6장 대화상자와 컨트롤

2020년도 1학기 윈도우 프로그래밍

학습 목표

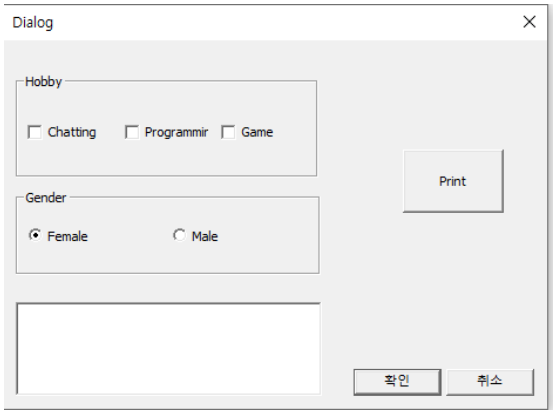
- **학습목표**
  - 대화상자를 만들고 사용할 수 있다.
  - 컨트롤 종류를 알고 각 컨트롤을 사용할 수 있다.
  - 다양한 컨트롤을 이용해 응용 프로그램을 개발할 수 있다.
  - 모덜리스 대화상자를 사용할 수 있다.

- **내용**
  - 대화상자 만들기
  - 컨트롤 종류
  - 버튼 컨트롤
  - 에디트 박스
  - 체크버튼과 라디오버튼
  - 콤보박스
  - 리스트박스
  - 모덜리스 대화상자

1. 대화상자 이용하기

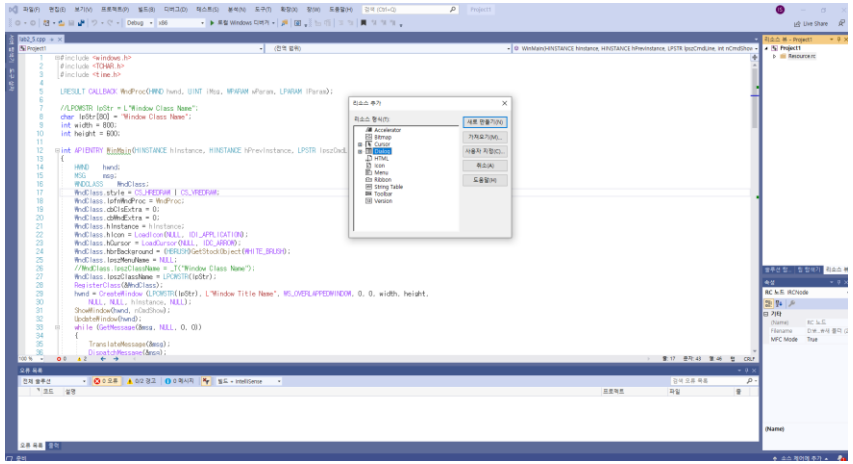
- **대화상자 (Dialog Box)**
  - 프로그램 수행 중 사용자와 간단한 입력/출력을 하기 위해 사용되는 윈도우
  - 많은 양의 정보를 효율적으로 입/출력해주는 매개체, 혹은 말 그대로 사용자와 대화하는 상자
  - 이 대화상자에서 사용하는 도구를 컨트롤이라고 한다.
    - 컨트롤은 사용자로부터 입력을 받거나 사용자에게 정보를 제공하기 위해 사용
    - 대표적 컨트롤: 버튼 컨트롤, 에디트 컨트롤, 콤보박스 컨트롤, 리스트 컨트롤 등

- **사용방법**
  1. 리소스에서 새로운 대화상자 만들기
     - 리소스 형태로 대화상자를 만들고, 대화상자 편집기로 컨트롤들을 디자인한다.
  2. 대화상자 띄우기
     - 대화상자를 메인 윈도우에서 띄운다.
  3. 대화상자에 대한 메시지 처리 함수 **DialogProc()** 작성
     - 별도의 콜백함수를 가지고 대화상자 메시지 처리

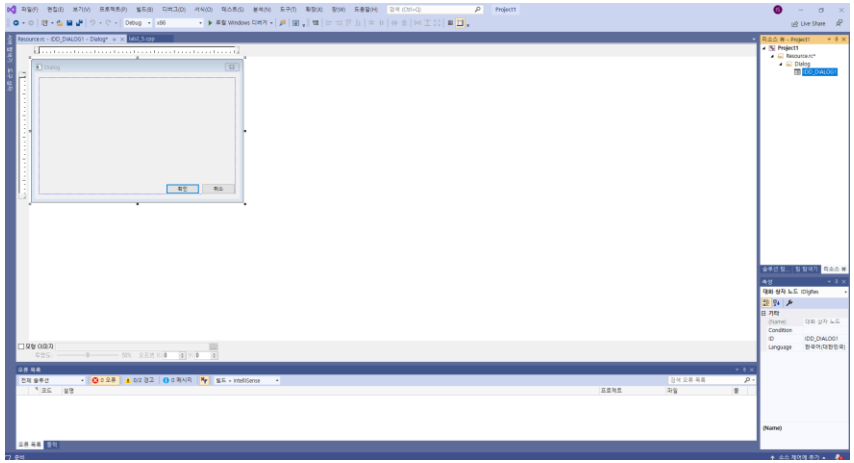


1) 대화상자 만들기

1. 리소스에서 대화상자 만들기 (Visual Studio 2019 환경)

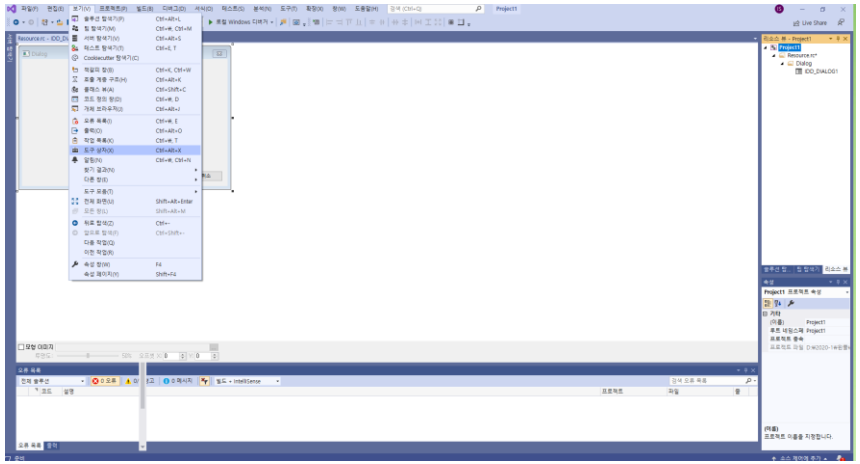


리소스에서 대화상자 추가하기

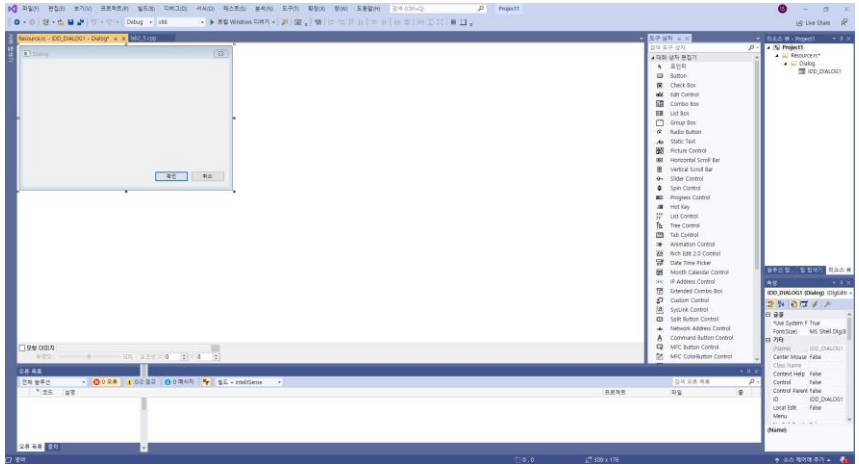


대화상자를 추가하면 리소스에 IDD_DIALOG1이 추가된 것을 확인

대화상자 만들기

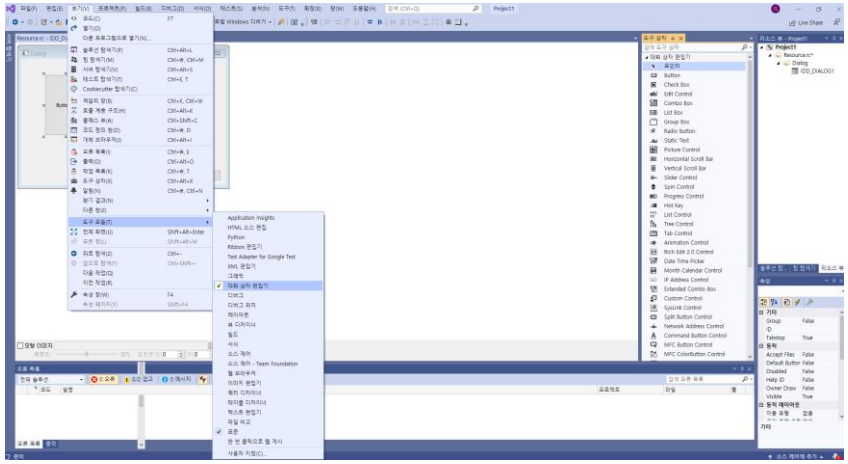


메뉴의 보기 → 도구상자를 열어 사용 가능한 컨트롤들 확인

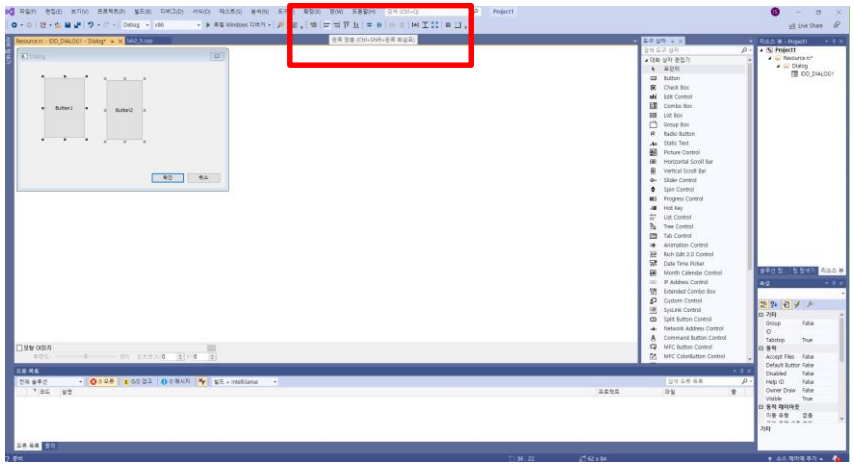


필요한 컨트롤을 선택한 후 대화상자에 추가

대화상자 만들기

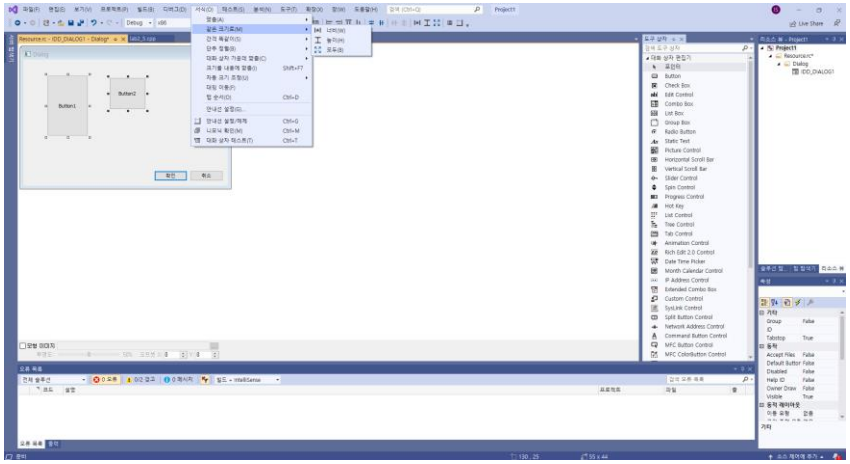


메뉴의 보기 → 도구모음 →
대화상자 편집기 체크



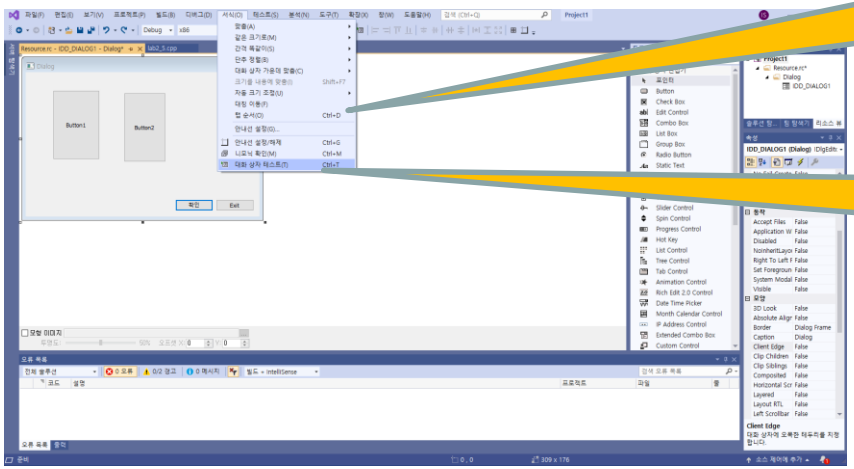
도구상자를 편집할 수 있다.

대화상자 만들기



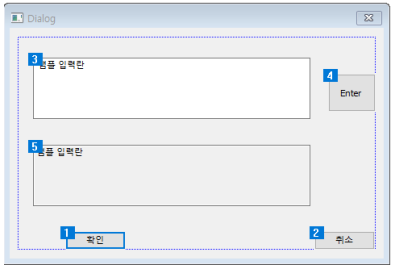
메뉴의 서식 → 도구상자의 서식을 편집할 수 있다.

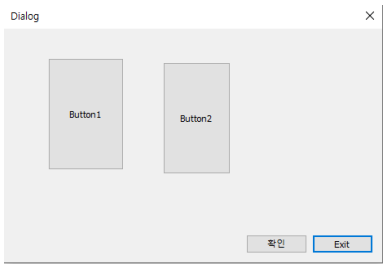
탭순서: 대화상자에서 탭을 이용하여 순서대로 이동할 수 있는 순서를 설정해준다. 마우스로 순서대로 클릭하여 순서를 설정해준다.



대화상자 테스트: 대화상자를 미리 테스트해볼 수 있다.

탭순서 실행한 결과





대화상자 테스트한 결과

2. 대화상자 띄우기

- 리소스에서 대화상자를 만든 후, 함수 호출을 하여 대화상자를 화면에 띄운다.

  DialogBox (hInstance, MAKEMAKEINTRESOURCE(IDD_DIALOG1), hWnd, lpDialogFunc)
- 대화상자를 생성하고 **WM_INITDIALOG** 메시지를 대화상자 프로시저로 보냄

int DialogBox (HINSTANCE hInstance, LPCTSTR lpTemplate,
HWND hWnd, LGPROC lpDialogFunc);

- 대화상자를 생성하고 **WM_INTDIALOG** 메시지를 대화상자 프로시저로 보냄
- 리턴값: IDOK 메시지
- hInstance : 응용의 프로그램 인스턴스 값
- lpTemplate : 대화상자의 ID
- hWnd: 윈도우의 핸들 값
- lpDialogFunc : 대화상자에서 발생하는 메시지 처리용 다이얼로그 함수

- 대화상자 종료하기

BOOL EndDialog (HWND hDlg, int nResult);

- hDlg: 종료할 대화상자 핸들
- nResult : 0 (대화상자 종료상태 표시)

3) 대화상자의 메시지 처리하기

3. 메시지 처리 함수: 대화상자 프로시저

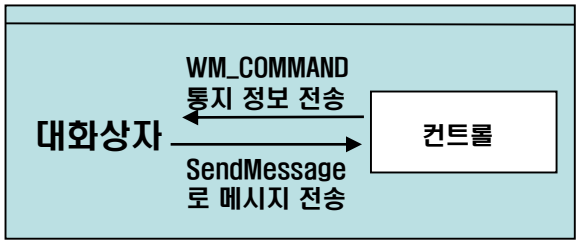
- 대화상자 내에서 발생하는 메시지들을 처리하는 함수로 대화상자 프로시저를 생성

```
BOOL CALLBACK DialogProc (HWND hDlg, UINT iMessage,
                          WPARAM wParam, LPARAM lParam)
```

  - **BOOL 형을 반환: 메시지를 처리했으면 TRUE를 리턴, 그렇지 않으면 FALSE를 리턴**
  - DefWindowProc 함수로 리턴하지 않는다.

- 메시지 처리:
  - **컨트롤에서 대화상자로 오는 메시지: WM_COMMAND**
    - LOWORD (wParam): 메시지를 보낸 컨트롤의 ID
    - HIWORD (wParam): 통지 코드

메시지를 보낸 곳	wParam		lParam
	HIWORD	LOWORD	
컨트롤	컨트롤에 따른 통지정보	컨트롤 ID	컨트롤 핸들값



대화상자의 메시지 처리하기

- **윈도우에 메시지를 보내는 함수: SendMessage**
  - 메시지를 메시지 큐에 넣지 않고 바로 윈도우 프로시저로 보내 처리
  - 메시지가 처리되기 전까지 반환되지 않음,
    - 윈도우 프로시저가 값을 반환해야만 SendMessage도 반환하여 끝마칠 수 있음
  - 부모 윈도우와 차일드 컨트롤 간의 통신, 윈도우 간 데이터 전송 및 처리를 위한 통신
    - 부모 윈도우가 차일드 컨트롤에게 명령을 내리거나 상태를 조사할 때 사용

LRESULT SendMessage (HWND hWnd, UINT iMsg, WPARAM wParam, LPARAM lParam);
- hWnd: 메시지를 전달받을 윈도우 핸들
- iMsg : 전달할 메시지
- wParam, lParam: 메시지의 추가적 정보, 메시지에 따라 다른 정보 반환

- **대화상자를 만들었을 때 발생 메시지: WM_INITDIALOG**
  - 윈도우 프로시저의 WM_CREATE 메시지 의미.
  - 대화 상자에 필요한 초기화 작업
    - wParam: 대화상자에서 제일 먼저 키보드 입력을 받을 컨트롤의 핸들값
    - lParam: 부가적인 정보를 저장하는데 일반적으로 0의 값을 가짐

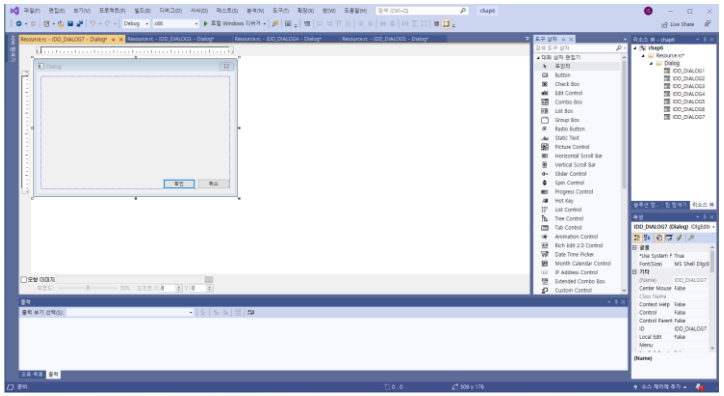
대화상자 띄우기

- **사용 예) 리소스에서 대화상자를 만든 후, 왼쪽 마우스 버튼을 클릭하면 띄우기**

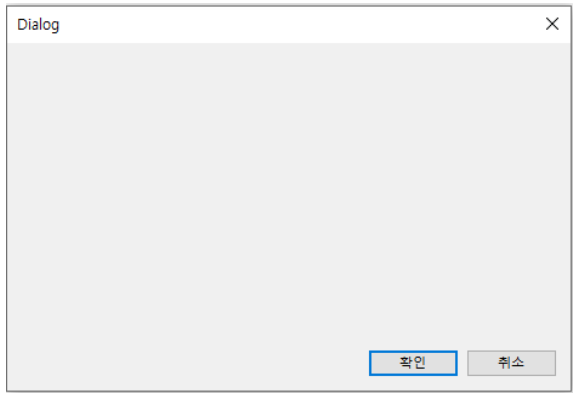
```
#include <windows.h>
#include <TCHAR.h>
#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM);
BOOL CALLBACK Dlalog_Proc (HWND, UINT, WPARAM, LPARAM);

HINSTANCE g_hInst;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpszCmdLine,  int nCmdShow)
{
        HWND        hwnd;
        MSG         msg;
        WNDCLASS    WndClass;

        g_hInst = hInstance;
        ...
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)
{
      switch (iMsg) {
      case WM_CREATE :
           break; ;

      case WM_LBUTTONDOWN :      //--- 마우스 클릭하면 대화상자 띄우기
                DialogBox (g_hInst, MAKEINTRESOURCE(IDD_DIALOG1), hwnd, Dlalog_Proc);
           break;
     }
     return DefWindowProc (hwnd, iMsg, wParam, lParam) ;
}
```

대화상자 띄우기

```
//--- 대화상자 메시지 처리함수
BOOL CALLBACK Dlalog_Proc (HWND hDlg, UINT iMsg, WPARAM wParam, LPARAM lParam)
{
        switch(iMsg){

        case WM_INITDIALOG:
             break;

        case WM_COMMAND:
             switch (LOWORD(wParam))  {
                  case IDOK:                  // 버튼
                            MessageBox (hDlg, L"test", L"test, ", MB_OK);
                            break;

                  case IDCANCEL:              // 버튼
                            EndDialog(hDlg,0);
                            break;
             }
             break;

        case WM_CLOSE:
             EndDialog(hDlg,0);
             break;
        }
        return 0;
}
```



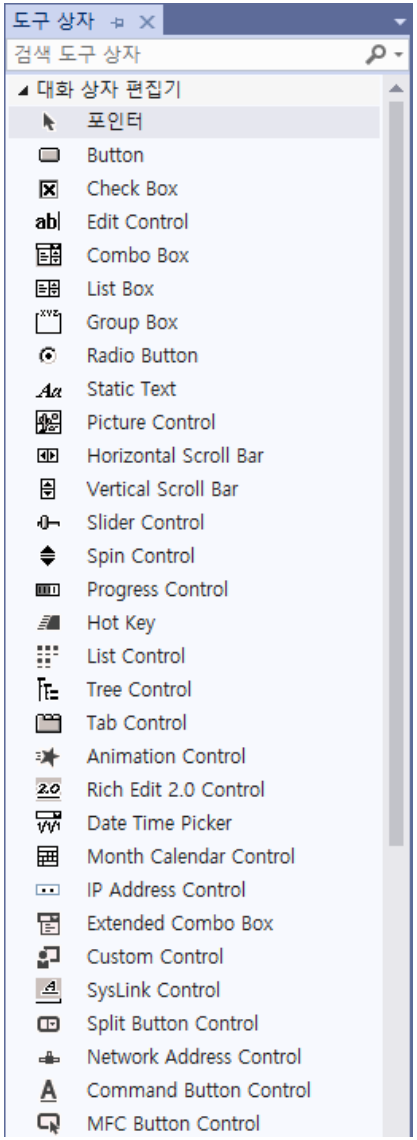
리소스에 처음 생성하면
나타나는 대화상자



실행하면
나타나는 대화상자

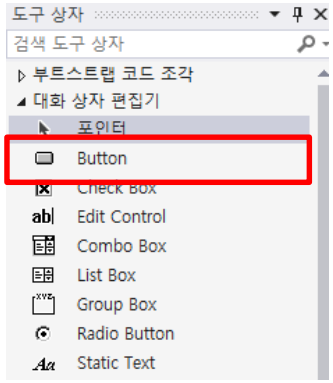
2. 컨트롤 종류

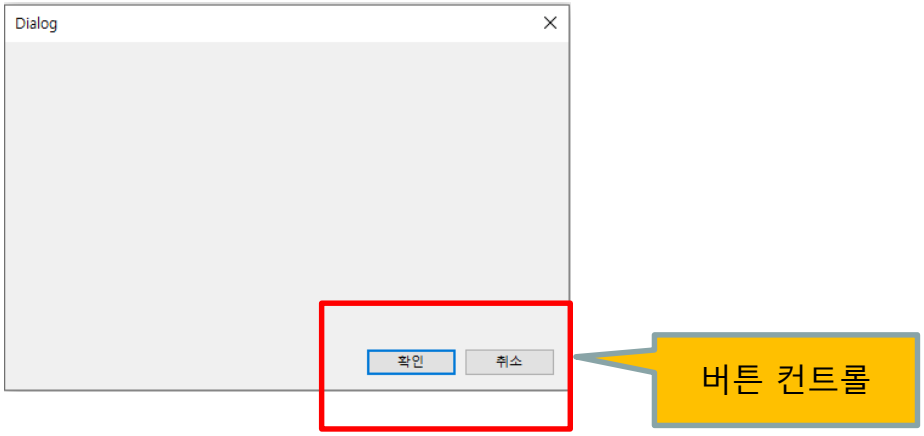
컨트롤	설명
Static Text	정적 텍스트로 보는 것만 가능하고 입력을 할 수 없음
Edit Box	텍스트 입출력을 위한 용도로 사용
Group Box	다른 컨트롤을 묶어 그룹 짓는 역할
Push Button	버튼을 클릭할 때 특정한 함수를 수행하게 할 때 사용
Check Box	특정한 기능을 선택하는 옵션에 사용
Radio Button	그룹 중에서 하나만 선택할 때 사용
List Box	리스트 박스는 여러 항목을 갖는 문자열 정보를 항목별로 보여주는 출력용 컨트롤
Combo Box	콤보박스는 데이터를 입력할 때 목록에서 하나를 선택하게 할 때 사용



1) 버튼 컨트롤

- **버튼 (Button)**
  - 버튼을 눌러 임의의 작업이 이루어진다.
  - 명령을 받아들이는 역할



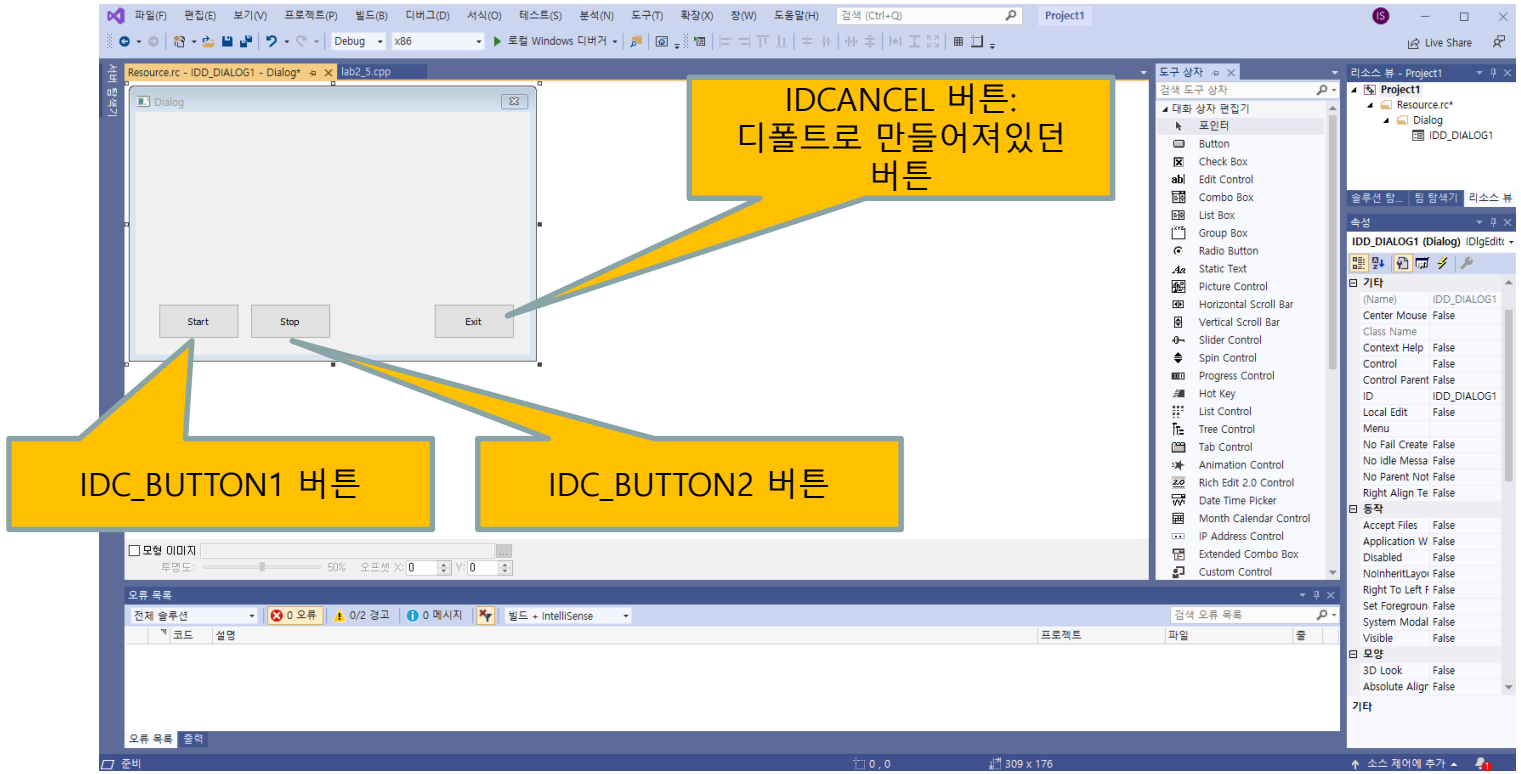


- **버튼 컨트롤에서 오는 통지 정보**

인자값		내용	
wParam	HIWORD(wParam)	컨트롤에 따른 통지 정보	• BN_CLICKED: 버튼이 클릭 되었음 • BN_DBLCLK: 버튼이 더블클릭 되었음 • BN_DISABLE: 버튼이 사용 불능 상태로 되었음 • BN_HILITE: 사용자가 버튼을 선택했음 • BN_SETFOCUS: 버튼이 포커스를 받았음
	LOWORD(wParam)	컨트롤 ID	
lParam		컨트롤 핸들값	

사용 예) 버튼 이용하기

- **버튼의 편집 및 배치**
  - 버튼을 마우스로 선택하여 이동, 크기 변환. 캡션 변경 가능



- **사용 예) 위 그림의 3개의 버튼 메시지 처리**
  - Start 버튼: 대화상자에 문자 출력,
  - Stop 버튼: 메시지 박스 출력
  - Exit 버튼: 대화상자 종료

사용 예

```
LRESULT CALLBACK WndProc (HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)
{
        switch (iMsg) {
        case WM_LBUTTONDOWN :       //--- 마우스 클릭하면 대화상자 띄우기
                DialogBox (g_hInst, MAKEINTRESOURCE(IDD_DIALOG1), hwnd, Dlalog_Proc);
             break;
      }
      return DefWindowProc (hwnd, iMsg, wParam, lParam) ;
}

BOOL CALLBACK Dlalog_Proc (HWND hDlg, UINT iMessage, WPARAM wParam, LPARAM lParam)
{
        HDC hdc;
        switch(iMessage) {
            case WM_INITDIALOG:
                break;
            case WM_COMMAND:
                    switch (LOWORD(wParam)) {
                        case IDC_BUTTON1:                                   //--- Start 버튼
                                hdc = GetDC (hDlg);                         //--- 대화상자의 hdc를 가져옴
                                TextOut (hdc, 0, 0, L " Hello World " , 11);  //--- 대화상자에 출력함.
                                ReleaseDC (hDlg, hdc);
                                break;
                        case IDC_BUTTON2:                                   //--- Stop 버튼
                                MessageBox (hDlg, L"Stop Button", L"test, ", MB_OK);
                                break;
                        case IDCANCEL:                                      //--- Exit 버튼
                                 EndDialog(hDlg,0);
                                 break;
                    }
                    break;
        }
        return 0;
}
```

컨트롤 관련 함수

- **핸들 가져오기: GetDlgItem 함수**
  - 대화상자에 있는 컨트롤의 핸들(HWND)을 구함

HWND GetDlgItem (HWND hDlg, int nIDDlgItem);
- hDlg: 대화상자 핸들
- nIDDlgItem: 핸들을 구할 컨트롤의 ID
- 리턴값: 이 컨트롤의 윈도우 핸들을 리턴

- 예) IDC_BUTTON1 아이디를 가진 컨트롤의 핸들 가져오기

```
HWND hButton;
hButton = GetDlgItem (hDlg, IDC_BUTTON1);
```

- **컨트롤 ID 가져오기: GetDlgCtrlID 함수**
  - 특정 컨트롤의 윈도우 핸들로부터 컨트롤 ID 구함

int GetDlgCtrlID (HWND hWndCtrl);
- hWndCtrl: ID를 구할 컨트롤의 윈도우 핸들
- 리턴값: 컨트롤의 ID

- 예) hButton 컨트롤의 ID 가져오기

```
int id;
id = GetDlgCtrlID (hButton);
```

컨트롤 관련 함수

- **컨트롤의 사용 상태 변경하기: EnableWindow 함수**
  - 컨트롤을 사용가능 상태 또는 사용불능 상태로 만들기

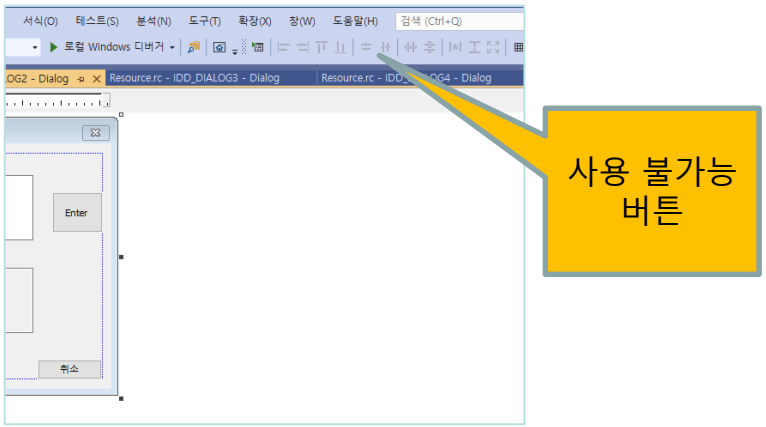
BOOL EnableWindow (HWND hWnd, BOOL bEnable);
- hWnd: 컨트롤의 핸들
- bEnable: 상태 설정 값, TRUE 면 사용 가능 상태, FALSE면 사용 불능 상태

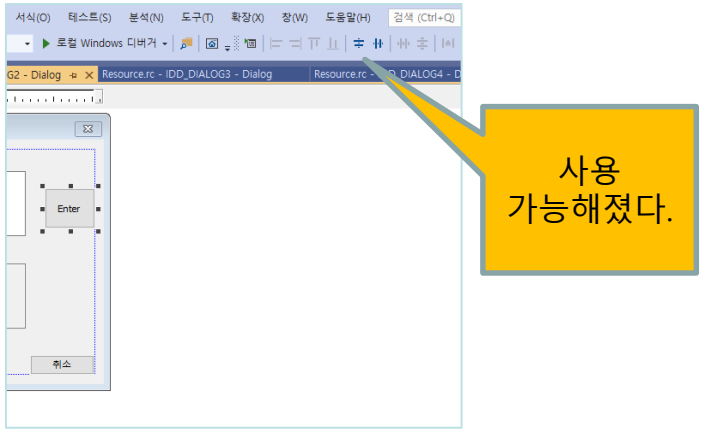
- 예) hButton 컨트롤을 사용 불능 상태로 만들기

```
hButton = GetDlgItem (hDlg, ID_TEST);      //--- ID_TEST 라는 id의 버튼 핸들 가져오기
EnableWindow (hbutton, FALSE);             //--- 버튼을 사용 불능 상태로 만들기
```

- 예) hButton 컨트롤을 사용 가능 상태로 만들기

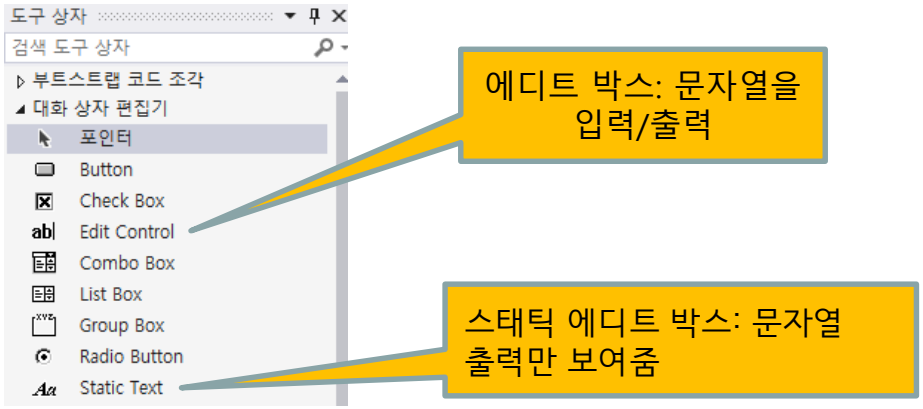
```
hButton = GetDlgItem (hDlg, ID_TEST);      //--- ID_TEST 라는 id의 버튼 핸들 가져오기
EnableWindow (hButton, TRUE);              //--- 버튼을 사용 가능 상태로 만들기
```

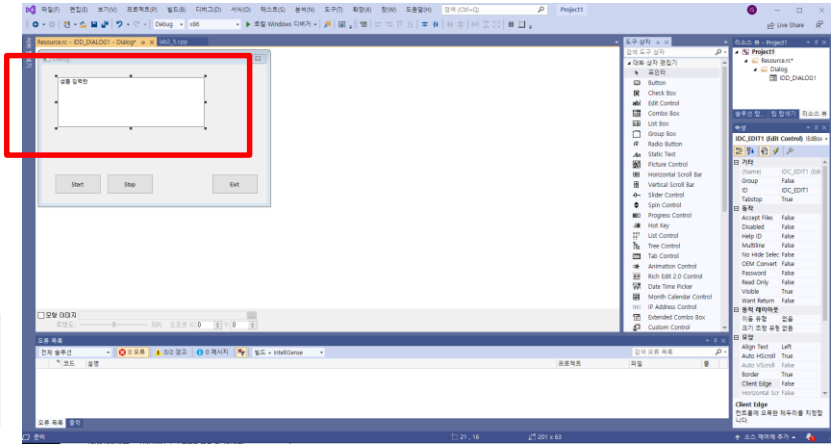




2) 에디트 박스 컨트롤

- **에디트 박스 (Edit Box)**
  - 사용자의 키보드 입력 또는 출력을 위한 편집창
    - Edit control: 입력, 출력 가능
      - Edit Control의 속성으로 Read_Only 로 설정 가능
    - Static Text: 문자열 출력만 가능





- **에디트 박스 통지 정보**

인자값		내용	
wParam	HIWORD(wParam)	컨트롤에 따른 통지 정보	• EN_CHANGE: 에디트 박스내의 내용이 변하였음 • EN_HSCROLL: 에디트 박스의 수평스크롤바를 선택하였음 • EN_VSCROLL: 에디트 박스의 수직스크롤바를 선택하였음 • EN_SETFOCUS: 에디트 박스가 포커스를 받았음
	LOWORD(wParam)	컨트롤 ID	
lParam		컨트롤 핸들값	

컨트롤 관련 함수

- **컨트롤 윈도우에서 텍스트 얻어오기: GetDlgItemText 함수**
  - 대화상자의 컨트롤에 WM_GETTEXT 메시지를 보내서 텍스트를 얻어온다.

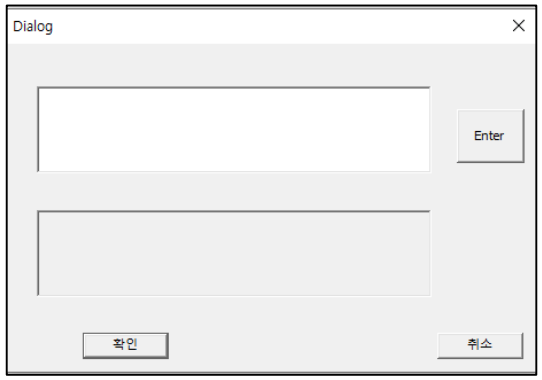
UINT GetDlgItemText (HWND hDlg, int nIDDlgItem, LPTSTR lpString, int nCount);
- 리턴값: 성공하면 읽은 문자수 리턴, 실패하면 0 리턴
- hDlg: 컨트롤을 가지고 있는 대화상자의 핸들
- nIDDlgItem: 컨트롤의 ID
- lpString: 얻어낸 텍스트 스트링을 저장할 버퍼의 주소
- nCount: 버퍼의 길이 (문자열의 길이가 버퍼의 길이보다 길면 문자열은 잘린다)

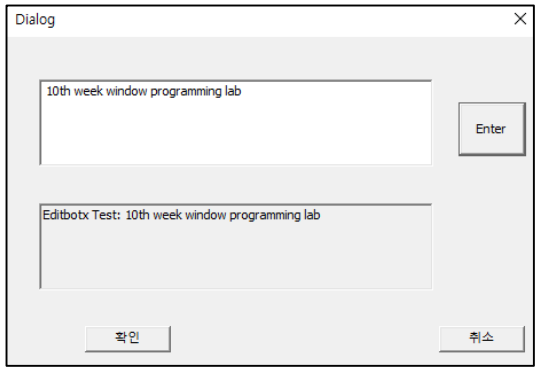
- **컨트롤 윈도우에 텍스트를 출력하기: SetDlgItemText 함수**
  - 대화상자의 컨트롤에 텍스트를 출력한다.

BOOL SetDlgItemText (HWND hDlg, int nIDDlgItem, LPTSTR lpString);
- 리턴값: 성공하면 0이 아닌 값 리턴, 실패하면 0 리턴
- hDlg: 컨트롤을 가지고 있는 대화상자의 핸들
- nIDDlgItem: 컨트롤의 ID
- lpString: 출력할 텍스트 스트링의 시작 주소

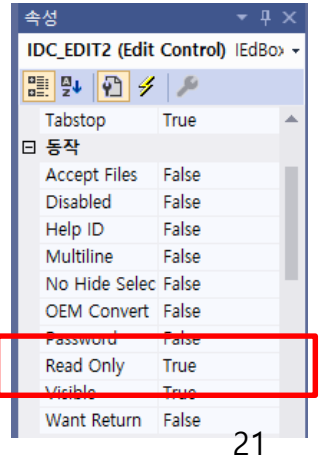
컨트롤 관련 함수

- **사용 예) 버튼과 에디트 박스 사용하여 문자 출력하기**
  - 버튼을 누르면 에디트 박스에 작성한 문장을 얻어 기존의 문자열에 붙여 다른 에디트 박스에 출력하기
    - 붙여넣기 하는 에디트 박스는 출력만 가능하게 한다
  - GetDlgItemText / SetDlgItemText 함수 사용하기





컨트롤 형태	ID	기능
에디트 박스	IDC_EDIT1	문자열을 입력받는다
에디트 박스	IDC_EDIT2	문자열을 출력만한다
버튼	IDC_BUTTON1	버튼을 누르면



사용 예

- **사용 예) 버튼과 에디트 박스 사용하여 문자 출력하기**

```
BOOL CALLBACK Dlg6_ Proc2 (HWND hDlg, UINT iMsg, WPARAM wParam, LPARAM lParam)
{
        static TCHAR editText[100];
        TCHAR resultText[100] = L"Editbotx Test: ";

        switch (iMsg)
        {
        case WM_COMMAND:
                switch (LOWORD(wParam)) {
                case IDC_BUTTON1:
                                GetDlgItemText (hDlg, IDC_EDIT1, editText, 100);    //--- 첫번째 에디트 박스에서 숫자를 가져옴
                                lstrcat(resultText, editText);                      //--- resultText 문자열에 덧 붙임
                                SetDlgItemText (hDlg, IDC_EDIT2, resultText);       //--- 두번째 에디트 박스에 문자열을 출력함
                                break;
                case IDOK:
                                SetDlgItemText (hDlg, IDC_EDIT1, L"");
                                SetDlgItemText (hDlg, IDC_EDIT2, L"");
                                break;
                case IDCANCEL:
                                EndDialog(hDlg, 0);
                                break;

                }
        }
        return 0;
}
```

컨트롤 관련 함수

- **컨트롤 윈도우에서 문자열을 정수값으로 변환하여 읽어오기: GetDlgItemInt 함수**
  - WM_GETTEXT 메시지로 텍스트를 읽어 정수형으로 변환하여 리턴

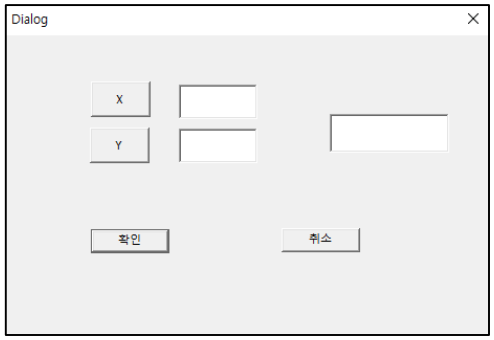
UINT GetDlgItemInt (HWND hDlg, int nIDDlgItem, BOOL*lpTranslated, BOOL bSigned);
- 리턴값: 변환된 정수값 , 실패시 0 리턴
- hDlg: 컨트롤을 가지고 있는 윈도우 핸들
- nIDDlgItem: 컨트롤의 ID
- lpTranslated: 변환의 성공여부 리턴받는 변수. 변환되면 TRUE, 아니면 FALSE 로 설정 (에러 검사를 할 필요가 없을 때는 NULL로 설정)
- bSigned: 부호가 있는 정수인지 지정. 부호를 갖는 정수(int)이면 TRUE, 부호없는 정수(UINT)이면 FALSE

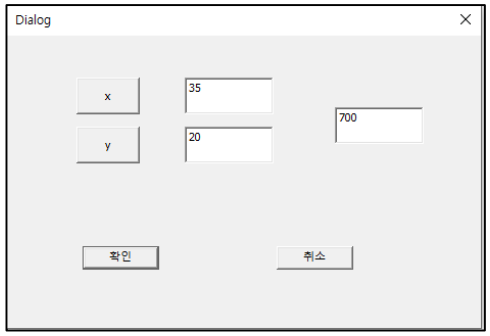
- **컨트롤 윈도우에 정수값을 출력하기: SetDlgItemInt 함수**
  - 대화상자의 컨트롤에 정수값을 출력

BOOL SetDlgItemInt (HWND hDlg, int nIDDlgItem, UINT uValue, BOOL bSigned);
- 리턴값: 성공하면 0이 아닌 값을 리턴, 실패하면 0 리턴
- hDlg: 컨트롤을 가지고 있는 윈도우 핸들
- nIDDlgItem: 컨트롤의 ID
- uValue: 컨트롤에 저장할 정수값
- bSigned: 부호가 있는 정수인지를 지정, 부호를 갖는 정수(int)이면 TRUE, 부호없는 정수(UINT)이면 FALSE

사용 예

- **사용 예) 버튼과 에디트 박스 사용하기**
  - GetDlgItemInt / SetDlgItemInt 함수 사용하기
  - 2개의 에디트 박스에 각각 숫자를 입력받고, 버튼을 눌러 값을 가져온 후 확인 버튼을 누르면 두 숫자의 곱셈값이 다른 에디트 박스에 출력된다.





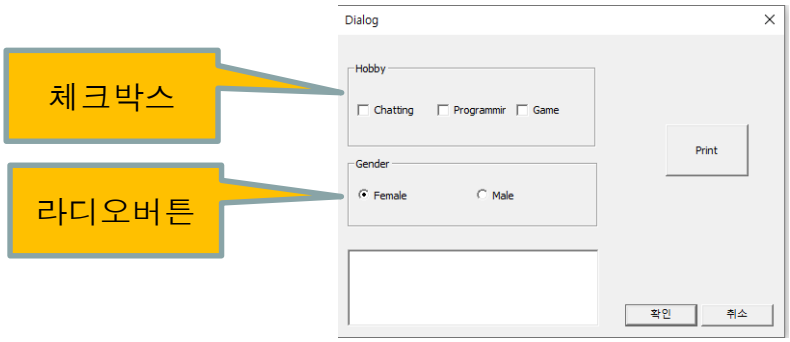
컨트롤 형태	ID	기능
에디트 박스	IDC_EDIT1	숫자를 입력받는다
에디트 박스	IDC_EDIT2	숫자를 입력받는다
에디트 박스	IDC_EDIT3	숫자를 출력한다
버튼	IDC_BUTTON1	버튼을 누르면 IDC_EDIT1에서 숫자를 가져와 IDC_EDIT3에 출력한다
버튼	IDC_BUTTON2	버튼을 누르면 IDC_EDIT2에서 숫자를 가져와 IDC_EDIT3에 출력한다
버튼	IDC_BUTTON3	IDC_EDIT1의 숫자와 IDC_EDIT2의 숫자를 곱해서 IDC_EDIT3 에 출력한다
버튼	IDCANCEL	대화상자 종료

- **사용 예) 버튼과 에디트 박스 사용하기**

```
BOOL CALLBACK Dlg6_ 3 (HWND hDlg, UINT iMsg, WPARAM wParam, LPARAM lParam)
{
    static int x, y, result;

    switch (iMsg)
        {
        case WM_COMMAND:
            switch (LOWORD(wParam)) {
            case IDC_BUTTON1:
                        x = GetDlgItemInt (hDlg, IDC_EDIT1, NULL, TRUE);   //--- x버튼 옆의 에디트 박스에서 숫자를 가져옴
                        SetDlgItemInt(hDlg, IDC_EDIT3, x, TRUE);           //--- 오른쪽 에디트 박스에 숫자를 적음
                        break;
            case IDC_BUTTON2:
                        y = GetDlgItemInt (hDlg, IDC_EDIT2, NULL, TRUE); //--- y버튼 옆의 에디트 박스에서 숫자를 가져옴
                        SetDlgItemInt (hDlg, IDC_EDIT3, y, TRUE);          //--- 오른쪽 에디트 박스에 숫자를 적음
                        break;
            case IDOK:
                        result = x * y;
                        SetDlgItemInt (hDlg, IDC_EDIT3, result, TRUE);     //  두 숫자의 곱을 적음
                        break;
            case IDCANCEL:
                        EndDialog (hDlg, 0);
                        break;

            }
        }
        return 0;
    }
```

3) 체크박스와 라디오 버튼 컨트롤

- **체크버튼과 라디오 버튼**
  - 체크 박스: 복수 항목 선택 가능
  - 라디오 버튼: 한 항목만 선택 가능

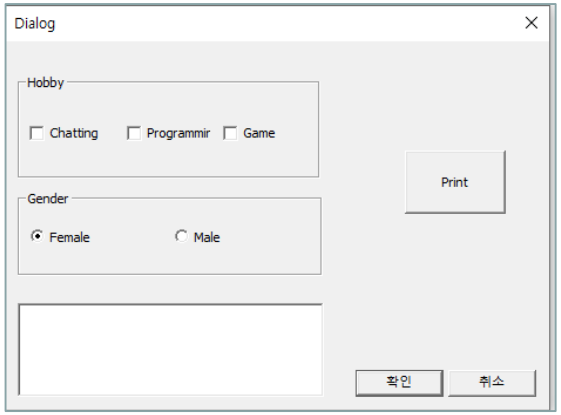


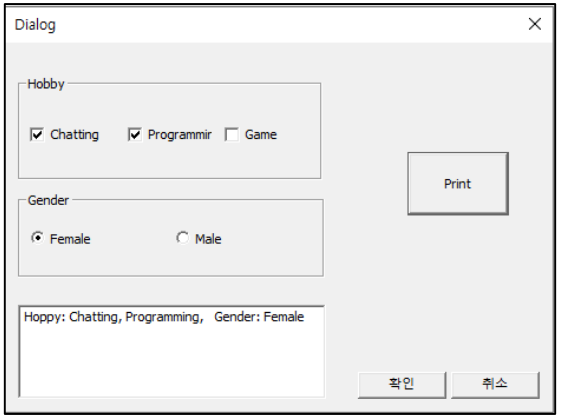
컨트롤에 보내는 메시지	의미	리턴 값 또는 체크 박스 상태
BM_GETCHECK	체크박스가 현재 체크되어 있는 상태인지 조사	• BST_CHECKED: 현재 체크되어 있다. • BST_UNCHECKED: 현재 체크되어 있지 않다. • BST_INDETERMINATE: 체크도 아니고 비 체크도 아닌 상태
BM_SETCHECK	체크 박스의 체크 상태를 변경, wParam에 변경할 체크상태를 보내준다	

LRESULT SendMessage (HWND hWnd, UINT Msg, WPARAM wParam, LPARAM lParam);
- 메시지를 메시지 큐에 넣지 않고 바로 윈도우 프로시저로 보냄
- hWnd: 메시지를 전달받을 윈도우 핸들
- Msg : 전달할 메시지
- wParam, lParam 메시지의 추가적 정보, 메시지에 따라 다른 정보 반환

사용 예

- **사용 예) 취미와 성별을 선택 후 선택한 결과 출력하기**
  - 취미: 1개이상 선택 가능 – 체크박스
  - 성별: 1개 선택 – 라디오버튼 (그룹 설정: 첫 번째 라디오 버튼의 group을 true로 설정)
  - 그 외, 버튼 에디트박스 사용
    - 출력 버튼, 클리어 버튼 (SendMessage 함수 사용)

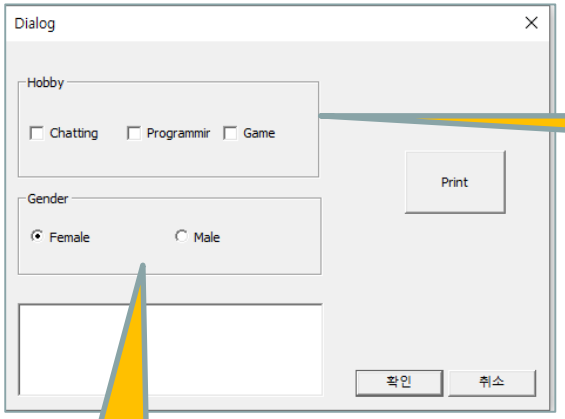




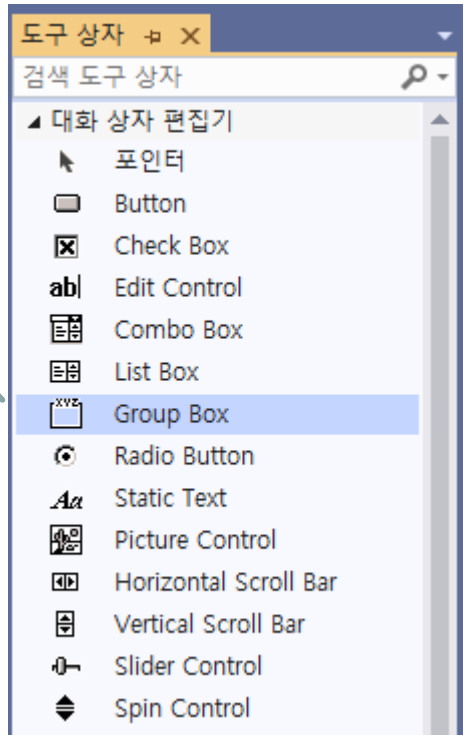
BOOL CheckRadioButton (HWND hDlg, int nIDFirstButton,
int nIDLastButton, int nIDCheckButton);

- 처음 선택될 라디오 버튼 선택
- hDlg: 라디오 버튼을 가지는 부모 윈도우(또는 대화상자)의 핸들
- nIDFirstButton : 각 그룹의 시작 버튼 아이디
- nIDLastButton: 각 그룹의 끝 버튼 아이디
- nIDCheckButton: 선택될 버튼의 아이디

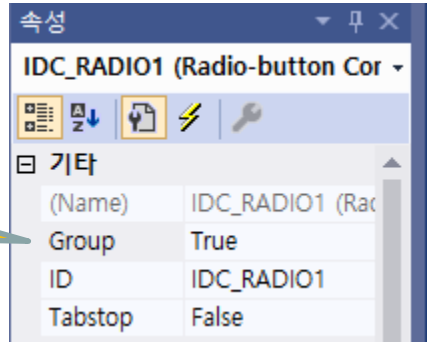
사용 예



그룹 박스



라디오 버튼의 그룹이 여러 개 있을 경우, 한 그룹의 첫번째 라디오 버튼 ➔ 그룹을 true로 설정



사용 예

```
BOOL CALLBACK DlalogProc (HWND hDlg, UINT iMsg, WPARAM wParam,LPARAM lParam)
{

    static int check[3], radio;
    TCHAR hobby[][20] = { L"Chatting", L"Programming", L"Game" };
    TCHAR gender[][20] = { L"Female", L"Male" };
    TCHAR output[100];
    HWND hCheck[3], hRadio[2];

    switch(iMsg)
    {
        case WM_INITDIALOG:
            CheckRadioButton (hDlg, IDC_RADIO1 , IDC_RADIO2, IDC_RADIO1);      //--- 시작 버튼, 끝 버튼, 체크할 버튼
        break;

        case WM_COMMAND:
        switch (LOWORD(wParam)) {
            case IDC_CHECK1:                                                    //--- 취미1
                check[0] = 1 – check[0];
            break;
            case IDC_CHECK2:                                                    //--- 취미2
                check[1] = 1 – check[1];
            break;
            case IDC_CHECK3:                                                    //--- 취미3
                check[2] = 1 - check[2];
            break;
```

사용 예

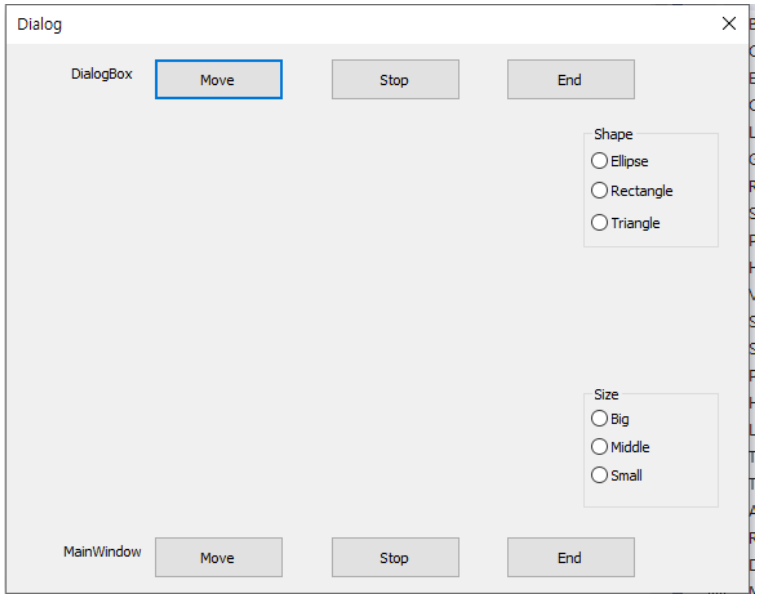
```
        case IDC_RADIO1:                                        //--- 성별1
                radio = 0;
        break;
        case IDC_RADIO2:                                        //--- 성별2
                radio = 1;
        break;

        case IDC_BUTTON1:                                       //--- 출력 버튼
                wsprintf (output, L"Hoppy: %s, %s, %s ₩n Gender: %s", check[0] ? hobby[0] : L"",
                                        check[1] ? hobby[1] : L"", check[2] ? hobby[2] : L"", gender[radio]);
                SetDlgItemText (hDlg, IDC_EDIT1, output);
        break;

        case IDOK:                                              //--- 확인 버튼 (모든 값을 초기화)
                lstrcpy (output, L" ");
                SetDlgItemText (hDlg, IDC_EDIT1, output);

                radio = check[0] = check[1] = check[2] = 0;
                hCheck[0] = GetDlgItem (hDlg, IDC_CHECK1);
                hCheck[1] = GetDlgItem(hDlg, IDC_CHECK2);
                hCheck[2] = GetDlgItem(hDlg, IDC_CHECK3);

                for ( int I = 0; I < 3; I ++ )
                        SendMessage (hCheck[i], BM_SETCHECK, BST_UNCHECKED, 0);
                CheckRadioButton (hDlg, IDC_RADIO1, IDC_RADIO2, IDC_RADIO1);
                break;

        case IDCANCEL:
                EndDialog(hDlg, 0);
        break;
        }
    }
    return 0;
}
```

실습 6-1

- **대화상자 또는 부모 윈도우에서 움직이는 원**
  - 부모 윈도우에서 왼쪽 마우스 버튼을 누르면 대화상자가 나타난다.
  - 대화상자 안에 다음의 컨트롤들을 만든다.
    - **스태틱 에디트 박스: Dialog Box 이름 붙이기**
    - 버튼1: **대화상자에서 원이 자유방향으로 튀기기를** 시작하기 위한 시작버튼
    - 버튼2: 대화상자에서 움직이는 원을 멈추기 위한 정지버튼
    - 버튼3: 대화상자를 닫는 종료버튼

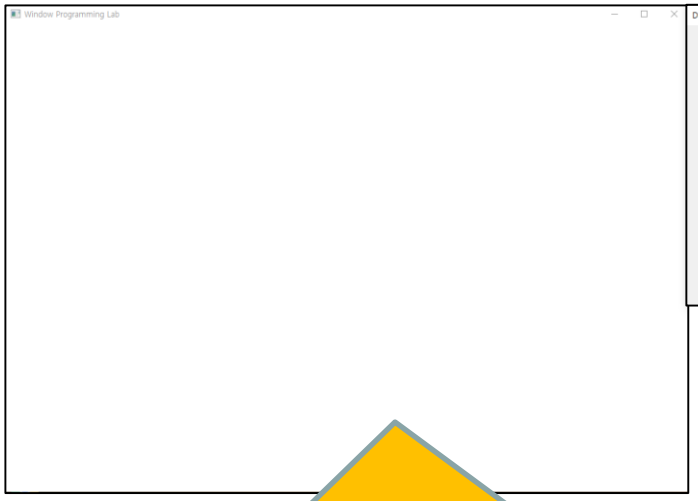
    - **스태틱 에디트 박스: Main Window 이름 붙이기**
    - 버튼4: **부모 윈도우에서 원이 자유방향으로 튀기기를** 시작하기 위한 시작버튼
    - 버튼5: 부모 윈도우에서 움직이는 원을 멈추기 위한 정지버튼
    - 버튼6: 프로그램을 닫는 종료버튼

    - **라디오 버튼: 도형 모양 선택**
      - 라디오 버튼 1: 원
      - 라디오 버튼 2: 사각형
      - 라디오 버튼 3: 삼각형
    - **라디오 버튼: 도형 크기 선택**
      - 라디오 버튼 4: small
      - 라디오 버튼 5: middle
      - 라디오 버튼 6: big

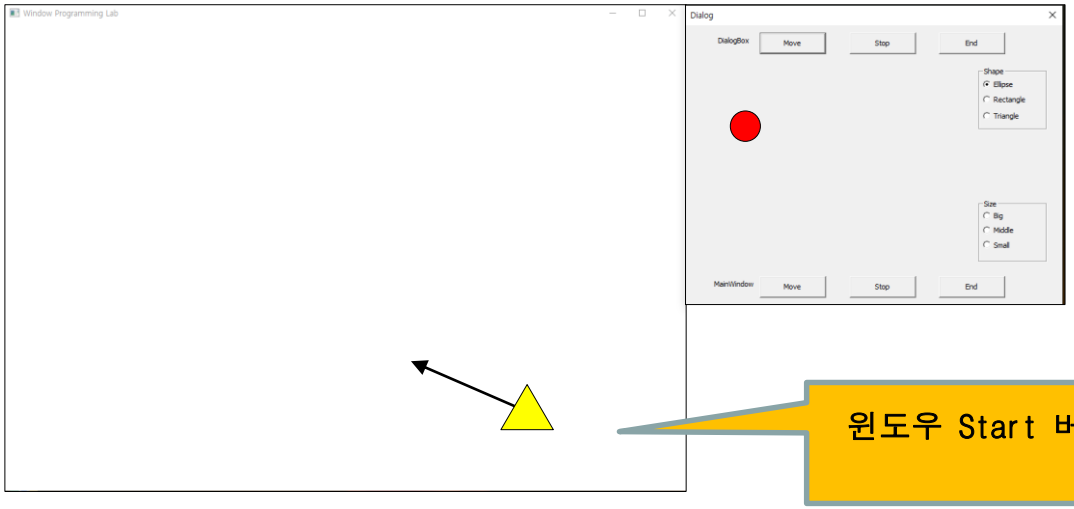


실습 6-1

시작 명령어에 따라 대화상자가 열림



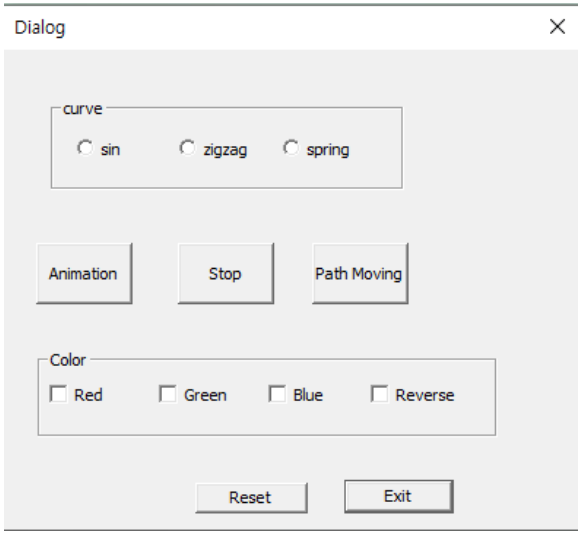
대화상자 Start 버튼을 누르니 대화상자 내에서 원이 튕기기를 한다. (컨트롤들 무시하고 대화상자 내부에서 공 튀기기 진행



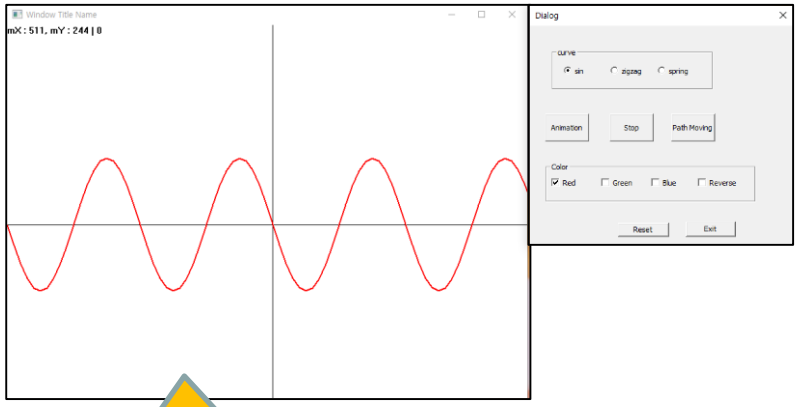
윈도우 Start 버튼을 누르니 윈도우 내에서 삼각형이 튕기기를 한다.

실습 6-2

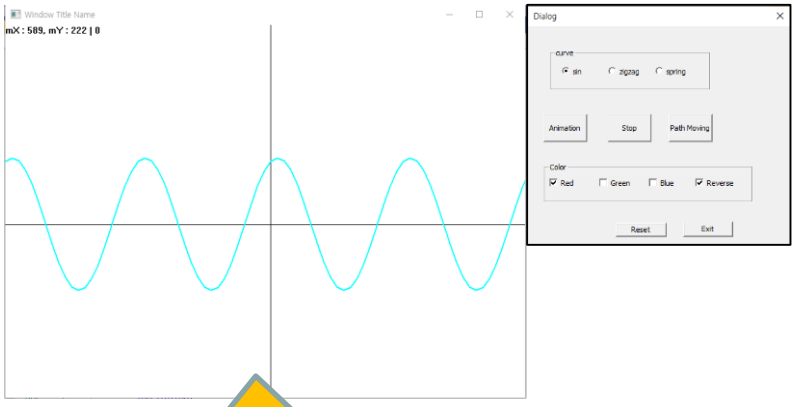
- **컨트롤 사용하기**
  - 대화상자에 라디오 버튼, 체크박스, 버튼을 만들고, 경로를 그리고 그 경로에서 원이 이동하는 애니메이션 만들기
  - 라디오 버튼
    - 라디오 버튼1: 사인 곡선 그리기
    - 라디오 버튼2: 사선을 이용한 지그재그 그리기
    - 라디오 버튼3: 스프링 그리기
  - 버튼
    - 버튼1 (Animation 버튼): 경로를 좌측으로 이동하는 애니메이션
    - 버튼2 (Stop 버튼): 애니메이션 멈추기 (버튼1, 버튼3에 대한 이동)
    - 버튼3 (Path Moving 버튼): 라디오 버튼으로 선택한 경로로 원이 움직이기 (곡선이 멈춰 있을 때 적용)
    - 버튼4: 리셋 버튼
    - 버튼5: 프로그램 종료
  - 체크버튼 (1개 이상 선택 가능: 1개 이상이 선택되면 선택된 색이 합성되어 결정된다)
    - 체크 버튼1: 선의 색이 빨강
    - 체크 버튼2: 선의 색이 초록
    - 체크 버튼3: 선의 색이 파랑
    - 체크 버튼4: 반전 (버튼 1, 2, 3 선택의 반전색)



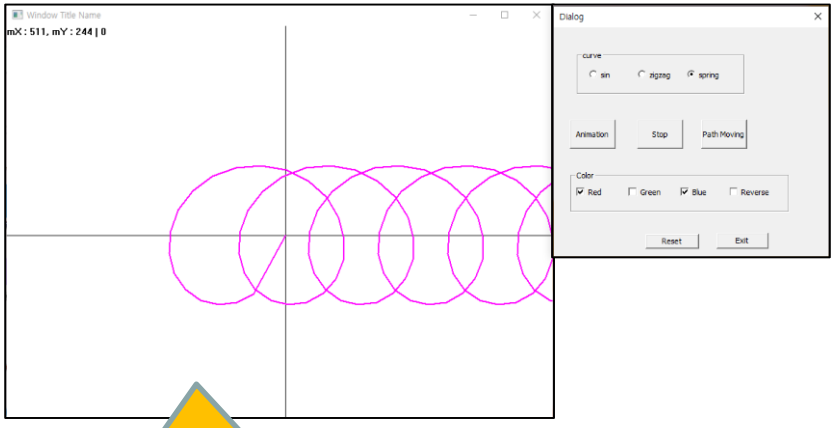
실습 6-2



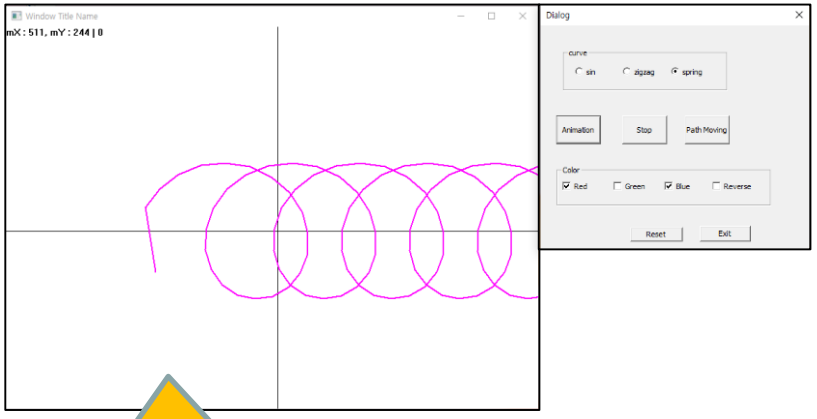
사인 곡선: 빨강색으로 그림



사인 곡선: 빨강색의 반전색으로 그려지고 애니메이션 되어 그림이 바뀌었다.

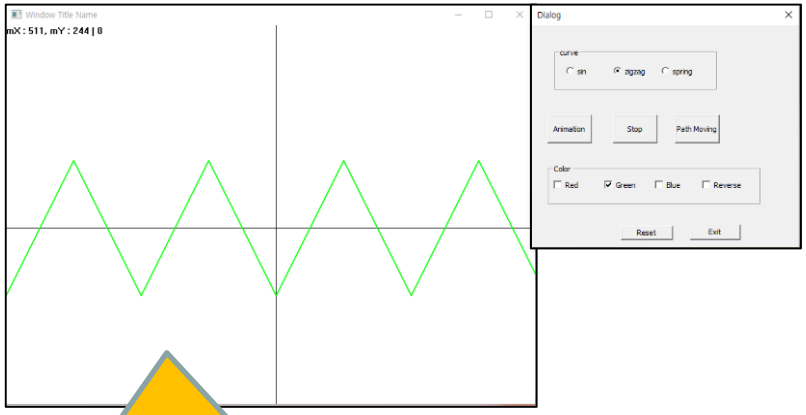


스프링 곡선: 빨강색+파랑색으로 그림

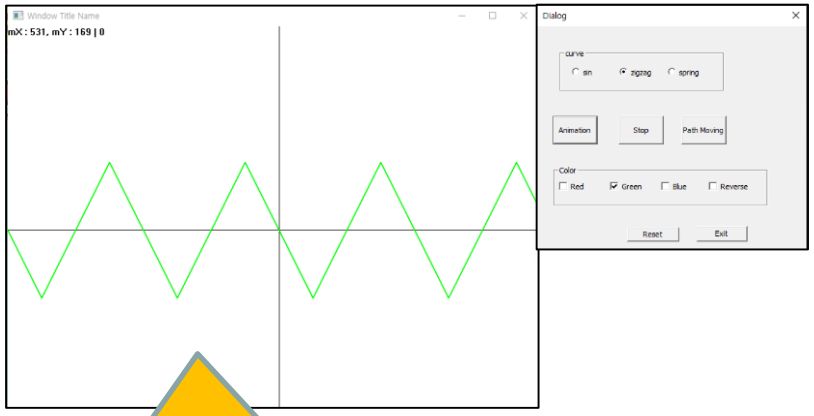


스프링 곡선 애니메이션

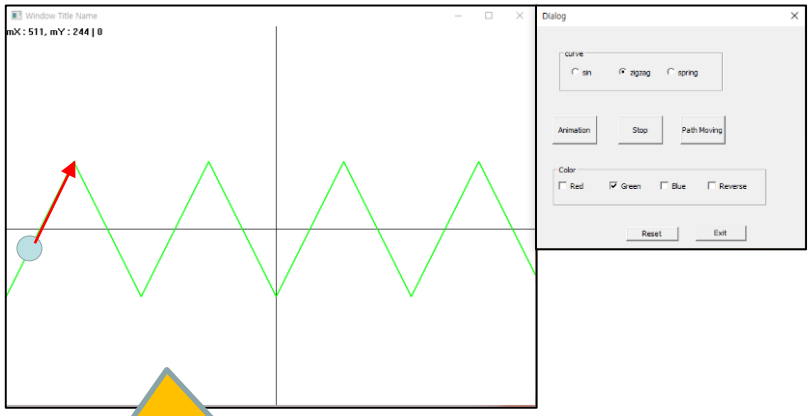
실습 6-2



지그재그 곡선: 초록색으로 그림



지그재그 곡선 애니메이션 (왼쪽으로 이동함)



Path Moving 버튼을 누르면 원이 경로(여기에서는 지그재그)를 따라 이동한다.

오늘은

- 대화상자 만들기
  - 리소스에서 대화상자 만들기
  - 대화상자 띄우기
  - 메시지 처리하기
- 컨트롤 만들기
  - 버튼, 체크박스, 라디오 버튼, 에디트 박스

- 다음 시간에는
  - 추가의 컨트롤 만들기

- 다음 시간에 만나요~~