 텍스트기반의 프로토콜이기 때문에, 일반 Telnet 프로그램으로 사용이 가능하다. 즉, 텔넷으로 메일서버에 연결해서 SMTP프로토콜에 맞도록 명령만 내려주면 바로 메일이 발송된다.

이런 작업들을 하기 위해서는 당연히 자신이 사용하는 메일서버가 있어야 한다. 이미 Outlook Express 등의 메일클라이언트를 사용하여 메일을 주고받을 수 있다면, 그곳에 있는 메일계정에 있는 SMTP서버의 주소를 메모해두면 된다. 그런 메일서버가 있는지 없는지를 모른다면, SMTP, POP3서비스를 무료로 제공하는 서비스에 가입을 하면 될 것이다. 그런 곳에 가입을 하면 Outlook Express 설정하는 법을 알려주는데, 이때 SMTP서버의 주소를 메모해두면 된다. 간단히 엠파스나 네이버 등에서 POP3라고 검색하면 그런 서비스업체들을 쉽게 찾을 수 있다.

다음이나 네이버에서 SMTP 메일을 발송하게 되면 이상한 곳에서 오는 메일로 간주하여 SMTP를 거부한다. 그리고 기본적으로 다음이나 네이버 같은 곳에서 SMTP서버를 지원하지 않고 있다. 그래서 SMTP로 보낼 수 없는 것이다. SMTP도 TCP/IP로 통신하고 있다. 시도를 했는데 연결은 되지만 접근을 하지 못하도록 막아놓은 것이다.

네이버 및 다음, 야후 등에서는 SMTP를 지원하지 않기 때문에 중부대학교 메일서버를 운용하게 되었다.

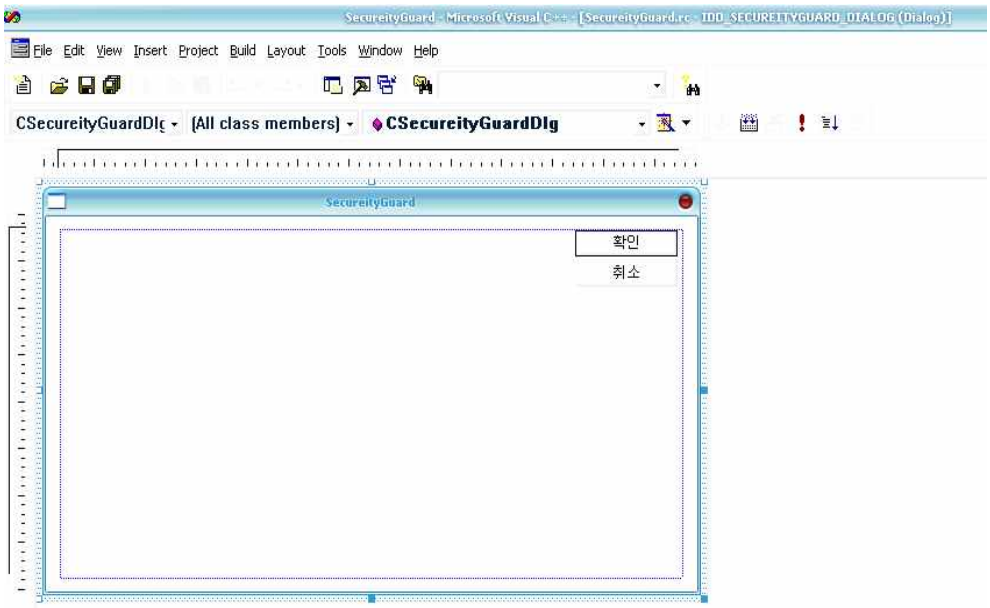
```
UINT CSecureityGuardDlg::ThreadSendToMail(LPVOID lParam)
{
//메일 보내는 루틴
CSecureityGuardDlg *pMF = (CSecureityGuardDlg *)lParam;
pMF->SetTimer(3,500,NULL);
//글자 깜빡이게 타이머 작동
CSMTPConnection smtp;
//smtp를 보내기 위한 연결자
//Connect to the server
if (!smtp.Connect(_T("mail.joongbu.ac.kr")))
//학교에 메일을 보낼 수 있게 해주는 smtp서버와 연결
{
//실패했을 때는 종료
CString sResponse = smtp.GetLastCommandResponse();
TRACE(_T("Failed to connect to SMTP server₩n"));
pMF->GetDlgItem(IDC_MAIL_STATIC)->ShowWindow(SW_HIDE);
pMF->KillTimer(3);
return 0;
}
CSMTPMessage m;      //메시지를 넣을 변수
CString strTo;   //받는 사람 넣을 변수
strTo = pMF->m_strMail;
strTo += "@joongbu.ac.kr";
m.AddRecipient(CSMTPAddress(strTo)); //받는 사람 명단을 넣음.
m.m_From = CSMTPAddress(" SecureCam@joongbu.ac.kr              ");
//보내는 사람
m.m_sSubject = "침입이 감지되었습니다!!!!!!!"; //메일 제목
m.AddBody("침입이 감지되었으니 확인 부탁 합니다");  //메일 내용
CSMTPAttachment a;
//파일 첨부하는 방법
CString strImgPath = pMF->m_strFilePaht;
if(strImgPath.GetLength() >= 1)
{
 if(_access(strImgPath,0) != -1)      //파일이 존재 유무
 {
          a.Attach(strImgPath);      //파일명이 존재하면, 파일을 첨부
          m.AddAttachment(&a);
```

```
        }
      }
      smtp.SendMessage(m);       //메일 보내기를 한다.
      //if(smtp.SendMessage(m))
      //AfxMessageBox("성공");
      //else
      //AfxMessageBox("실패");
      smtp.Disconnect();       //메일 보낸 후 연결을 끊는다.
      pMF->GetDlgItem(IDC_MAIL_STATIC)->ShowWindow(SW_HIDE);
      pMF->KillTimer(3); //깜빡임 종료시킨다.
      return 1;
}
```

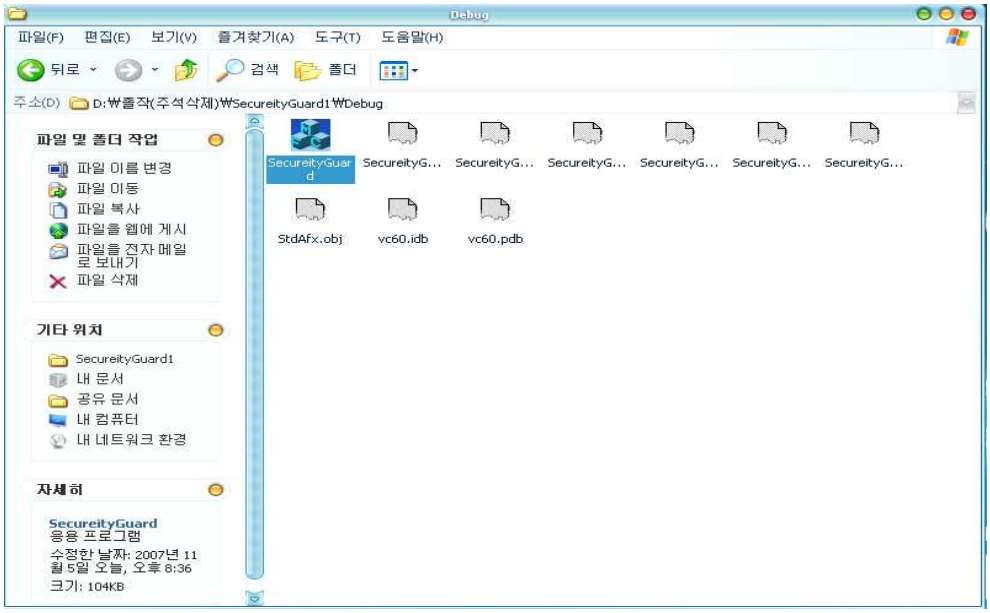
### 3.6. 구축환경

- Windows XP
- visual c++ 6.0
- SAMSUNG MPC-C10 CAMERA

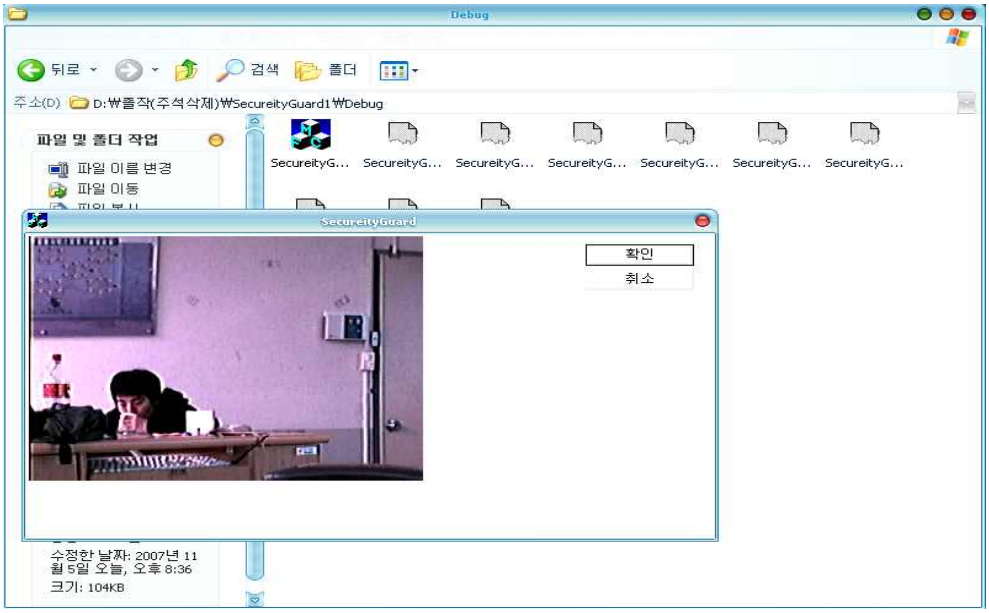
# 4. 운　　영



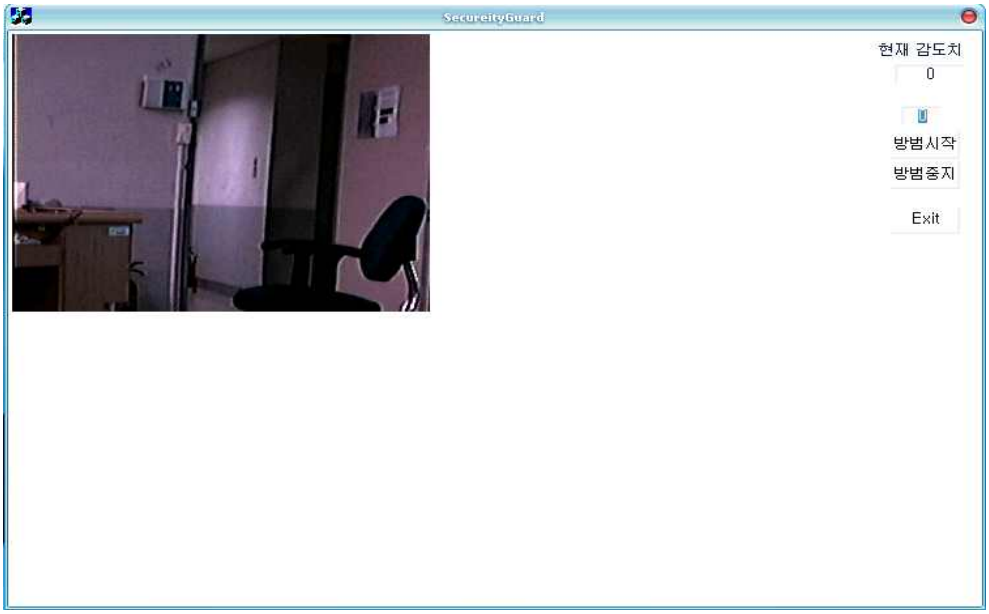

[그림 1.] 컴퓨터와 VFW를 연동시키고 캠을 C++와 연동시켜서 보여주는 GUI 화면



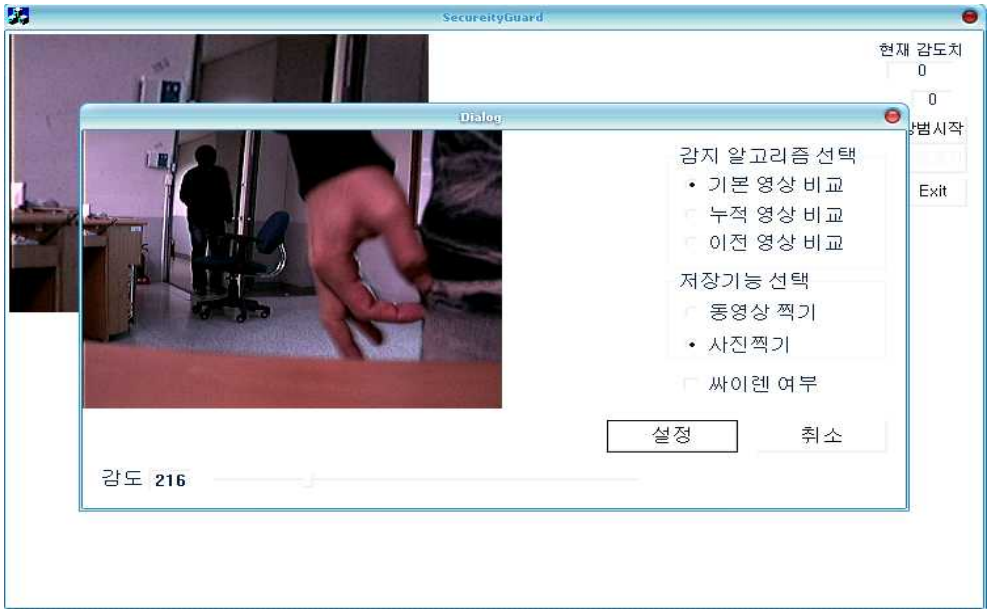

[그림 2.] 위의 화면에 컴파일을 시키는 화면



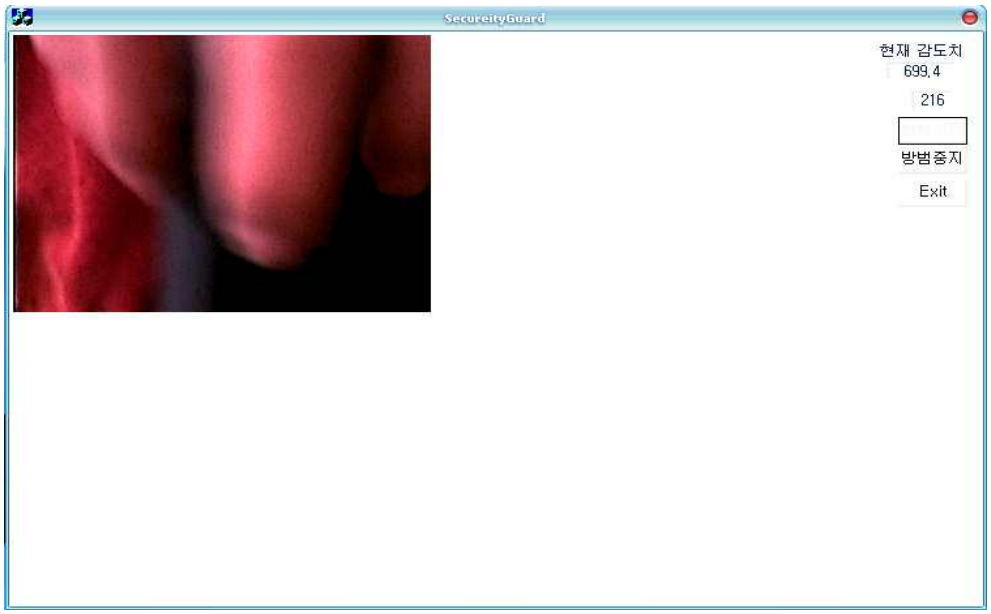

[그림 3.] 컴파일 후 캠에 비추어진 화면



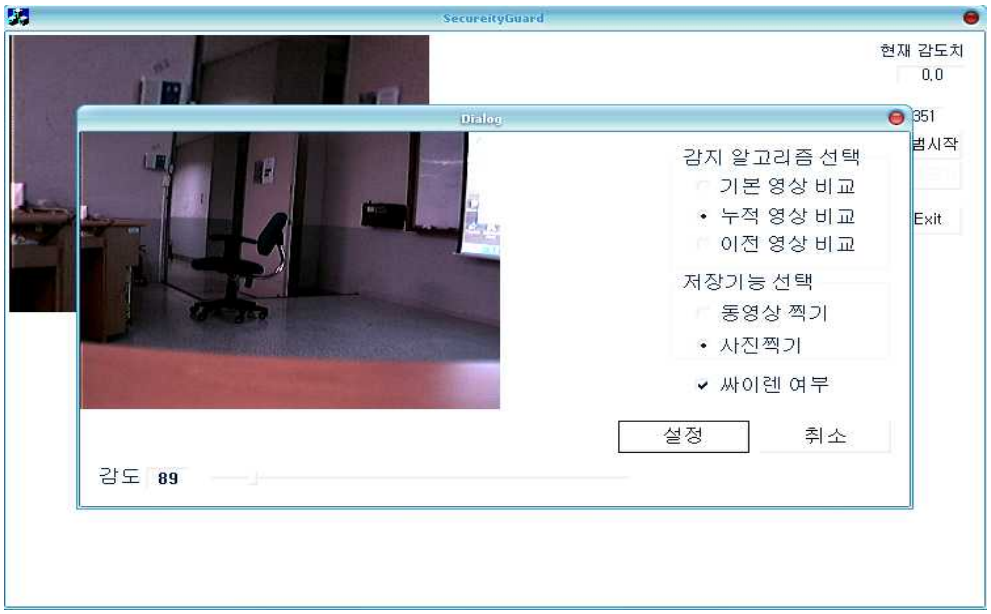

[그림 4.] 기본영상 로직 알고리즘을 이용하여 화면을 저장



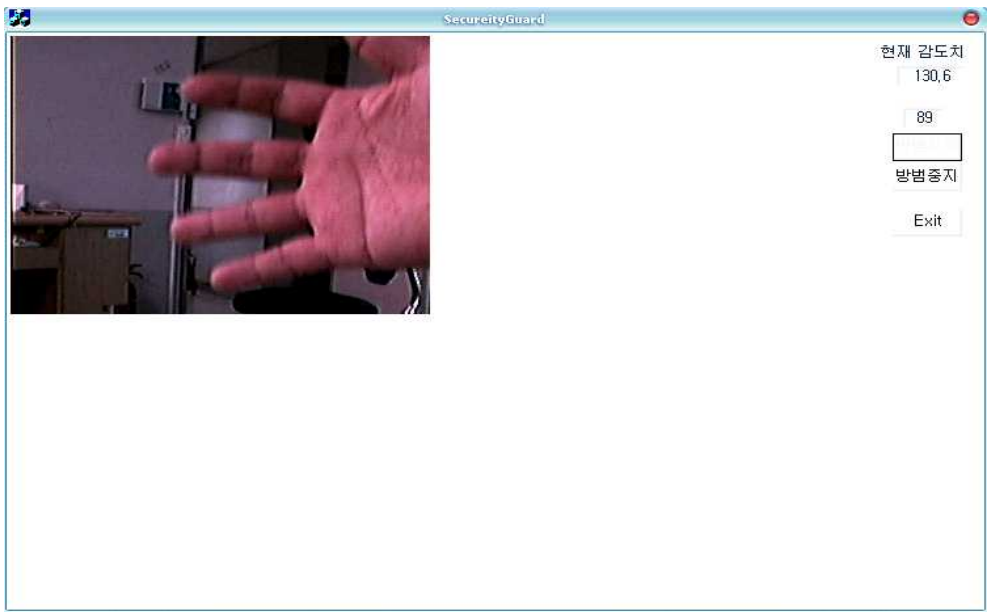

[그림 5.] 기본영상비교 알고리즘을 이용하여 캠에 화면에 찍히는 화면 설정 및 감도, 사진 찍기, 사이렌 여부 설정화면



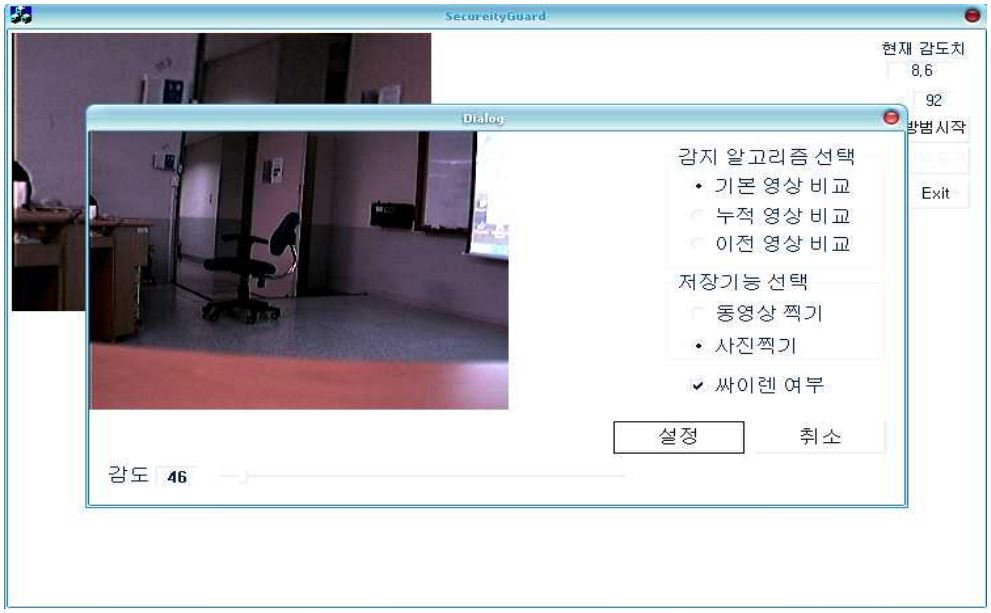

[그림 6.] 감도 설정 후 화면에 찍히는 과정 확인



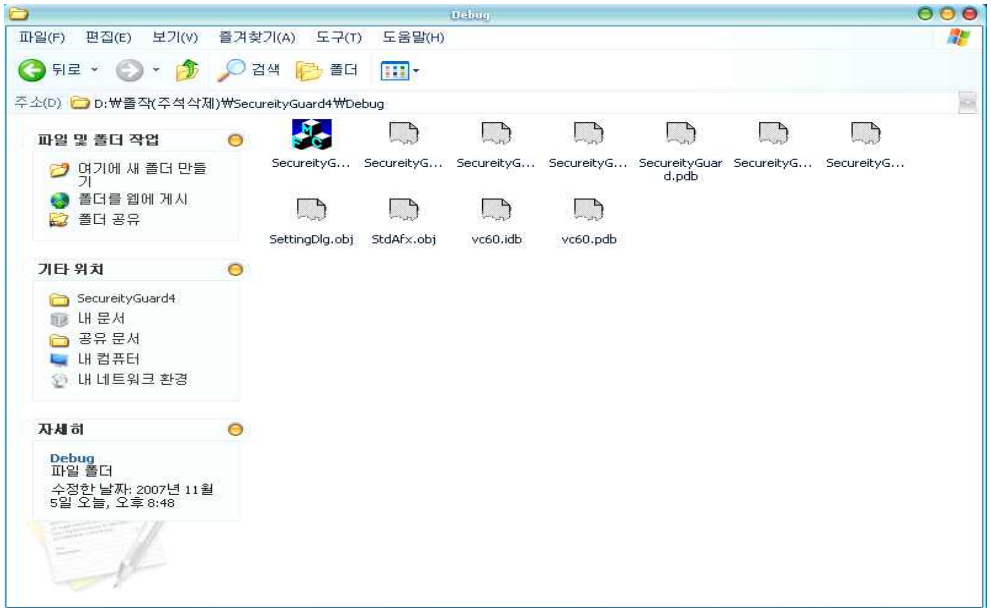

[그림 7.] 누적영상에 의한 영상비교 화면 설정과정



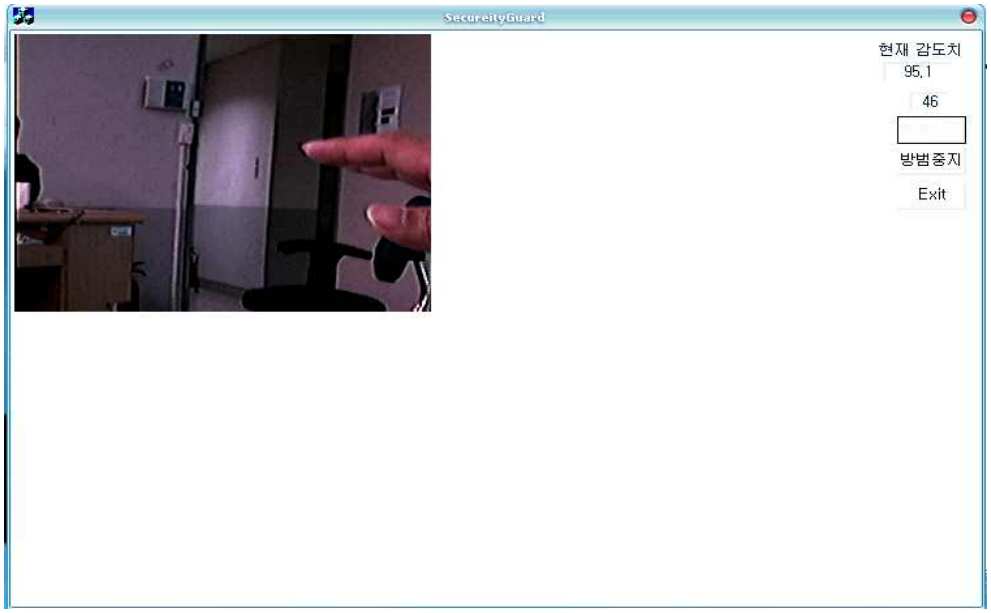

[그림 8.] 누적영상에 의해서 찍힌 감도치 초과 시 화면



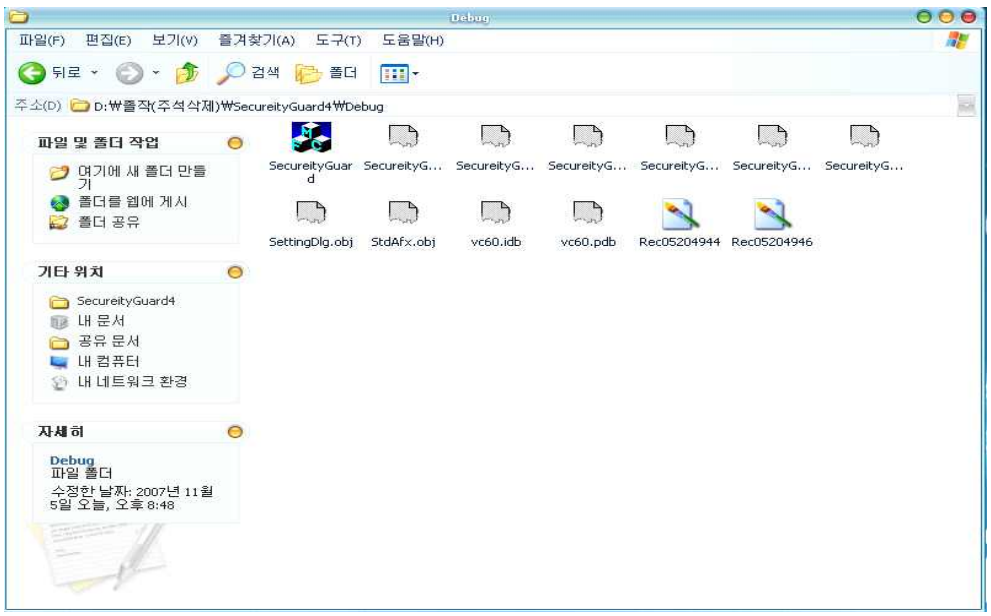

[그림 9.] 기본영상비교에 의해서 사진 찍기를 부여하여 감도설정과 사이렌 설정하기 위한 화면



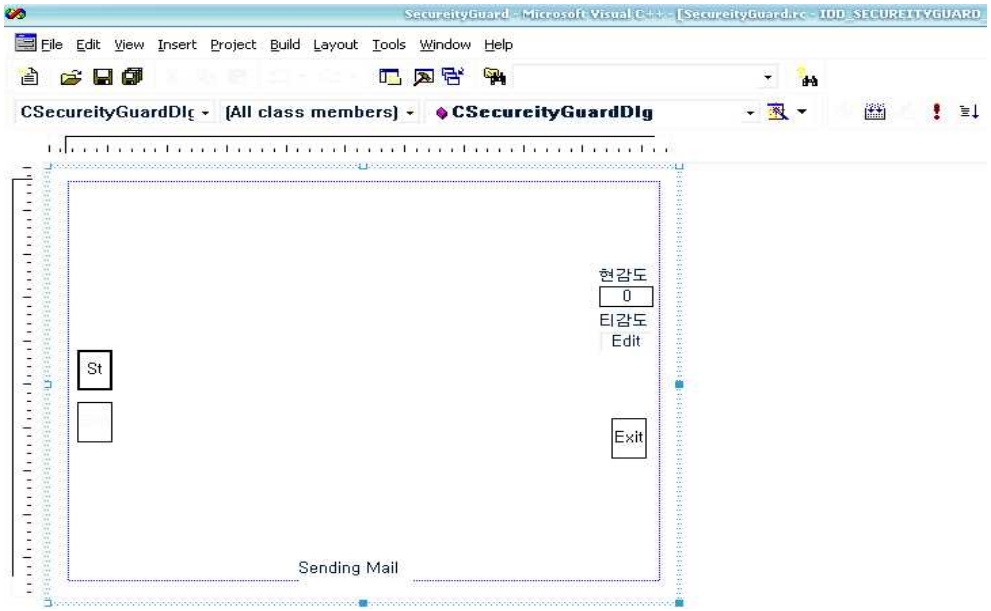

[그림 10.] 위의 화면을 컴파일 시켰을 시에 디버깅된 화면



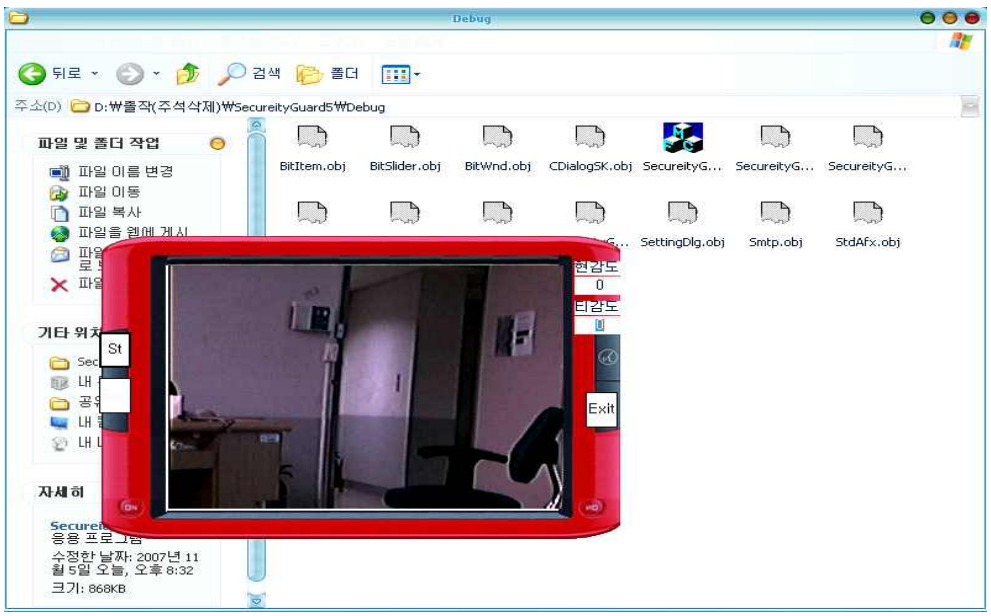

[그림 11.] 감도치를 46으로 설정 후에 감도치 초과 후 찍힌 화면




[그림 12.] 감도치 초과하여 디버그 된 폴더에 저장된 이미지 파일을 보여준 화면



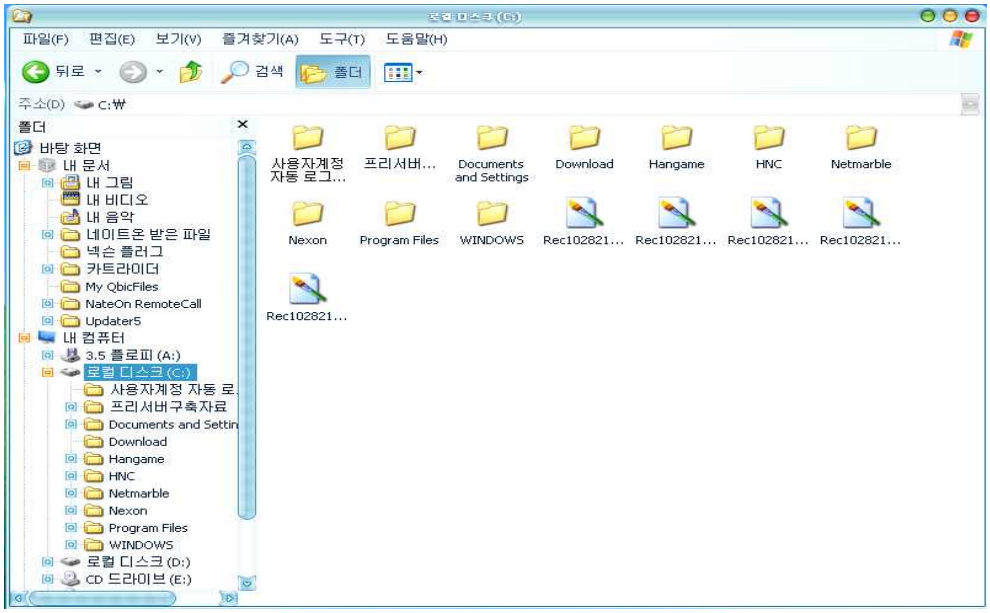

[그림 13.] 기능을 추가시킨 GUI LAYOUT 설정화면



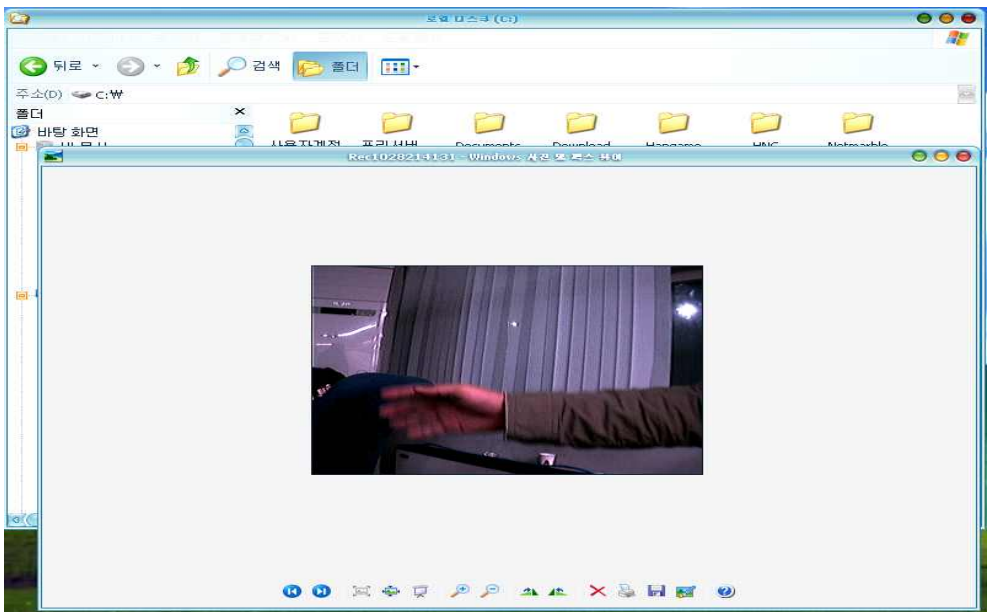

[그림 14.] GUI기능을 추가시킨 컴파일 소스의 디버깅된 결과 화면


[그림 15.] 기본영상비교로 사진 찍기, 사이렌알람, 자동메일서비스 기능을 부가 시킨 화면


[그림 16.] 감도가 59를 초과 했을 때 찍힌 사진을 보여주고 메일을 보내는 화면



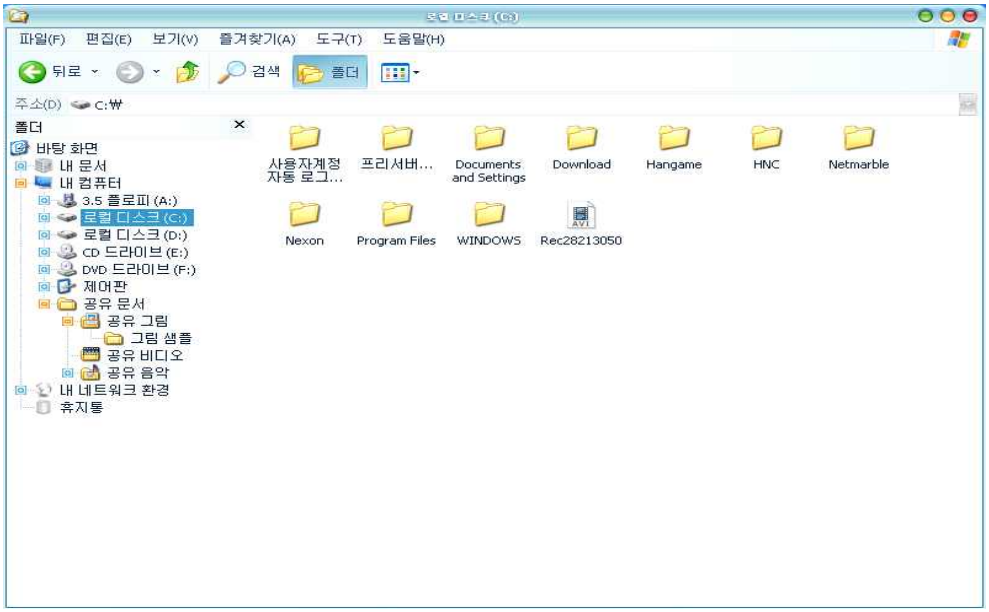

[그림 17.] 컴퓨터에 자동 저장되는 화면(년/월/일 시간 순서로 저장)




[그림 18.] 저장 폴더에 찍힌 파일을 열어본 화면

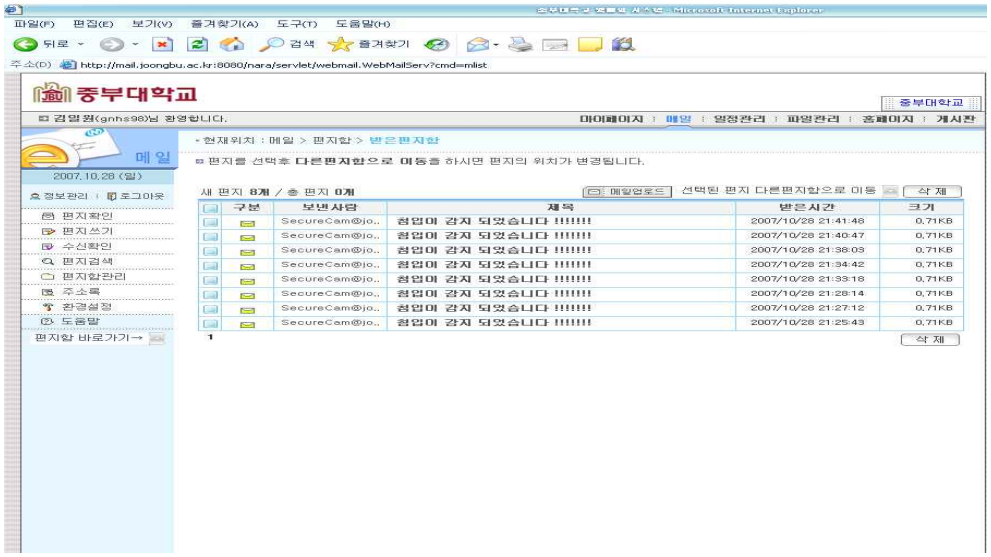

[그림 19.] 동영상 촬영 설정 화면


[그림 20.] 감도가 59를 초과 했을 때 찍힌 동영상을 보여주고 메일을 보내는 화면

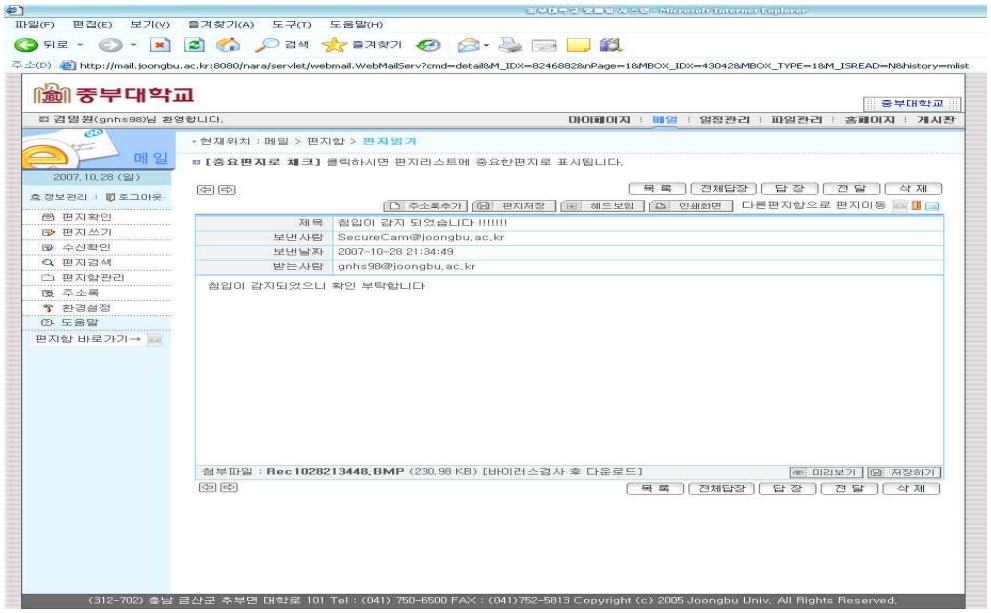



[그림 21.] 컴퓨터에 자동 저장되는 화면(년/월/일 시간 순서로 저장)

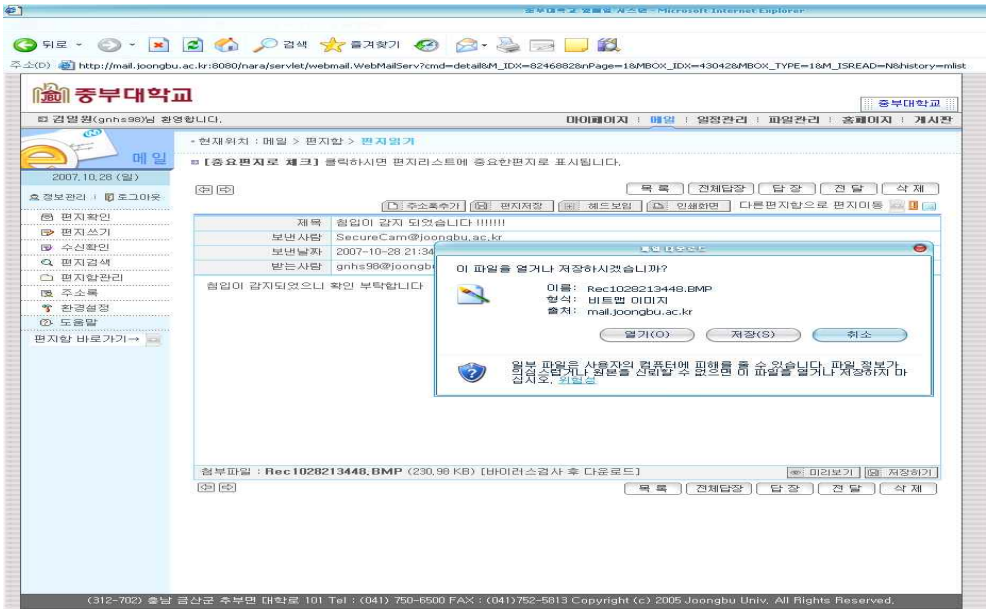

[그림 22.] 저장된 동영상 파일을 재생한 화면



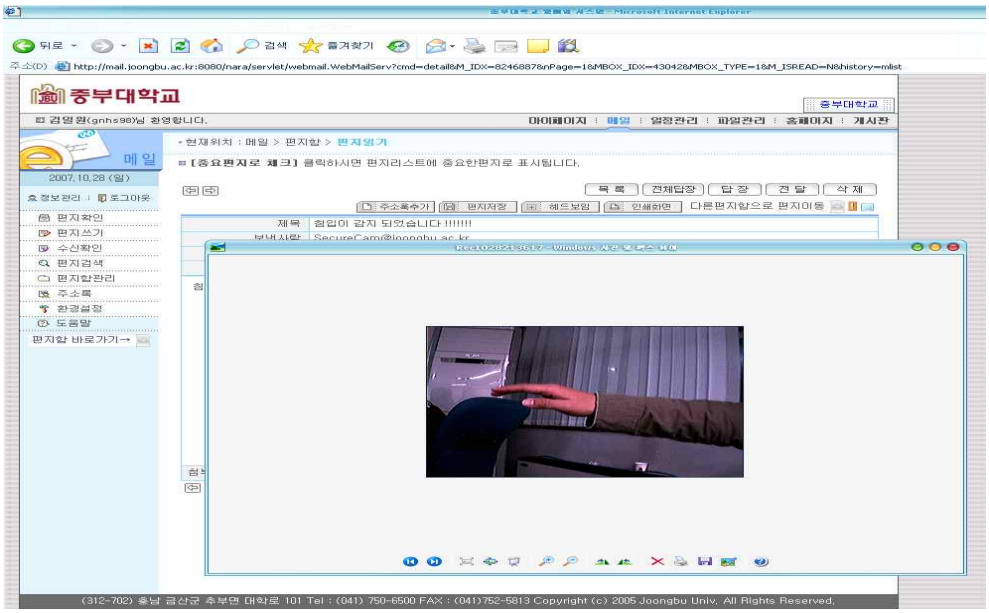

[그림 23.] 감지된 사진을 이메일로 전송된 목록




[그림 24.] 이메일에 첨부된 파일과 메시지 화면




[그림 25.] 저장된 메일을 확인하는 화면




[그림 26.] 메일에 전송된 첨부파일 실행 화면

# 5. 결　　론

## 5.1. 과제의 성과

오늘날 웹캠은 사용자들에게 너무나도 간단한 용도로만 사용되고 있다. 웹캠은 알고 있는 것보다 더 많은 기능과 다양한 용도로 사용될 수 있다. 그래서 저희 조는 웹캠의 장점을 살리면서 보안에 도움을 줄 수 있는 방법을 고민하다 웹캠을 이용한 보안 감시 시스템을 선택하게 되었다.

저희 조가 웹캠을 이용하여 보안에 도움이 되는 점을 살펴보면 다음과 같다.
CCTV의 장점을 살리면서 웹캠이라는 저가의 카메라를 이용하여 카메라상의 화면에 1픽셀의 빛의 오차를 이용해 기존의 영상과 비교분석하고 C++의 MFC 함수를 운용하여 오차 발생 시에 이미지화면이나 동영상을 찍는 기능을 수행한다.
화상 캠으로 감시기능이 있는 제품들도 출시되어 있지만 4년 동안의 대학생활을 하면서 내가 배웠던 프로그램을 이용해서 캠의 감시기능을 직접 프로그래밍 해보는 시간을 가졌다.
VFW를 이용하여 컴퓨터와 캠을 연동시켰고, 전반적인 C++에서 MFC함수를 사용하였으며 사진을 찍고 저장하여 기존영상 비교 하였다. 또한 매 초마다 화면을 자동 저장하여 오차의 범위에 따라서 오차를 초과 하였을 시에 화면을 찍는 방식을 사용하여 변화가 감지되면 알람기능으로 인해 청각적으로도 알 수 있게 해두었다.
또한 사용자가 24시간 컴퓨터 화면을 감시 할 수는 없기 때문에 화면이 찍혔을 시에 자동적으로 감지된 화면을 사용자의 메일에 전송 할 수 있게 했다.

일반적으로 기존의 화상 캠 사용자들은 메신저의 화상채팅이나 사직을 찍는 곳에 캠을 사용하고 있는 것이 현실이다. 점차 새로운 제품들이 출시되고 있으나 일반인들이 쓰기에는 가격이 비싸거나 거의 사용하지 않는 기능들을 가지고 있다.
하지만, 위와 같은 기능 추가로 인하여 사용자는 적은 비용으로 언제, 어디서든 컴퓨터만 있으면 캠에 의한 보안 상황을 확인할 수 있게 되었다.

## 5.2. 향후 과제

이번 프로젝트에서 우리 조는 캠의 기본적인 기능에 부가하여 보안에 도움이 되는 시스템을 도입하였다. 프로젝트를 진행해 나가면서 우리는 앞으로 변화할 캠에 대해서 생각해보았다. 다양한 기능을 추가할 캠은 역시 정보화 시대의 흐름에 발 맞혀 보안 분야가 가미된 화상 통신 시스템으로 거듭날 것이다. 추측하여 본 생각을 정리하면 3가지로 요약할 수 있다.

*동공 인식 신원확인 기능

디지털 장비를 사용함에 앞서 사용자의 신원을 확인하기 위해 캠에서 동공을 인식하여 사용자로 허가된 사람인지를 확인하는 절차이다. 마이너리티 리포트라는 영화에서 미래의 화면을 보일 때 비슷한 장면이 나온 것을 생각해보면 미래의 캠이 꼭 가지게 될 기능이 될 것이다.

*유비쿼터스 안에서의 캠

미래는 곧 유비쿼터스 시대에 돌입하게 된다. 캠 또한 여기에서 예외일 수는 없다. 현재는 캠 보안으로 pc에 저장하거나 자동으로 e-mail에 전송하여 확인하는 방법이 사용되지만, 유비쿼터스 시대에서는 손안에 작은 pc로써 캠에 의한 보안 상태 설정 및  확인 그리고 사고 발생 시 문제 해결까지 편리하게 사용될 것으로 보인다.

*인공지능 시스템

캠에 인공지능 시스템을 추가하여 사용자의 얼굴 식별 및 지문 인식 기능이 가능하도록 하여 보안기능을 향상시킬 수 있다.

어릴 적 상상 속에서만 존재하던 일들이 이제는 점차 현실화 되고 있다. 캠의 미래는 어떤 모습일 가에 대해 토론하는 것은 즐거운 일이었다. 현재 정보보호에 관심이 많은 학생 으로써 일상 깊이 침투해 있는데 캠의 정보보호 활용에 대해 깊이 고민해보고 직접 다루어 보았다는 점에 큰 의미를 가질 수 있었다.

# [참고 문헌]

## 서적

[1] 신재석, microsoft visual c++ bible 6.0, 삼양출판사, 2000

[2] 김진, MFC Internals : based on the MFC 6.x , 세창, 2002

[3] 이종우, Windows CE MFC programming, 사이텍미디어, 2002

[4] 박승규, (완전정복) visual c++ 6.0, 동일, 2002

[5] 이상엽, Visual C++ Programming Bible Ver 6.x, 영진출판사, 1998

[6] 조순복, (Visual C++ 6.0을 위한) C++ 프로그래밍 기법, 신화전산기획, 2001

[7] 황선규, 영상 처리 프로그래밍 by visual c++, 한빛미디어, 2007

## 사이트

[1] 마이크로소프트 : http://www.microsoft.com/korea/msdn/

[2] CDialogSK-A Skinnable Dialog Class : http://www.codeproject.com/dialog/cdialogsk.asp

[3] A cool bitmap slider like Media Palyer's : http://www.codeproject.com/miscctrl/CoolSlider.asp

[4] GUI(Graphic User Interface) : http://jys92.com.ne.kr/VisualC/GUI.htm

[5] GUI elements 그래픽 사용자 인터페이스 구성요소 : http://www.terms.co.kr/GUIelements.htm

[6] 구글 : http://www.google.co.kr/

[7] 네이버 : http://www.naver.com/

[8] 야후 : http://www.kr.yahoo.com/

부록#1.

## SecureityGuardDlg.cpp

```
#include "stdafx.h"
#include "SecureityGuard.h"
#include "SecureityGuardDlg.h"

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

#include "vfw.h"
#include "SettingDlg.h"
#include "Smtp.h"
#include  <io.h>

BITMAPINFO g_BmInfo;

LPBYTE g_Video_MemBuffer;
LPBYTE g_Video_TeachBuffer;

LRESULT CALLBACK capCallbackOnFrame(HWND hWnd, LPVIDEOHDR lpVHdr);

class CAboutDlg : public CDialog
{
public:
        CAboutDlg();

        enum { IDD = IDD_ABOUTBOX };

        protected:
        virtual void DoDataExchange(CDataExchange* pDX);

rotected:

        DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
```

```
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)

END_MESSAGE_MAP()

CSecureityGuardDlg::CSecureityGuardDlg(CWnd* pParent /*=NULL*/)
	: CDialogSK(CSecureityGuardDlg::IDD, pParent)
{

	m_nCnt = 0;

	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
	m_nGrabMode = MODE_NONE;
	m_bFirstMailing =TRUE;
}

void CSecureityGuardDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialogSK::DoDataExchange(pDX);

	DDX_Text(pDX, IDC_EDIT1, m_nCnt);

}

BEGIN_MESSAGE_MAP(CSecureityGuardDlg, CDialogSK)

	ON_WM_SYSCOMMAND()
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_BN_CLICKED(IDC_START_BUTTON, OnStartButton)
	ON_BN_CLICKED(IDC_END_BUTTON, OnEndButton)
	ON_BN_CLICKED(IDC_EXIT_BUTTON, OnExitButton)
	ON_WM_TIMER()

END_MESSAGE_MAP()
```

```
BOOL CSecureityGuardDlg::OnInitDialog()
{
	CDialogSK::OnInitDialog();

	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		CString strAboutMenu;
		strAboutMenu.LoadString(IDS_ABOUTBOX);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING,	IDM_ABOUTBOX,
strAboutMenu);
		}
	}

	SetIcon(m_hIcon, TRUE);
	SetIcon(m_hIcon, FALSE);

	SetBitmap(IDB_BACK_BITMAP);
    SetStyle(LO_RESIZE);
	SetTransparentColor(RGB(0, 0, 255));

	m_bCapturing = FALSE;
	m_bPicturing = FALSE;
	g_Video_MemBuffer = NULL;
	return InitVfwScreen();
}

void CSecureityGuardDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX)
	{
		CAboutDlg dlgAbout;
		dlgAbout.DoModal();
	}
	else
```

```
	{
		CDialogSK::OnSysCommand(nID, lParam);
	}
}

void CSecureityGuardDlg::OnPaint()
{
	if (IsIconic())
	{
		CPaintDC dc(this);
		SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;

		dc.DrawIcon(x, y, m_hIcon);
	}
	else
	{
		CDialogSK::OnPaint();
	}
}

HCURSOR CSecureityGuardDlg::OnQueryDragIcon()
{
	return (HCURSOR) m_hIcon;
}

BOOL CSecureityGuardDlg::InitVfwScreen()
{

	m_hWndCap = capCreateCaptureWindow("Capture Window", WS_CHILD |
			WS_VISIBLE, 75, 90, IMG_WIDTH, IMG_HEIGHT, this->m_hWnd, NULL);


	if(capSetCallbackOnFrame(m_hWndCap, capCallbackOnFrame) ==FALSE) return FALSE;
	if(capDriverConnect(m_hWndCap, 0) == FALSE) return FALSE;
```

```
capGetVideoFormat(m_hWndCap, &g_BmInfo, sizeof(BITMAPINFO));

if (g_BmInfo.bmiHeader.biBitCount != 24)
{
        g_BmInfo.bmiHeader.biBitCount =24;
        g_BmInfo.bmiHeader.biCompression = 0;
        g_BmInfo.bmiHeader.biSizeImage =
                g_BmInfo.bmiHeader.biWidth*g_BmInfo.bmiHeader.biHeight * 3;
        capSetVideoFormat(m_hWndCap,&g_BmInfo, sizeof(BITMAPINFO));
}

CAPTUREPARMS captureParms;
capCaptureGetSetup( m_hWndCap, &captureParms, sizeof(CAPTUREPARMS) );
captureParms.fCaptureAudio = false;
captureParms.fMCIControl = false;
captureParms.fLimitEnabled = false;
captureParms.dwRequestMicroSecPerFrame = 46667;
captureParms.wTimeLimit = 200;
captureParms.fUsingDOSMemory = true;
captureParms.fMakeUserHitOKToCapture = true;
captureParms.wPercentDropForError = 10;
captureParms.wNumVideoRequested = 32;
captureParms.fAbortLeftMouse = false;
captureParms.fAbortRightMouse = false;
captureParms.wChunkGranularity = 0;
captureParms.fYield = true;
captureParms.dwIndexSize = 27000;
captureParms.wNumAudioRequested = 4;
captureParms.vKeyAbort = VK_ESCAPE;
captureParms.fStepMCIDevice = 0;
captureParms.dwMCIStartTime = 10000;
captureParms.dwMCIStopTime = 20000;
captureParms.fStepCaptureAt2x = 0;
captureParms.wStepCaptureAverageFrames = 5;
captureParms.dwAudioBufferSize = 0;
captureParms.fDisableWriteCache = 0;
captureParms.AVStreamMaster = 0;
capCaptureSetSetup( m_hWndCap, &captureParms, sizeof(CAPTUREPARMS) );

capPreviewRate(m_hWndCap, 33);
capOverlay(m_hWndCap, false);
```

```
	capPreview(m_hWndCap, true);
	return TRUE;
}

LRESULT CALLBACK capCallbackOnFrame(HWND hWnd, LPVIDEOHDR lpVHdr)
{


	static bool bSounding = false;
	CSecureityGuardDlg *pMain = (CSecureityGuardDlg*)AfxGetMainWnd();를 가져올
	수 없다. 그래서 AfxGetMAinWnd 메인 클래스 포인터를 가져온다.


	if(g_Video_MemBuffer==NULL) g_Video_MemBuffer= (LPBYTE)new
BYTE[g_BmInfo.bmiHeader.biHeight*g_BmInfo.bmiHeader.biWidth*3];


memcpy(g_Video_MemBuffer,lpVHdr->lpData,g_BmInfo.bmiHeader.biHeight*g_BmInfo.bm
iHeader.biWidth*3);
	if(pMain->m_nGrabMode == MODE_NONE) return (LRESULT) TRUE;

	if(pMain->m_nGrabMode == MODE_ACCUM)
	{

		for (int i=0;
i<g_BmInfo.bmiHeader.biWidth*g_BmInfo.bmiHeader.biHeight; i++)
		{
			long nIndex = i*3;
			g_Video_TeachBuffer[nIndex] =
g_Video_TeachBuffer[nIndex] +
				(BYTE)(0.1 *
(g_Video_MemBuffer[nIndex] - g_Video_TeachBuffer[nIndex]) + 0.5);
			nIndex++;
			g_Video_TeachBuffer[nIndex] =
g_Video_TeachBuffer[nIndex] +
				(BYTE)(0.1 *
(g_Video_MemBuffer[nIndex] - g_Video_TeachBuffer[nIndex]) + 0.5);
			nIndex++;
			g_Video_TeachBuffer[nIndex] =
g_Video_TeachBuffer[nIndex] +
				(BYTE)(0.1 *
```

```
(g_Video_MemBuffer[nIndex] - g_Video_TeachBuffer[nIndex]) + 0.5);
                }
        }


        double dbRate = 0.0;
        for (int i=0; i<g_BmInfo.bmiHeader.biWidth*g_BmInfo.bmiHeader.biHeight; i++)
        {
                long nOffset = 60;
                long nIndex = i*3;
                BOOL bR = (g_Video_TeachBuffer[nIndex] - nOffset) <
g_Video_MemBuffer[nIndex] &&
                                        (g_Video_TeachBuffer[nIndex] + nOffset) >
g_Video_MemBuffer[nIndex];
                nIndex++;
                BOOL bG = (g_Video_TeachBuffer[nIndex] - nOffset) <
g_Video_MemBuffer[nIndex] &&
                                        (g_Video_TeachBuffer[nIndex] + nOffset) >
g_Video_MemBuffer[nIndex];
                nIndex++;
                BOOL bB = (g_Video_TeachBuffer[nIndex] - nOffset) <
g_Video_MemBuffer[nIndex] &&
                                        (g_Video_TeachBuffer[nIndex] + nOffset) >
g_Video_MemBuffer[nIndex];
                if(!bR || !bG || !bB)
                        dbRate = dbRate+1.0;
        }

        dbRate                  =                  dbRate                  /
(g_BmInfo.bmiHeader.biWidth*g_BmInfo.bmiHeader.biHeight);
        dbRate = dbRate * 1000.0;


        if(pMain->m_nGrabMode == MODE_FRAME)
        {

memcpy(g_Video_TeachBuffer,g_Video_MemBuffer,g_BmInfo.bmiHeader.biWidth*g_BmInf
o.bmiHeader.biHeight*3);
        }

        CString strRate;
```

```
		strRate.Format("%.1f",dbRate);
		pMain->SetDlgItemText(IDC_COUNT_STATIC,strRate);

		if(dbRate >= pMain->m_nCnt)
		{

			if(pMain->m_bSilen)
			{
				if(!bSounding)
				{
					bSounding = true;
					PlaySound("경찰.wav",NULL,SND_ASYNC|SND_LOOP);
				}
			}
			if(!pMain->m_bPic)
			{
				pMain->SetCapture();
			}
			else
			{
				pMain->SetPicCap();
			}
		}
		else
		{
			if(bSounding)
			{
				PlaySound(NULL,NULL,0);
				bSounding = FALSE;
			}
		}

		return TRUE;
}

void CSecureityGuardDlg::OnStartButton()
{
		CSettingDlg dlg;
		if(g_Video_TeachBuffer == NULL)
			g_Video_TeachBuffer = new BYTE[g_BmInfo.bmiHeader.biWidth*g_
```

```
BmInfo.bmiHeader.biHeight*3];

memcpy(g_Video_TeachBuffer,g_Video_MemBuffer,g_BmInfo.bmiHeader.biWidth*g_BmInf
o.bmiHeader.biHeight*3);

dlg.SetImage(g_Video_TeachBuffer,g_BmInfo.bmiHeader.biWidth,g_BmInfo.bmiHeader.biHeight);

        if(dlg.DoModal() == IDOK)
        {

                m_bFirstMailing =TRUE;
                m_nGrabMode = dlg.m_nAlg;
                m_nCnt = dlg.m_nSens;
                m_bSilen = dlg.m_sCheck;
                m_bPic = (BOOL)dlg.m_nMovie;
                m_strMail = dlg.m_strMail;
                UpdateData(FALSE);

                GetDlgItem(IDC_START_BUTTON)->EnableWindow(FALSE);
                GetDlgItem(IDC_END_BUTTON)->EnableWindow(TRUE);
        }
}

void CSecureityGuardDlg::OnEndButton()
{
        PlaySound(NULL,NULL,0);
        if(m_bCapturing)
        {
                KillTimer(0);
                capCaptureStop(m_hWndCap);
                SetTimer(2,5000,NULL);
        }

        GetDlgItem(IDC_START_BUTTON)->EnableWindow(TRUE);
        GetDlgItem(IDC_END_BUTTON)->EnableWindow(FALSE);
        m_nGrabMode = MODE_NONE;
}

void CSecureityGuardDlg::OnExitButton()
{
        SendMessage(WM_CLOSE);
```

```
}

void CSecureityGuardDlg::SetPicCap()
{
	if(m_bPicturing) return;
	m_bPicturing = TRUE;
	SaveScreenImg();
	SetTimer(1,3000,NULL);
}

void CSecureityGuardDlg::SetCapture()
{
	if(m_bCapturing) return;
	m_bCapturing = TRUE;

	if(m_bFirstMailing)
	{
		m_bFirstMailing = FALSE;
		m_strFilePaht = "";
		CWinThread *m_pInsThread =
AfxBeginThread(CSecureityGuardDlg::ThreadSendToMail,
		this, THREAD_PRIORITY_NORMAL,
CREATE_SUSPENDED);
		m_pInsThread->ResumeThread();
	}

	CString str;
	m_strFileName = CTime::GetCurrentTime().Format("%d%H%M%S");
	str.Format("c:₩₩Rec%s.avi", m_strFileName);
	capFileSetCaptureFile(m_hWndCap, (LPCSTR)str);
	capFileAlloc( m_hWndCap, (1024L * 1024L * 5));
    capCaptureSequence(m_hWndCap);
	SetTimer(0,20000,NULL);
}

void CSecureityGuardDlg::OnTimer(UINT nIDEvent)
{
	if(nIDEvent == 0)
	{
		KillTimer(0);
		capCaptureStop(m_hWndCap);
```

```
                SetTimer(2,5000,NULL);
        }
        else if(nIDEvent == 1)
        {
                KillTimer(1);
                m_bPicturing = FALSE;
        }
        else if(nIDEvent == 2)
        {
                m_bCapturing = FALSE;
                KillTimer(2);
        }
        else if(nIDEvent == 3)
        {
                static bool bShow = true;
                if(bShow)
                {
                        GetDlgItem(IDC_MAIL_STATIC)->ShowWindow(SW_SHOW);
                        bShow = false;
                }
                else
                {
                        GetDlgItem(IDC_MAIL_STATIC)->ShowWindow(SW_HIDE);
                        bShow = true;
                }
        }
        CDialogSK::OnTimer(nIDEvent);
}

void CSecureityGuardDlg::SaveScreenImg()
{

        CString str;
        m_strFileName = CTime::GetCurrentTime().Format("%m%d%H%M%S");
        str.Format("c:₩₩Rec%s.BMP", m_strFileName);

        capFileSaveDIB(m_hWndCap,(LPCTSTR)str);
        if(m_bFirstMailing)
        {
                m_bFirstMailing = FALSE;
                m_strFilePaht = str;
```

```
                CWinThread *m_pInsThread = AfxBeginThread(CSecureityGuardDlg::
ThreadSendToMail,
                this, THREAD_PRIORITY_NORMAL, CREATE_SUSPENDED);
                m_pInsThread->ResumeThread();
        }

}

UINT CSecureityGuardDlg::ThreadSendToMail(LPVOID lParam)
{

        CSecureityGuardDlg *pMF = (CSecureityGuardDlg *)lParam;
        pMF->SetTimer(3,500,NULL);

        CSMTPConnection smtp;
        if (!smtp.Connect(_T("mail.joongbu.ac.kr")))
        {

                CString sResponse = smtp.GetLastCommandResponse();
                TRACE(_T("Failed to connect to SMTP server₩n"));
                pMF->GetDlgItem(IDC_MAIL_STATIC)->ShowWindow(SW_HIDE);
                pMF->KillTimer(3);
                return 0;
        }
        CSMTPMessage m;
        CString strTo;
        strTo = pMF->m_strMail;
        strTo += "@joongbu.ac.kr";
        m.AddRecipient(CSMTPAddress(strTo));
        m.m_From = CSMTPAddress("SecureCam@joongbu.ac.kr");
        m.m_sSubject = "침입이 감지 되었습니다 !!!!!!!";
        m.AddBody("침입이 감지되었으니 확인 부탁합니다");

        CSMTPAttachment a;
        CString strImgPath = pMF->m_strFilePaht;
        if(strImgPath.GetLength() >= 1)
        {
                if(_access(strImgPath,0) != -1)
                {
                        a.Attach(strImgPath);
                        m.AddAttachment(&a);
```

```
                }
        }

        smtp.SendMessage(m);
        smtp.Disconnect();
        pMF->GetDlgItem(IDC_MAIL_STATIC)->ShowWindow(SW_HIDE);
        pMF->KillTimer(3);
        return 1;
}
```