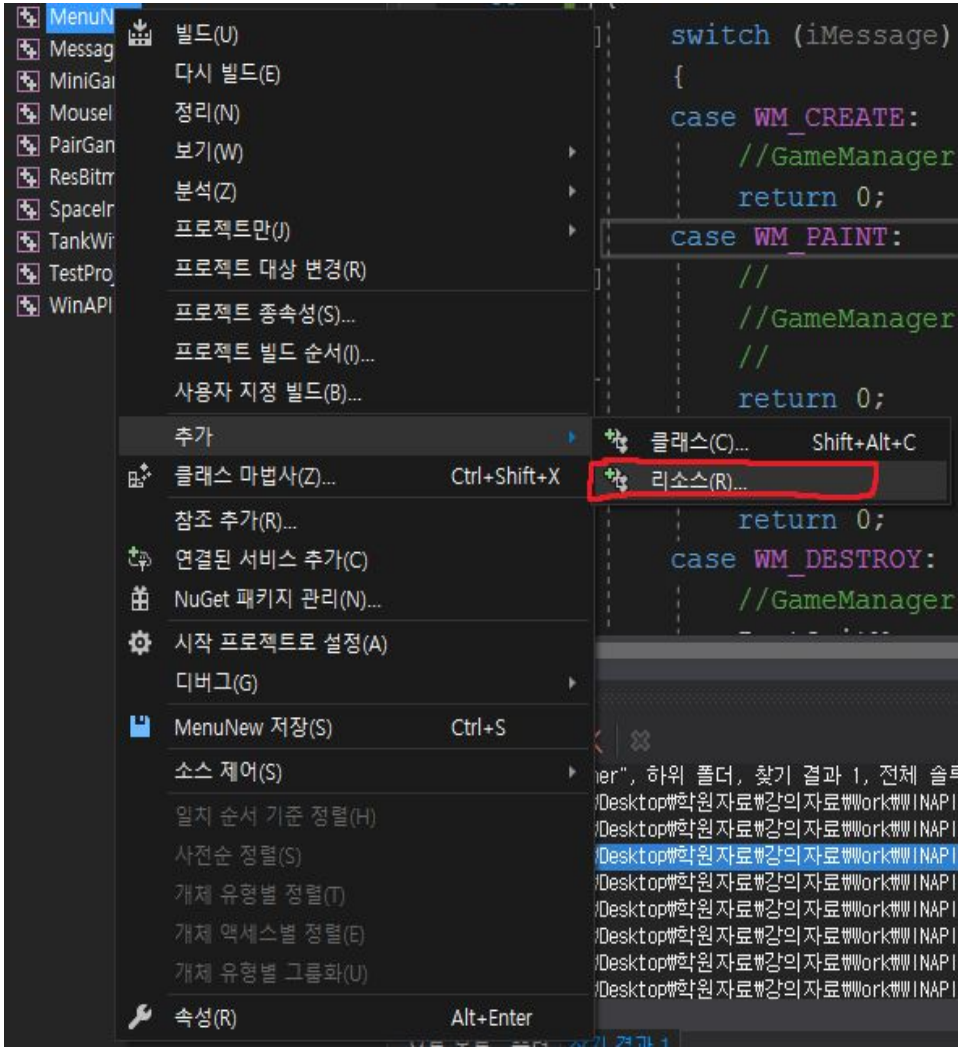
리소스작성,컨트롤,Dailog

SoulSeek

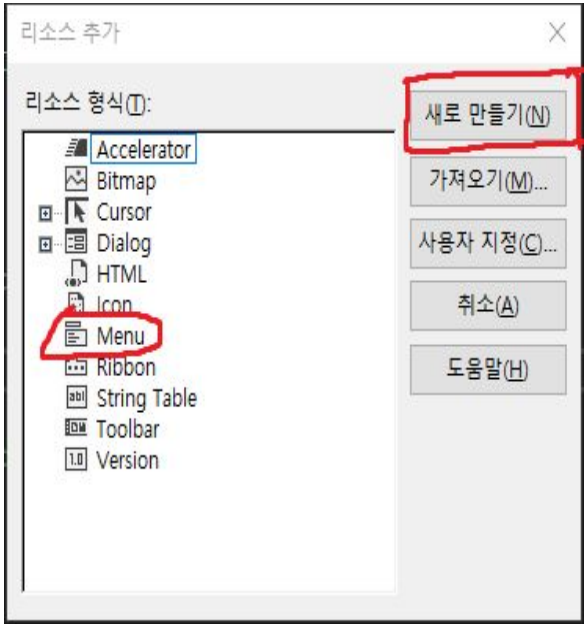
리소스 작성(Menu)

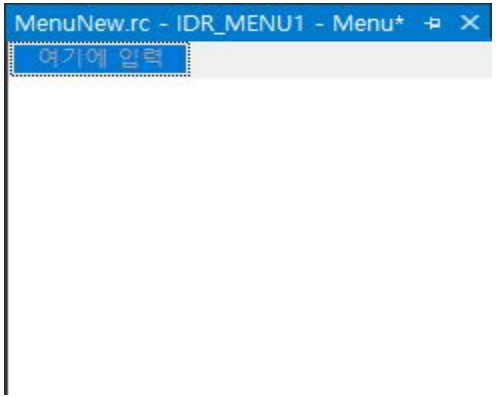
-Menu 작성

-메뉴를 리소스편집기에서 생성한다.

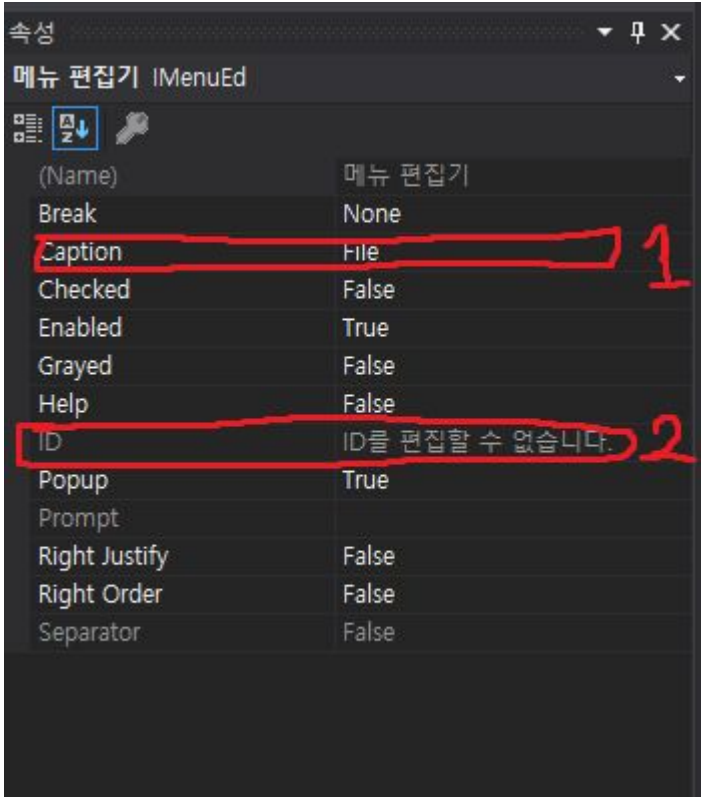


-Menu를 선택해서 생성하고 하위메뉴도 함께 작성한다.





-리소스 편집기에서 클릭을하면 속성창을 볼 수 있는데 그 곳에서 Caption과 ID를 확인할 수 있다 Main Menu의 ID는 변경할 수 없다.



-추가된 것들을 resource.h에서 확인가능하다.

```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++에서 생성한 포함 파일입니다.
// MenuNew.rc에서 사용되고 있습니다.
//
#define IDR_MENU1                       101
#define IDS_STRING102                   102
#define ID_FILE_OPEN                    40001

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        103
#define _APS_NEXT_COMMAND_VALUE         40002
#define _APS_NEXT_CONTROL_VALUE         1001
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

-윈도우 클래스에서 메뉴 작성을 추가해준다.

```
WndClass.cbClsExtra = 0;
WndClass.cbWndExtra = 0;
WndClass.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);
WndClass.hCursor = LoadCursor(NULL, IDC_ARROW);
WndClass.hIcon = LoadIcon(NULL, IDI_APPLICATION);
WndClass.hInstance = hInstance;
WndClass.lpfnWndProc = WndProc;
WndClass.lpszClassName = g_szClassName;
WndClass.lpszMenuName = NULL;

//메뉴 리소스를 작성해서 윈도우 클래스에 배치
WndClass.lpszMenuName = MAKEINTRESOURCE(IDR_MENU1);

WndClass.style = CS_HREDRAW | CS_VREDRAW;
RegisterClass(&WndClass);
```

-선택 가능한 하위메뉴를 선택되었을때, WM_COMAND 라는 메시지를 받게된다.

-해당 메뉴의 ID로 받아서 동작을 처리하면된다.

-LOWORD(wParam)은 메뉴, 액셀러레이터

컨트롤의 ID등이 해당된다.

-HIWORD(wParam)은 컨트롤이 보내주는 통지

메세지, 메뉴가 선택된 경우는 0이되며

액셀러레이터가 선택된 경우는 1이된다.

```
switch (iMessage)
{
case WM_CREATE:
    return 0;

case WM_COMMAND:
    switch (LOWORD(wParam))
    {
    case ID_FILE_OPEN:


        break;
    }

    return 0;
```

Control

-사용자와의 인터페이스를 이루는 도구.

-Window안의 작은 Window.

-WNDCLASS 의해 정의된 것을 사용.

윈도우 클래스	컨트롤
button	버튼, 체크, 라디오
static	텍스트
scrollbar	스크롤 바
edit	에디트
listbox	리스트 박스
combobox	콤보 박스

*Button

-CreateWindow이용해 생성한다.

-WM_COMMAND메시지로 전달 받는다.

```
CreateWindow(TEXT("button"), TEXT("Click Me"), WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON,
    20, 20, 100, 25, hWnd, (HMENU)0, g_hInst, NULL);
```

-버튼이 가질 수 있는 스타일

스타일	속성
BS_PUSHBUTTON	푸시 버튼
BS_DEFPUSHBUTTON	디폴트 푸시 버튼
BS_CHECKBOX	체크 박스
BS_3STATE	3가지 상태를 가지는 체크 박스
BS_AUTOCEHCKBOX	자동 체크 박스
BS_AUTO3STATE	3가지 상태를 가지는 자동 체크 박스
BS_RADIOBUTTON	라디오 버튼
BS_GROUPBOX	그룹 박스

*WM_COMMAND의 모양.

```
case WM_COMMAND:
        switch(LOWORD(wParam) {        // ID에 따른 분기
        case 메뉴1:처리1;break;
        case 메뉴2:처리2;break;
        case 액셀러레이터1:처리3;break;
        case 컨트롤1:
                switch(HIWORD(wParam)) {// 통지 코드에 따른 분기
                case 통지코드1:처리1;break;
                case 통지코드2:처리2;break;
                ..........
                }
                break;
        }
return 0;
```

* CheckBox

-자동체크박스와 수동체크박스

```
HWND cBox, autocBox;
BOOL bElipse = FALSE;
```

- 옵션을 바꿔서 생성한다.

```
//CheckBox Button
//수동 체크박스
cBox = CreateWindow(TEXT("button"), TEXT("Draw Ellipse?"), WS_CHILD | WS_VISIBLE
    | BS_CHECKBOX, 20, 20, 160, 25, hWnd, (HMENU)0, g_hInst, NULL);

//자동 체크박스
autocBox = CreateWindow(TEXT("button"), TEXT("Good bye Message?"), WS_CHILD | WS_VISIBLE
    | BS_AUTOCHECKBOX, 20, 50, 160, 25, hWnd, (HMENU)1, g_hInst, NULL);
```

-체크박스 컨트롤 메시지

메시지	설명
BM_GETCHECK	체크 박스가 현재 체크되어 있는 상태인지를 조사하며 추가정보는 없다.
BM_SETCHECK	체크 박스의 체크 상태를 변경하며 wParam에 변경할 체크 상태를 보내주면 된다.

-상태

상수	의미
BST_CHECKED	현재 체크되어 있다.
BST_UNCHECKED	현재 체크되어 있지 않다.
BST_INDETERMINATE	체크도 아니고 안체크도 아닌 상태

-수동 체크박스 메시지 처리

```
case 0:
    //수동 : 체크여부를 판단하고  체크로 바꿔주고 명령을 준다.
    if (SendMessage(cBox, BM_GETCHECK, 0, 0) == BST_UNCHECKED)
    {
        //체크로 변환
        SendMessage(cBox, BM_SETCHECK, BST_CHECKED, 0);
        //명령수행.
    }
    else
    {
        //체크로 변환
        SendMessage(cBox, BM_SETCHECK, BST_UNCHECKED, 0);
    }
    //상태가 변경되어서 표현해줘야하기 때문에 다시 그려줘야한다.
    InvalidateRect(hWnd, NULL, TRUE);

    break;
```

-자동체크박스 메시지 처리

```
case WM_DESTROY:

    //자동 : 체크박스 클릭 순간에는 특정 명령을 내릴 수 없고 체크 상태만 파악해서 사용.
    if (SendMessage(autocBox, BM_GETCHECK, 0, 0) == BST_CHECKED)
        MessageBox(hWnd, "Good bye", "Check", MB_OK);

    PostQuitMessage(0);
    return 0;
```

*RadioButton

-체크박스와 비슷하다.

```
//RadioButton
rRect = CreateWindow(TEXT("button"), TEXT("Rectangle"), WS_CHILD | WS_VISIBLE
        | BS_RADIOBUTTON, 20, 20, 160, 25, hWnd, (HMENU)0, g_hInst, NULL);

rEllip = CreateWindow(TEXT("button"), TEXT("Ellipse"), WS_CHILD | WS_VISIBLE
    | BS_RADIOBUTTON, 20, 20, 160, 25, hWnd, (HMENU)0, g_hInst, NULL);

rBlack = CreateWindow(TEXT("button"), TEXT("Black"), WS_CHILD | WS_VISIBLE
    | BS_RADIOBUTTON, 20, 20, 160, 25, hWnd, (HMENU)0, g_hInst, NULL);

rRed = CreateWindow(TEXT("button"), TEXT("Red"), WS_CHILD | WS_VISIBLE
    | BS_RADIOBUTTON, 20, 20, 160, 25, hWnd, (HMENU)0, g_hInst, NULL);

rBlue = CreateWindow(TEXT("button"), TEXT("Blue"), WS_CHILD | WS_VISIBLE
    | BS_RADIOBUTTON, 20, 20, 160, 25, hWnd, (HMENU)0, g_hInst, NULL);
```

-그룹화

```
//GroupButton
CreateWindow("button", "Graph", WS_CHILD | WS_VISIBLE |
    BS_GROUPBOX, 5, 5, 120, 110, hWnd, (HMENU)0, g_hInst, NULL);

rRect = CreateWindow(TEXT("button"), TEXT("Rectangle"), WS_CHILD | WS_VISIBLE
        | BS_AUTORADIOBUTTON, 20, 20, 160, 25, hWnd, (HMENU)ID_R1, g_hInst, NULL);

rEllip = CreateWindow(TEXT("button"), TEXT("Ellipse"), WS_CHILD | WS_VISIBLE
    | BS_AUTORADIOBUTTON, 20, 20, 160, 25, hWnd, (HMENU)ID_R2, g_hInst, NULL);
```

BOOL CheckRadioButton(HWND hDlg, int nIDFirstButton, int nIDLastButton,

int IDCheckButton)

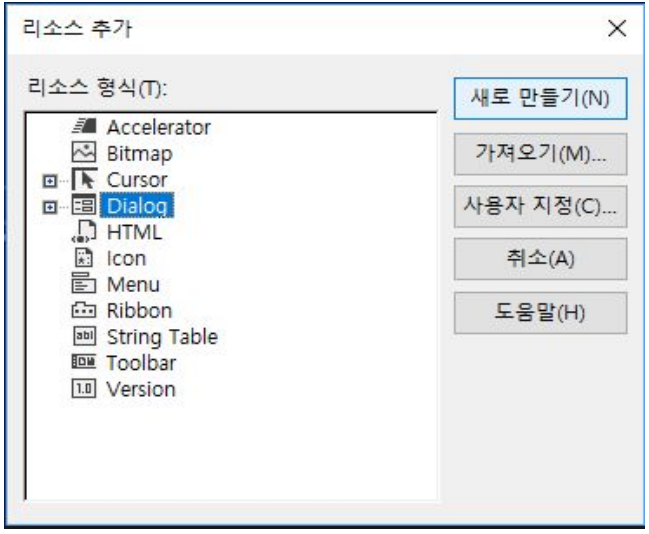
```
//CheckRadioButton
CheckRadioButton(hWnd, ID_R1, ID_R3, ID_R1);
CheckRadioButton(hWnd, ID_R4, ID_R6, ID_R4);
```

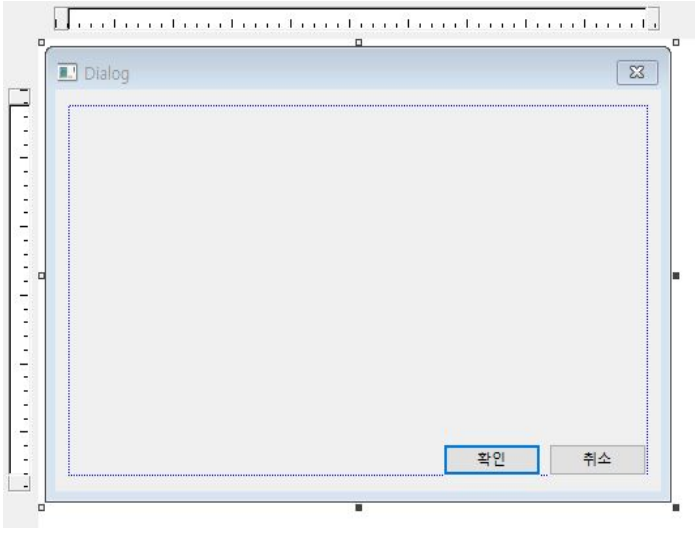
DialLogBox

여러가지 컨트롤러 들이 사용된다, 윈도우 안의 작은 윈도우

DiaLogBox 템플리스트, DiaLogBox 프록시저

*DiaLogBox 템플리스트





*DialogBox 생성

```
//DialogBox 생성(인스턴스, 리소스(템플리트), 다이얼로그가 뿌려질 윈도우, DialogBox 프록시져)
DialogBox(g_hInst, MAKEINTRESOURCE(IDD_DIALOG1), hWnd, AboutDlgProc);
```

DiaLogBox 프록시저

```
//다이얼로그 프록시저
BOOL CALLBACK AboutDlgProc(HWND hDlg, UINT iMessage, WPARAM wParam, LPARAM lParam);
```

```
 //DiallogBox 프록시저
BOOL CALLBACK AboutDlgProc(HWND hDlg, UINT iMessage, WPARAM wParam, LPARAM lParam)
{
    switch (iMessage)
    {
        //DiallogBox 진입시.
    case WM_INITDIALOG:
        return TRUE;
    case WM_COMMAND:
        switch (wParam)
        {
            //DiallogBox의 확인, 취소 버튼
        case IDOK:
        case IDCANCEL:
            EndDialog(hDlg, 0);
            return TRUE;
        }
        break;
    }
    return FALSE;
}
```

*DialLogBox프록시저

-윈도우 프록시저와 비슷한 역활을 하지만 지턴값이 다르다.

-WM_INITDIALOG 메시지 사용

-WM_COMMAND를 주로 사용한다.

-BOOL EndDialog(HWND hDlg, int nResult);