

7장 차일드 윈도우

2019년도 1학기 윈도우 프로그래밍

학습 목표

- **학습목표**

- 차일드 윈도우 만들기
- 버튼, 에디트 박스, 콤보 박스 등 컨트롤 윈도우를 활용할 수 있다.
- 윈도우 분할하기

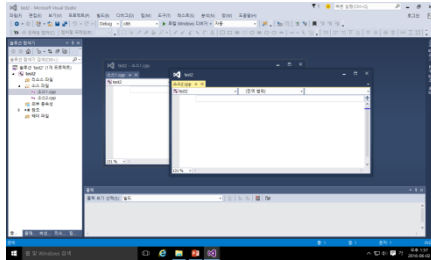
- **내용**

- 차일드 윈도우를 활용하여 컨트롤 윈도우 만들기
- 윈도우를 분할하여 차일드 윈도우로 사용하기

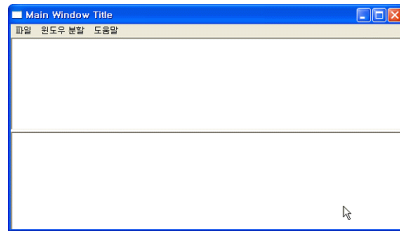
여러 개의 윈도우 만들기

- 1개 이상의 윈도우를 만드는 방법

- MDI (Multiple Document Interface): 여러 개의 문서를 여러 개의 화면에 출력하는 형태
 - 예) MS 워드, 엑셀, 비주얼 스튜디오 같은 형태



- 윈도우 분할: 기존의 윈도우를 여러 개의 자식 윈도우로 분할하는 형태



- 차일드 윈도우: 부모 윈도우 아래의 자식 윈도우를 생성하는 형태
 - 사용자가 만든 윈도우를 부모 윈도우로 두고, 차일드 윈도우를 만든다.

차일드 윈도우

• 차일드 윈도우는

- 기존의 윈도우를 만드는 방법으로 만든다. (CreateWindow, CreateWindowEx 함수)
- 차일드 윈도우는 각자의 윈도우 클래스를 가진다.
 - 1개 이상의 윈도우 클래스를 등록한다. (RegisterClass, RegisterClassEx 함수)
 - 차일드 윈도우 클래스는 클래스 이름으로 구분한다.
 - 각자의 윈도우 프로시저를 가질 수 있다.
- 차일드 윈도우 안에 또 다른 차일드 윈도우를 가질 수 있다.
- 윈도우 스타일을 **WS_CHILD | WS_VISIBLE** 형태로 설정한다.
 - WS_POPUP 스타일은 WS_CHILD와 같이 설정할 수 없다.
- 컨트롤들을 차일드 윈도우로 만들 수 있다.
- 부모 윈도우 가져오기
 - **HWND GetParent** (HWND hWnd);
- 부모 윈도우 변경하기
 - **HWND SetParent** (HWND hWndChild, HWND hWndNewParent);
- 소유 윈도우 가져오기
 - **HWND GetWindow** (HWND hWnd, UINT uCmd);
 - uCmd: GW_OWNDR / GW_CHILD...

차일드 윈도우 만들기

- 일반적인 윈도우를 차일드 윈도우로 만들기

- 차일드 윈도우 클래스 등록

```
WNDCLASSEX wc;
```

```
//윈도우 클래스를 등록한다.
```

```
wc.cbSize = sizeof(WNDCLASSEX);
```

```
wc.style = CS_HREDRAW | CS_VREDRAW;
```

```
wc.lpfnWndProc = WndProc;
```

```
wc.cbClsExtra = 0;
```

```
wc.cbWndExtra = 0;
```

```
wc.hInstance = hInstance;
```

```
wc.hIcon = LoadIcon(NULL, IDI_APPLICATION);
```

```
wc.hCursor = LoadCursor(NULL, IDC_ARROW);
```

```
wc.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);
```

```
wc.lpszMenuName = NULL;
```

```
wc.lpszClassName = "ParentClass";
```

```
wc.hIconSm = LoadIcon(NULL, IDI_APPLICATION);
```

```
RegisterClassEx(&wc);
```

```
// 차일드 윈도우 클래스를 등록한다.
```

```
wc.hCursor = LoadCursor(NULL, IDC_HELP);
```

```
wc.hbrBackground = (HBRUSH) GetStockObject (GRAY_BRUSH);
```

```
wc.lpszClassName = "ChildClass";
```

```
wc.lpfnWndProc = ChildProc;
```

```
RegisterClassEx(&wc);
```

```
// 클래스 스타일
```

```
// 윈도우 프로시저 지정
```

```
// 윈도우클래스 데이터 영역
```

```
// 윈도우의 데이터 영역
```

```
// 인스턴스 핸들
```

```
// 아이콘 핸들
```

```
// 사용할 커서 핸들
```

```
// 바탕색 브러쉬 핸들
```

```
// 메뉴 이름
```

```
// 윈도우 클래스 이름
```

```
// 윈도우 클래스를 등록
```

```
// 차일드 윈도우 클래스 이름
```

```
// 차일드 윈도우 프로시저 지정
```

```
// 자식 윈도우 클래스를 등록
```

차일드 윈도우 만들기

- 차일드 윈도우 만들기

```
LRESULT CALLBACK WndProc (HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
    HWND child_hWnd;
    switch (uMsg)
    {
        case WM_CREATE: //메인윈도우가 생성될 때 자식 윈도우 생성
            child_hWnd = CreateWindow ( " ChildClass ", NULL, WS_CHILD | WS_VISIBLE | WS_BORDER,
                                     10, 10, 200, 500, hWnd, NULL, g_hInst, NULL);

            break;
    }
    return DefWindowProc(hWnd, uMsg, wParam, lParam);
}
```

- 차일드 윈도우 프로시저 만들기

```
LRESULT CALLBACK ChildProc (HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
    switch (uMsg) {
        case WM_LBUTTONDOWN: // 마우스 좌측 버튼을 누른 경우
            MessageBox (hWnd, L"Left Mouse Button", L"Mouse Test ", MB_OK);
            break;
        case WM_DESTROY:
            PostQuitMessage(0);
            break;
    }
    return DefWindowProc(hWnd, uMsg, wParam, lParam);
}
```

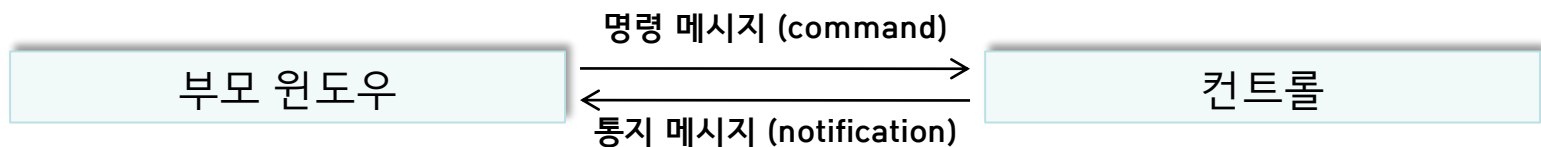
차일드 윈도우 만들기

- **차일드 윈도우 클래스**: 등록된 윈도우 클래스로 차일드 윈도우 클래스를 설정
- **차일드 윈도우 스타일**: **WS_CHILD | WS_VISIBLE** 스타일을 기본으로 설정
 - 위의 두 스타일 외에, WS_BORDER나 WS_THICKFRAME 등의 스타일을 같이 설정할 수 있다.
 - WS_CHILD 스타일과 WS_POPUP 스타일을 같이 사용할 수 없다.
- 또는 확장된 윈도우 스타일을 설정할 수 있는 **CreateWindowEx** 함수를 사용한다.
- HWND **CreateWindowEx** (**DWORD dwExStyle**, LPCTSTR lpClassName, LPCTSTR lpWindowName, DWORD dwStyle, int x, int y, int nWidth, int nHeight, HWND hWndParent, HMENU hMenu, HANDLE hInstance, LPVOID lpParam);
 - dwExStyle
 - WS_EX_CLIENTEDGE: 작업영역이 썩 들어간 음각 모양으로 만든다.
 - WS_EX_WINDOWEDGE: 양각 모양의 경계선을 가진 윈도우를 만든다.
 - WS_EX_DLGMODALFRAME: 이중 경계선을 가진 윈도우를 만든다.
- 윈도우 스타일
 - WS_OVERLAPPEDWINDOW: WS_CAPTION / WS_HSCROLL / WS_VSCROLL / WS_SYSMENU / WS_MAXIMIZEBOX / WS_MINIMIZEBOX / WS_THICKFRAME / WS_BORDER
 - WS_OVERLAPPEDWINDOW: 기존의 윈도우와 같은 형태로 생성
 - WS_THICKFRAME: 크기를 바꿀 수 있다
 - WS_BORDER: 테두리만 있고 크기와 위치는 바꿀 수 없다
 - WS_POPUPWINDOW: WS_POPUP | WS_BORDER | WS_SYSMENU
 - 팝업 스타일의 차일드 윈도우: WS_POPUPWINDOW | WS_VISIBLE | WS_CAPTION

컨트롤 차일드 윈도우

- 대화상자에서 사용했던 컨트롤:
 - 윈도우로 부모 윈도우 아래의 자식 윈도우로 존재한다.
- 컨트롤 차일드 윈도우 클래스 이름:
 - 버튼 컨트롤 윈도우 클래스: "button"
 - 에디트 컨트롤 윈도우 클래스: "edit"
 - 콤보박스 윈도우 클래스: "combobox"
 - 리스트박스 윈도우 클래스: "listbox"
- 발생 메시지: WM_COMMAND 메시지

컨트롤	윈도우 클래스 이름	스타일	명령 메시지	통지 메시지
버튼	button	BS_	BM_	BN_
리스트 박스	listbox	LBS_	LB_	LBN_
콤보 박스	combobox	CBS_	CB_	CBN_
에디트	edit	ES_	EM_	EN_



컨트롤 차일드 윈도우

- 기본적으로 `CreateWindow` 함수로 자식 윈도우를 만든다.
 - `HWND CreateWindow (LPCTSTR lpClassName, LPCTSTR lpWindowName, DWORD dwStyle, int x, int y, int nWidth, int nHeight, HWND hWndParent, HMENU hMenu, HANDLE hInstance, PVOID lpParam);`
- **lpClassName**: 차일드 윈도우 클래스 이름
 - 예) 버튼 클래스: `button`
 - 예) 에디트 박스 클래스: `edit`
 - 예) 리스트 박스: `listbox`
- **lpWindowName**: 윈도우 캡션
- **dwStyle**: 윈도우 스타일
 - 윈도우 스타일 | 차일드 스타일: `WS_CHILD | WS_VISIBLE` 항상 포함
 - 예) `WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON`
- **x, y, nWidth, nHeight**: 윈도우 위치 (x, y), 크기 (폭, 높이)
- **nWndParent**: 부모 윈도우 핸들
- **hMenu**: 차일드 윈도우 아이디
- **hInstance**: 인스턴스 핸들

컨트롤 윈도우 활용하기

- 컨트롤 윈도우의 텍스트를 읽어온다.
 - `int GetWindowText (HWND hWnd, LPTSTR lpString, int nMaxCount);`
 - hWnd: 컨트롤 윈도우 핸들
 - lpString: 텍스트가 저장될 버퍼
 - nMaxCount: 텍스트의 길이
- 컨트롤 윈도우의 텍스트를 변경한다.
 - `BOOL SetWindowText (HWND hWnd, LPCTSTR lpString);`
 - hWnd: 컨트롤 윈도우 핸들
 - lpString: 텍스트

버튼 컨트롤

- 버튼의 종류

윈도우 클래스 이름	윈도우 스타일	버튼 내용
button	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_AUTORADIOBUTTON	자동 라디오 버튼
	BS_GROUPBOX	그룹 박스

버튼 만들기

- 버튼 만들기

```
#define IDC_BUTTON 100
```

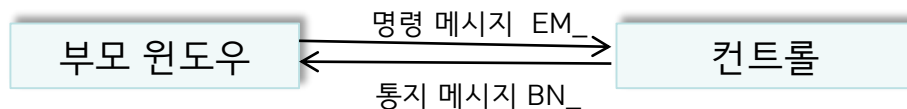
```
HWND hButton;
```

```
hButton = CreateWindow ("button", "확인",  
    WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON, 200, 0, 100, 25,  
    hwnd, (HMENU) IDC_BUTTON, g_hInst, NULL);
```

- 버튼 클래스: **button**
- 윈도우 스타일: **WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON**
- 버튼 이벤트 (버튼 통지)

Notify	Meaning
BN_CLICKED	버튼 위에서 마우스가 클릭되었을 때
BN_DBLCLK	버튼 위에서 마우스가 더블 클릭되었을 때
BN_SETFOCUS	버튼 위 마우스 커서가 올 때
BN_KILLFOCUS	버튼 위에서 마우스가 벗어날 때
BN_PAINT	버튼 내부를 Drawing 할 때

- WM_COMMAND 메시지
 - HIWORD(wParam): 통지 코드
 - LOWORD(wParam): 컨트롤의 ID

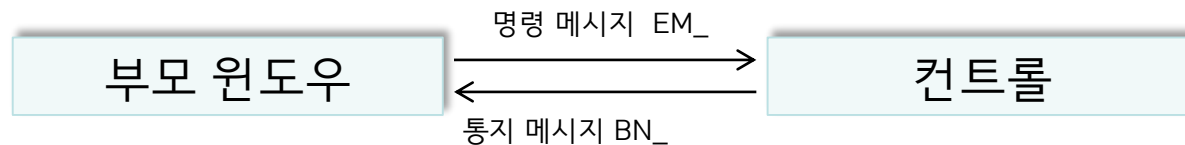


버튼 만들기

- 버튼 클래스: button
- 윈도우 스타일: WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON
- 버튼 이벤트 (버튼 통지)

Notify	Meaning
BN_CLICKED	버튼 위에서 마우스가 클릭되었을 때
BN_DBCLK	버튼 위에서 마우스가 더블 클릭되었을 때
BN_SETFOCUS	버튼 위 마우스 커서가 올 때
BN_KILLFOCUS	버튼 위에서 마우스가 벗어날 때
BN_PAINT	버튼 내부를 Drawing 할 때

- WM_COMMAND 메시지
 - HIWORD(wParam): 통지 코드
 - LOWORD(wParam): 컨트롤의 ID



버튼 만들기

```
#define IDC_BUTTON 100 // 버튼 컨트롤의 ID
```

```
LRESULT CALLBACK WndProc (HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)
```

```
{
```

```
    HDC hdc;
```

```
    static HWND hButton;
```

```
    switch (iMsg)
```

```
    {
```

```
        case WM_CREATE:
```

```
            hButton = CreateWindow ( "button", "확인",  
                                   WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON,  
                                   200, 0, 100, 25, hwnd,  
                                   (HMENU) IDC_BUTTON, hInst, NULL);
```

```
// 버튼의 윈도우 클래스 이름은 "button"
```

```
// 차일드 윈도우이고 누르는 형태의 버튼 스타일
```

```
        break;
```

```
        case WM_COMMAND:
```

```
            switch(LOWORD(wParam)) {
```

```
                case IDC_BUTTON:
```

```
                    hdc = GetDC(hwnd);
```

```
                    TextOut (hdc, 0, 100, "Hello World", 11);
```

```
                    ReleaseDC (hwnd, hdc);
```

```
                    break;
```

```
            }
```

```
        break;
```

```
    }
```

```
}
```

체크박스 만들기

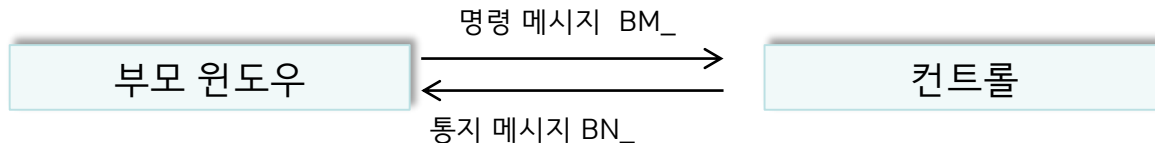
- 체크박스 만들기

```
#define IDC_CHECK 200
```

```
HWND hCheck;
```

```
hCheck = CreateWindow ("button", "체크 박스 테스트",  
    WS_CHILD | WS_VISIBLE | BS_CHECKBOX, 100, 0, 100, 25,  
    hwnd, (HMENU) IDC_CHECK, hInst, NULL);
```

- 체크 박스 클래스: **button**
- 체크 박스 스타일: **WS_CHILD | WS_VISIBLE | BS_AUTOCHECKBOX**
 - **BS_CHECKBOX 스타일**: 체크 상태를 수동으로 바꿔준다.
 - **BS_AUTOCHECKBOX 스타일**: 체크 상태가 자동으로 바뀌어진다.
- 체크박스 통지 메시지:
 - 차일드 윈도우 → 부모 윈도우: **BN_CLICKED** 메시지를 보낸다.



체크박스 만들기

- 부모 윈도우가 체크 박스의 현재 상태를 알아보거나 상태를 바꾸고자 할 때 차일드 윈도우로 메시지를 보낸다.
 - SendMessage 함수를 이용하여 차일드 윈도우로 메시지를 보낸다.
 - 보내는 메시지: **BM_GETCHECK** / **BM_SETCHECK**

메시지	의미	리턴 값 또는 체크 박스 상태
BM_GETCHECK	체크박스가 현재 체크되어 있는 상태인지 조사	<ul style="list-style-type: none">• BST_CHECKED: 현재 체크되어 있다.• BST_UNCHECKED: 현재 체크되어 있지 않다.• BST_INDETERMINATE: 체크도 아니고 비체크도 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

체크박스 만들기

- 체크박스 버튼 컨트롤 (수동)

```
#define IDC_BUTTON5 100
```

```
LRESULT CALLBACK WndProc ( HWND hWnd, UINT iMsg, WPARAM wParam, LPARAM lParam)
```

```
{  
static HWND hCheck;  
Static int cList[2];
```

```
switch (iMsg) {
```

```
case WM_CREATE:
```

```
    hCheck = CreateWindow ( "button", "Grid",  
                           WS_CHILD | WS_VISIBLE | BS_CHECKBOX,  
                           10, 210, 180, 40, hwnd, (HMENU) IDC_BUTTON5,  
                           hInst, NULL);
```

```
// 윈도우 클래스 이름은 button  
// 차일드 윈도우, 수동 체크 박스
```

```
    break;
```

```
case WM_COMMAND:
```

```
    switch (LOWORD(wParam) ) {
```

```
    case IDC_BUTTON5: // grid check box
```

```
        if (SendMessage (hCheck, BM_GETCHECK, 0, 0) == BST_UNCHECKED) {
```

```
            SendMessage (hCheck, BM_SETCHECK, BST_CHECKED, 0);
```

```
            cList[0] = 1;
```

```
        }
```

```
        else {
```

```
            SendMessage (hCheck, BM_SETCHECK, BST_UNCHECKED, 0);
```

```
            cList[0] = 0;
```

```
        }
```

```
        break;
```

```
    }
```

```
}
```

```
return DefWindowProc (hWnd, iMsg, wParam, lParam);
```

```
}
```

체크박스 만들기

- 체크박스 버튼 컨트롤 (자동)

```
#define IDC_BUTTON5 100
```

```
LRESULT CALLBACK 쭈예갯 ( HWND hWnd, UINT iMsg, WPARAM wParam, LPARAM lParam)
```

```
{  
    Static HWND hCheck;  
    Static int cList[2];
```

```
    Switch (iMsg) {
```

```
        case WM_CREATE:
```

```
            hCheck = CreateWindow ( " button ", " Grid ",  
                                   WS_CHILD | WS_VISIBLE | BS_AUTOCHECKBOX,  
                                   10, 210, 180, 40, hWnd, (HMENU) IDC_BUTTON5,  
                                   hInst, NULL);
```

```
// 윈도우 클래스 이름은 button  
// 차일드 윈도우, 자동 체크 박스
```

```
            break;
```

```
        case WM_COMMAND:
```

```
            switch (LOWORD(wParam) ) {
```

```
                case IDC_BUTTON5: // grid check box: 자동 체크박스로 만들어 체크박스의 상태만 조사한다.
```

```
                    if (SendMessage (hCheck, BM_GETCHECK, 0, 0) == BST_UNCHECKED) {
```

```
                        cList[0] = 1;
```

```
                    }
```

```
                    else {
```

```
                        cList[0] = 0;
```

```
                    }
```

```
                    break;
```

```
            }
```

```
        }
```

```
    return DefWindowProc (hWnd, iMsg, wParam, lParam);
```

```
}
```

라디오 버튼 만들기

- 라디오 버튼 만들기

```
#define IDC_RADIO 300
```

```
HWND hRadio;
```

```
hRadio = CreateWindow ( "button", "라디오 버튼 테스트",  
    WS_CHILD | WS_VISIBLE | BS_AUTORADIOBUTTON,  
    100, 0, 100, 30, hWnd, (HMENU) IDC_RADIO, g_hInst, NULL);
```

- 라디오 버튼 클래스: **button**
- 라디오 버튼 스타일: **WS_CHILD | WS_VISIBLE | BS_AUTORADIOBUTTON**
 - 그룹의 시작을 위하여 첫번째 라디오 버튼에 **WS_GROUP** 스타일 추가한다.
- 라디오 버튼 그룹 박스 만들기
 - 그룹 박스 클래스: **button**
 - 그룹 박스 컨트롤 스타일: **BS_GROUPBOX**

라디오 버튼 만들기

- 라디오 버튼 초기화 함수

```
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 : 각각 그룹의 시작 버튼 아이디
- nIDLastButton: 각각 그룹의 끝 버튼 아이디
- nIDCheckButton: 선택될 버튼의 아이디

라디오 버튼 만들기

```
#define ID_R1    100
#define ID_R2    200
#define ID_R3    300
```

```
LRESULT CALLBACK WndProc ( HWND hWnd, UINT iMsg, WPARAM wParam, LPARAM lParam)
```

```
{
```

```
    HWND r1, r2, r3;
```

```
    Static int shape;
```

```
    switch (iMsg) {
```

```
    case WM_CREATE:
```

```
        // 그룹 박스로 윈도우 만들기
```

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

```
        // 버튼 만들기: 그룹 1
```

```
        r1= CreateWindow ("button", "Rectangle", WS_CHILD | WS_VISIBLE | BS_AUTORADIOBUTTON | WS_GROUP,
                          10, 20, 100, 30, hWnd, (HMENU) ID_R1, g_hInst, NULL);
```

```
        r2= CreateWindow ("button", "Ellipse", WS_CHILD | WS_VISIBLE | BS_AUTORADIOBUTTON,
                          10, 50, 100, 30, hWnd, (HMENU) ID_R2, g_hInst, NULL);
```

```
        r3= CreateWindow ("button", "Line", WS_CHILD | WS_VISIBLE | BS_AUTORADIOBUTTON,
                          10, 80, 100, 30, hWnd, (HMENU) ID_R3, g_hInst, NULL);
```

```
        CheckRadioButton (hWnd, ID_R1, ID_R3, ID_R1);
```

```
        break;
```

```
    case WM_COMMAND:
```

```
        switch (LOWORD (wParam)) {
```

```
            case ID_R1:  shape = 1;    break;
```

```
            case ID_R2:  shape = 2;    break;
```

```
            case ID_R3:  shape = 3;    break;
```

```
        }
```

```
        break;
```

```
    }
```

```
}
```

에디트 컨트롤 만들기

- 에디트 컨트롤 만들기

```
#define IDC_EDIT    101
```

```
HWND hEdit;
```

```
hEdit = CreateWindow ( "edit", "에디팅",  
                      WS_CHILD | WS_VISIBLE | WS_BORDER,  
                      0, 0, 200, 25, hwnd, (HMENU) IDC_EDIT, hInst, NULL);
```

- 에디트 컨트롤 클래스: **edit**
- 에디트 컨트롤 스타일: **WS_CHILD | WS_VISIBLE | WS_BORDER**

에디트 컨트롤 만들기

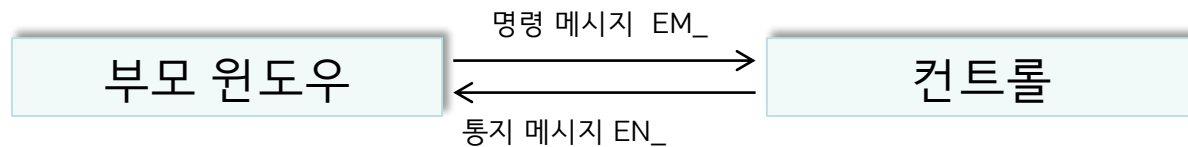
•에디트 컨트롤 윈도우 스타일

클래스 이름	스타일	의미
edit	ES_AUTOHSCROLL	수평 스크롤을 지원
	ES_AUTOVSCROLL	여러 줄을 편집할 때 수직 스크롤을 지원
	ES_LEFT	왼쪽 정렬
	ES_RIGHT	오른쪽 정렬
	ES_CENTER	중앙 정렬
	ES_LOWERCASE	소문자로 변환하여 표시
	ES_UPPERCASE	대문자로 변환하여 표시
	ES_MULTILINE	여러 줄을 편집
	ES_READONLY	읽기 전용, 편집할 수 없다.
	ES_PASSWORD	입력되는 모든 문자를 *로 보여준다.

에디트 컨트롤 만들기

•에디트 컨트롤 메시지 통지

메시지	의미
EN_CHANGE	Editbox의 내용이 변경된 후 발생 (화면에 갱신된 후)
EN_UPDATE	Editbox 내용이 변경되려고 할 때 발생 (사용자가 타이프한 후 화면에 갱신되기 직전에 발생)
EN_SETFOCUS	포커스를 받을 때 발생
EN_KILLFOCUS	포커스를 잃을 때 발생
EN_HSCROLL/EN_VSCROLL	수평 / 수직 스크롤바 클릭
EN_MAXTEXT	지정한 문자열 길이를 초과
EN_ERRSPACE	메모리 부족



에디트 컨트롤 만들기

```
#define IDC_BUTTON 100
#define IDC_EDIT 101 // 에디트 컨트롤의 ID

HWND hButton, hEdit;
char str[100];

LRESULT CALLBACK WndProc (HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)
{
    case WM_CREATE:
        // 박스 주위에 테두리가 있는 에디트 컨트롤스타일
        hEdit = CreateWindow ("edit", "에디팅", WS_CHILD | WS_VISIBLE | WS_BORDER,
                               0, 0, 200, 25, hwnd, (HMENU) IDC_EDIT, hInst, NULL);

        break;

    case WM_COMMAND:
        switch(LOWORD(wParam)) {
            case IDC_BUTTON:
                GetDlgItemText(hwnd, IDC_EDIT, str, 100);
                hdc = GetDC (hwnd);
                TextOut (hdc, 0, 100, str, strlen(str));
                ReleaseDC(hwnd, hdc);

                break;
        }
        break;
    }
    return DefWindowProc (hwnd, iMsg, wParam, lParam);
}
```

콤보 박스 만들기

- 콤보박스 클래스: **combobox**
- 콤보 박스 스타일

스타일	의미
CBS_SIMPLE	에디트만 가진다.
CBS_DROPDOWN	에디트와 리스트 박스를 가진다.
CBS_DROPDOWNLIST	리스트 박스만 가지며 에디트에 항목을 입력할 수는 없다
CBS_AUTOHSCROLL	콤보 박스에서 항목을 입력할 때 자동 스크롤

- 콤보박스의 다운 버튼을 눌렀을 때
 - CBN_DROPDOWN 통지가 보내진다.

메시지	의미
CBN_DBLCLK	콤보 박스를 더블클릭하였다.
CBN_ERRSPACE	메모리가 부족하다.
CBN_KILLFOCUS	키보드 포커스를 잃었다.
CBN_SELCANCEL	사용자가 선택을 취소하였다.
CBN_SELCHANGE	사용자에 의해 선택이 변경되었다.
CBN_SETFOCUS	키보드 포커스를 얻었다.

콤보 박스에 전달되는 메시지

- 부모 윈도우가 콤보 박스에 보내는 메시지

메시지	의미	전달 값
CB_ADDSTRING	콤보 박스에 텍스트를 아이템으로 추가하는 메시지로써 리스트의 마지막에 추가된다.	wParam: 사용하지 않음 lParam: 텍스트 스트링의 시작 주소
CB_DELETESTRING	콤보 박스에 있는 아이템들 중 하나를 삭제하는 메시지	wParam: 삭제하기 원하는 아이템의 인덱스로 0부터 시작한다. lParam: 0
CB_GETCOUNT	콤보 박스의 아이템 리스트에 들어 있는 아이템의 개수를 얻기 위한 메시지로 개수 값은 SendMessage()함수가 리턴한다.	wParam: 0 lParam: 0
CB_GETCURRESEL	현재 선택된 아이템의 인덱스 번호를 얻기 위한 메시지로 인덱스 번호는 SendMessage()함수가 리턴한다.	wParam: 0 lParam: 0
CB_SETCURRESEL	콤보 박스 컨트롤의 텍스트 편집 공간에 지정 한 항목의 텍스트를 보여준다.	wParam: 나타내고자 하는 항목의 인덱스 번호 lParam: 사용않음

콤보 박스 만들기

```
#define IDC_BUTTON 100
#define IDC_EDIT 101
#define IDC_COMBO 102           // 콤보 박스 컨트롤의 ID

HWND hButton, hEdit, hCombo;
char str[100];

LRESULT CALLBACK WndProc (HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)
{
    switch (iMsg)
    {
        case WM_CREATE:
            hCombo = CreateWindow ("combobox", NULL, WS_CHILD | WS_VISIBLE | CBS_DROPDOWN, // 콤보 박스
                                   0, 100, 200, 300, hwnd, (HMENU) IDC_COMBO, hInst, NULL);

            return 0;
        case WM_COMMAND:
            switch(LOWORD(wParam)) {
                case ID_COMBOBOX:
                    switch (HIWORD(wParam)) {
                        case CBN_SELCHANGE:
                            i= SendMessage (hCombo, CB_GETCURSEL,0,0);
                            SendMessage (hCombo, CB_GETLBTEXT, i, (LPARAM)str);
                            SetWindowText (hWnd, str);

                            break;
                    }
                    break;
                case IDC_BUTTON:
                    GetDlgItemText (hwnd, IDC_EDIT, str, 100);
                    if (strcmp(str, ""))
                        SendMessage (hCombo,CB_ADDSTRING,0,(LPARAM)str);

                    break;
            }
        }
    }
    break;
}
```

리스트 박스 만들기

- 리스트 박스 클래스 이름: **listbox**
- 리스트 박스 스타일

스타일	의미
LBS_MULTIPLESEL	여러개의 항목을 선택할 수 있도록 한다. 이 스타일을 적용하지 않으면 디폴트로 하나만 선택할 수 있다.
LBS_NOTIFY	사용자가 목록중 하나를 선택했을 때 부모 윈도우로 통지 메시지를 보내도록 한다.
LBS_SORT	추가된 항목들을 자동 정렬하도록 한다.
LBS_OWNERDRAW	문자열이 아닌 비트맵이나 그림을 넣을 수 있도록 한다
LBS_STANDARD	LBS_NOTIFY LBS_SORT WS_BORDER (가장 일반적인 스타일)

- 리스트 박스에서 메시지가 발생했을 때 부모 윈도우로 보내는 통지 메시지

메시지	의미
LBN_DBLCLK	리스트 박스를 더블클릭하였다.
LBN_ERRSPACE	메모리가 부족하다.
LBN_KILLFOCUS	키보드 포커스를 잃었다.
LBN_SELCANCEL	사용자가 선택을 취소하였다.
LBN_SELCHANGE	사용자에 의해 선택이 변경되었다.
LBN_SETFOCUS	키보드 포커스를 얻었다.

리스트 박스에 전달되는 메시지

- 부모 윈도우가 리스트 박스에게 보내는 메시지

메시지	의미	전달 값
LB_ADDSTRING	리스트 박스에 텍스트를 아이템으로 추가하는 메시지로써 리스트의 마지막에 추가된다.	wParam: 사용하지 않음 lParam: 텍스트 스트링의 시작 주소
LB_DELETESTRING	리스트 박스에 있는 아이템들 중 하나를 삭제하는 메시지	wParam: 삭제하기 원하는 아이템의 인덱스로 0부터 시작한다. lParam: 0
LB_GETCOUNT	리스트 박스의 아이템 리스트에 들어 있는 아이템의 개수를 얻기 위한 메시지로 개수 값은 SendMessage()함수가 리턴한다.	wParam: 0 lParam: 0
LB_GETCURSEL	현재 선택된 아이템의 인덱스 번호를 얻기 위한 메시지로 인덱스 번호는 SendMessage()함수가 리턴한다.	wParam: 0 lParam: 0
LB_GETTEXT	아이템 리스트중 wParam에서 지정한 인덱스 아이템의 텍스트를 얻어오는 메시지	wParam: 얻어올 아이템의 인덱스 번호 lParam: 얻어온 텍스트를 저장할 버퍼의 시작 주소
LB_INSERTSTRING	리스트 박스에 텍스트를 아이템으로 리스트 중간에 추가하는 메시지	wParam: 아이템 리스트중 추가될 위치의 인덱스 번호 lParam: 텍스트 스트링의 시작 주소

리스트 박스 만들기

```
#define ID_LISTBOX 100
char Items[][15]={"First", "Second", "Third", "Fourth"};
char str[128];
HWND hList;

LRESULT CALLBACK WndProc(HWND hWnd,UINT iMessage,WPARAM wParam,LPARAM lParam)
{
    int i;
    switch(iMessage) {
        case WM_CREATE:
            hList=CreateWindow ("listbox", NULL, WS_CHILD | WS_VISIBLE | WS_BORDER | LBS_STANDARD ,
                               10, 10, 100, 200, hWnd, (HMENU) ID_LISTBOX, g_hInst, NULL);
            for ( i=0; i<4; i++)
                SendMessage (hList, LB_ADDSTRING, 0, (LPARAM)Items[i]);
            return 0;

        case WM_COMMAND:
            switch (LOWORD(wParam))
            {
                case ID_LISTBOX:
                    switch (HIWORD(wParam))
                    {
                        case LBN_SELCHANGE:
                            i=SendMessage (hList, LB_GETCURSEL,0,0);
                            SendMessage (hList, LB_GETTEXT, i, (LPARAM)str);
                            SetWindowText (hWnd, str);
                            break;
                    }
                }
            } return 0;
    }
    return (DefWindowProc (hWnd, iMessage, wParam, lParam));
}
```

스크롤 바 만들기

- 스크롤 바 클래스 이름: **scrollbar**
- 스크롤 바 스타일:
 - 수평 스크롤바: **SBS_HORZ**
 - 수직 스크롤바: **SBS_VERT**
- 스크롤 바의 범위를 지정

```
BOOL SetScrollRange( HWND hWnd, int nBar, int nMinPos,  
                    int nMaxPos, BOOL bRedraw );
```

- 스크롤 바의 최대값(nMaxPos), 최소값(nMinPos)을 지정
- hWnd: 스크롤 바의 윈도우 핸들
- nBar: 메인 윈도우에 부착된 스크롤 바 또는 별도의 스크롤 바 컨트롤을 지정 (이 값이 SBS_CTL이면 별도의 컨트롤을 지정한다.)
 - SB_CTL: 스크롤 바 컨트롤 지정
 - SB_HORZ: 일반적인 수평 스크롤바 지정
 - SB_VERT: 일반적인 수직 스크롤바 지정
- nMaxPos: 스크롤 바의 최대 위치
- bRedraw: 화면의 값이 변하면 스크롤 바를 다시 그릴지를 결정

스크롤 바 만들기

- 스크롤 바의 현재 값을 지정

```
int SetScrollPos ( HWND hWnd, int nBar, int nPos, BOOL bRedraw );
```

스크롤 바의 현재 값 지정

- hWnd: 스크롤바 컨트롤
- nBar: 위치를 설정할 스크롤 바
 - SB_CTL: 별도의 스크롤 바 컨트롤
 - SB_HORZ: 일반적인 수평 스크롤바
 - SB_VERT: 일반적인 수직 스크롤바
- nPos: 스크롤 바의 새로운 위치
- bRedraw: 위치를 설정한 후 다시 그릴 지

- 스크롤 바에서 발생 메시지

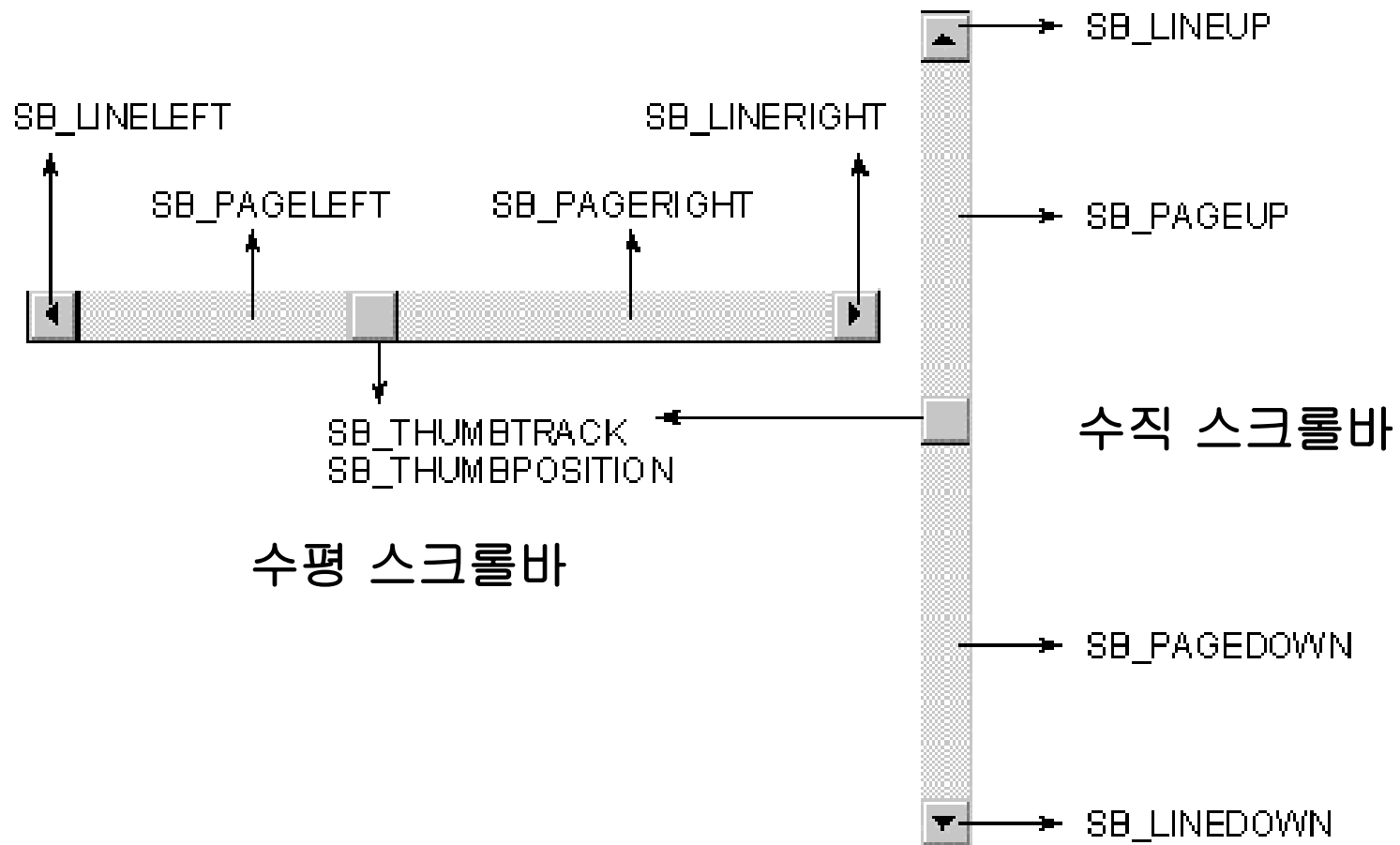
메시지	발생 컨트롤	인자
WM_HSCROLL	수평 스크롤 바	LOWORD(wParam): 스크롤 바 내의 어디를 눌렀는가? HIWORD (wParam): 현재 위치 lParam: 스크롤 바의 윈도우 핸들
WM_VSCROLL	수직 스크롤 바	

스크롤 바 만들기

- LOWORD(wParam) 의 가능한 값

값	설명
SB_LINELEFT 또는 SB_LINEUP	사용자가 왼쪽 화살표 버튼을 눌렀다는 뜻이며 이때는 왼쪽으로 한 단위 스크롤 시킨다.
SB_LINERIGHT 또는 SB_LINEDOWN	사용자가 오른쪽 화살표 버튼을 눌렀다는 뜻이며 이때는 오른쪽으로 한 단위 스크롤 시킨다.
SB_PAGELEFT 또는 SB_PAGEUP	사용자가 왼쪽 몸통 부분을 눌렀다는 뜻이며 이때는 한 페이지 왼쪽으로 스크롤 시킨다.
SB_PAGERIGHT 또는 SB_PAGEDOWN	사용자가 오른쪽 몸통 부분을 눌렀다는 뜻이며 이때는 한 페이지 오른쪽으로 스크롤 시킨다.
SB_THUMBPOSITION	박스를 드래그한 후 마우스 버튼을 놓았다.
SB_THUMBTRACK	스크롤 박스를 드래그하고 있는 중이다. 이 메시지는 마우스 버튼을 놓을 때 까지 계속 전달된다.

스크롤 바 만들기



스크롤 바 만들기

```
#define ID_SCRRED 100
#define ID_SRCGREEN 101
#define ID_SCRBLUE 102
```

```
LRESULT CALLBACK WndProc (HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)
{
```

```
    static HWND hRed,hGreen,hBlue;
```

```
    static int Red,Green,Blue;
```

```
    HDC hdc;
```

```
    PAINTSTRUCT ps;
```

```
    HBRUSH MyBrush, OldBrush;
```

```
    int TempPos;
```

```
    switch(iMessage) {
```

```
        case WM_CREATE:
```

```
            hRed = CreateWindow ("scrollbar", NULL, WS_CHILD | WS_VISIBLE | SBS_HORZ,
                                10,10,200,20, hwnd, (HMENU)ID_SCRRED, g_hInst, NULL);
```

```
            hGreen= CreateWindow ("scrollbar" , NULL,WS_CHILD | WS_VISIBLE | SBS_HORZ,
                                10,40,200,20, hwnd, (HMENU)ID_SRCGREEN, g_hInst, NULL);
```

```
            hBlue= CreateWindow ("scrollbar", NULL, WS_CHILD | WS_VISIBLE | SBS_HORZ,
                                10,70,200,20, hwnd, (HMENU)ID_SCRBLUE, g_hInst, NULL);
```

```
            SetScrollRange(hRed,SB_CTL,0,255,TRUE);
```

```
            SetScrollPos(hRed,SB_CTL,0,TRUE);
```

```
            SetScrollRange(hGreen,SB_CTL,0,255,TRUE);
```

```
            SetScrollPos(hGreen,SB_CTL,0,TRUE);
```

```
            SetScrollRange(hBlue,SB_CTL,0,255,TRUE);
```

```
            SetScrollPos(hBlue,SB_CTL,0,TRUE);
```

```
            Red = Green = Blue = 0;
```

```
        break;
```

스크롤 바 만들기

case WM_HSCROLL:

```
if ((HWND)IParam == hRed)      TempPos = Red;
if ((HWND)IParam == hGreen)    TempPos = Green;
if ((HWND)IParam == hBlue)     TempPos = Blue;

switch (LOWORD(wParam)) {
    case SB_LINELEFT:          TempPos=max(0,TempPos-1);      break;
    case SB_LINERIGHT:         TempPos=min(255,TempPos+1);    break;
    case SB_PAGELEFT:          TempPos=max(0,TempPos-10);     break;
    case SB_PAGERIGHT:         TempPos=min(255,TempPos+10);   break;
    case SB_THUMBTRACK:         TempPos=HIWORD(wParam);        break;
}
```

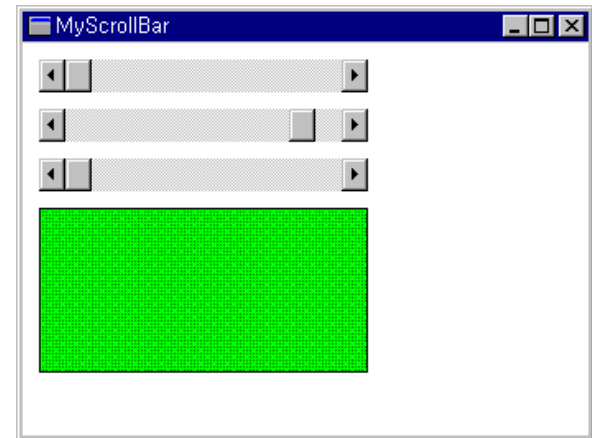
```
if ((HWND)IParam == hRed)      Red=TempPos;
if ((HWND)IParam == hGreen)    Green=TempPos;
if ((HWND)IParam == hBlue)     Blue=TempPos;
```

```
SetScrollPos((HWND)IParam, SB_CTL,TempPos,TRUE);
InvalidateRect (hWnd,NULL,true);
```

break;

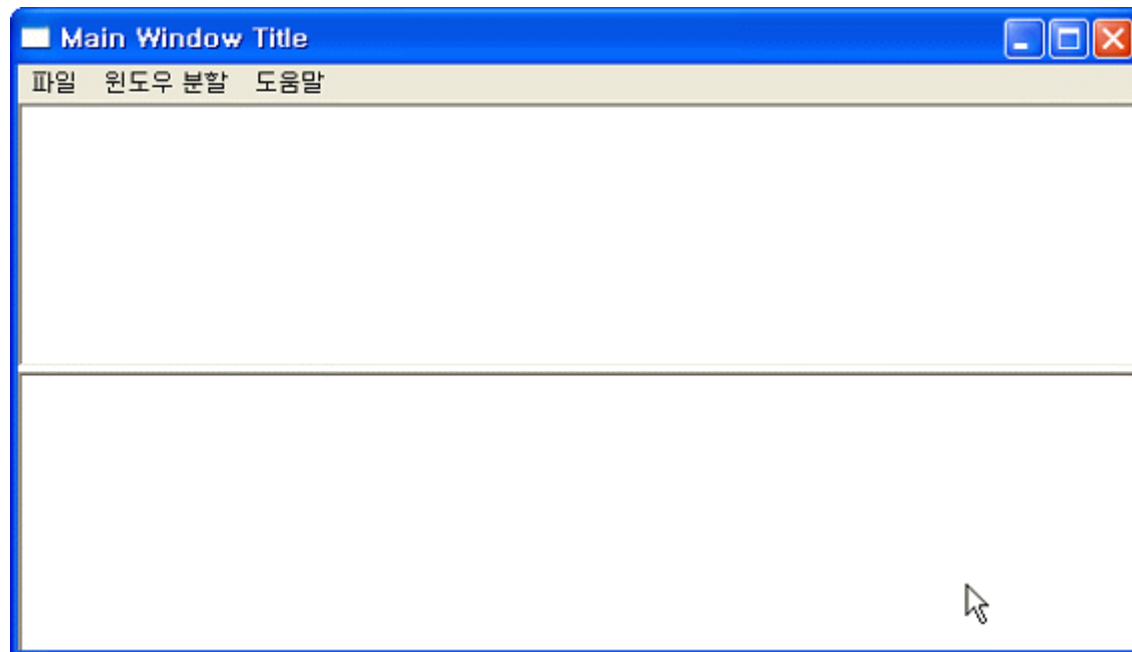
case WM_PAINT:

```
hdc=BeginPaint(hWnd,&ps);
MyBrush=CreateSolidBrush(RGB(Red,Green,Blue));
OldBrush=(HBRUSH)SelectObject(hdc,MyBrush);
Rectangle(hdc,10,100,210,200);
SelectObject(hdc,OldBrush);
DeleteObject(MyBrush);
EndPaint(hWnd,&ps);
break;
```



윈도우 분할하기

- 프레임 윈도우를 분할하여 차일드 윈도우 관리
 - 분할된 윈도우는 자식 윈도우이지만 팝업 윈도우는 아니므로 자식 윈도우에 타이틀 바를 포함하는 독립적인 프레임이 존재하지는 않는다.
 - 분할 윈도우도 메인 윈도우와 같은 방법으로 생성하고, **CreateWindowEx** 함수를 사용한다. (CreateWindowEx 함수를 사용하면 윈도우 가장자리의 스타일을 설정할 수 있다.)



윈도우 분할: 차일드 윈도우 생성 함수

- 원도우 생성 함수

HWND CreateWindowEx (

DWORD dwExStyle,	// 생성되는 확장 윈도우의 스타일
LPCTSTR lpClassName,	// 등록된 윈도우클래스
LPCTSTR lpWindowName,	// 윈도우 타이틀 텍스트
DWORD dwStyle,	// 기본 윈도우 스타일
int x,	// 생성 윈도우 위치의 x값
int y,	// 생성 윈도우 위치의 y값
int nWidth,	// 생성 윈도우의 너비
int nHeight,	// 생성 윈도우의 높이
HWND hWndParent,	// 부모 윈도우 핸들
HMENU hMenu,	// 사용될 메뉴의 핸들
HINSTANCE hInstance,	// 어플리케이션 인스턴스
LPVOID lpParam	

);

- dwExStyle 스타일:

스타일	내용
WS_EX_DLGMODALFRAME	이중 경계선을 가진 윈도우를 만든다
WS_EX_WINDOWEDGE	양각 모양의 경계선을 가진 윈도우를 만든다.
WS_EX_CLIENTEDGE	작업영역이 썩 들어간 음각 모양으로 만든다.
WS_EX_MDICHILD	MDI 차일드 윈도우를 만든다.
WS_EX_OVERLAPPEDWINDOW	(WS_EX_WINDOWEDGE WS_EX_CLIENTEDGE)복합 속성

윈도우 분할: 차일드 윈도우 클래스 등록

- 원도우 클래스 등록

```
WNDCLASSEX wndclass ;                                // 변수 선언

// 메인 윈도우 클래스 생성 및 등록
wndclass.cbSize      = sizeof(wndclass) ;
wndclass.style       = CS_HREDRAW | CS_VREDRAW ;
wndclass.lpfnWndProc  = MainWndProc;                // 메인윈도우 프로시저
wndclass.cbClsExtra   = 0 ;
wndclass.cbWndExtra   = 0 ;
wndclass.hInstance   = hInstance ;
wndclass.hIcon        = LoadIcon(NULL,IDI_APPLICATION);
wndclass.hCursor      = LoadCursor(NULL,IDC_ARROW) ;
wndclass.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);
wndclass.lpszMenuName = MAKEINTRESOURCE(IDR_MENU1) ;
wndclass.lpszClassName = "Window Class Name";      // 메인윈도우 클래스 이름: 클래스 구분자
wndclass.hIconSm      = LoadIcon(NULL,IDI_APPLICATION);
RegisterClassEx(&wndclass);                          // 프레임 윈도우 클래스 등록

// 차일드 윈도우 클래스 생성 및 등록 : 차일드를 위해 wndclass를 재사용
wndclass.lpfnWndProc  = ChildWndProc;                // 차일드윈도우 프로시저
wndclass.lpszMenuName = NULL ;
wndclass.lpszClassName = "Child Window Class Name";  // 차일드윈도우 클래스 이름: 클래스 구분자

RegisterClassEx(&wndclass);                          // 차일드 윈도우 클래스 등록
```


윈도우 분할: 윈도우 프로시저

- 메인 윈도우 프로시저

```
HWND ChildHwnd[2];
```

```
LRESULT CALLBACK MainWndProc (HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)
{
    RECT rectView;
    switch (iMsg)
    {
        case WM_CREATE:
            GetClientRect(hwnd, &rectView);

            ChildHwnd[0] = CreateWindowEx ( WS_EX_CLIENTEDGE,
                "Child Window Class Name", NULL,
                WS_CHILD | WS_VISIBLE | WS_CLIPCHILDREN,
                0, 0, rectView.right, rectView.bottom/2-1,
                hwnd, NULL, hInst, NULL );

            ChildHwnd[1] = CreateWindowEx ( WS_EX_CLIENTEDGE,
                "Child Window Class Name", NULL,
                WS_CHILD | WS_VISIBLE | WS_CLIPCHILDREN,
                0, rectView.bottom/2+1, rectView.right, rectView.bottom/2-1,
                hwnd, NULL, hInst, NULL );

            break;
    }
}
```

윈도우 분할: 윈도우 프로시저

- 차일드 윈도우 프로시저

```
LRESULT CALLBACK ChildWndProc(HWND hwnd,UINT iMsg, WPARAM wParam,LPARAM lParam)
{
    HDC hdc;
    static int x[2]={20,20}, y[2]={20,20}, flag[2];
    int select;

    switch (iMsg) {
        case WM_TIMER:
            hdc = GetDC(hwnd);
            x[wParam] = x[wParam] + 20;
            Ellipse(hdc, x[wParam]-20, y[wParam]-20, x[wParam]+20, y[wParam]+20);
            ReleaseDC(hwnd, hdc);
            break;
        case WM_LBUTTONDOWN:
            if (hwnd == ChildHwnd[0])
                select = 0;
            else
                select = 1;
            flag[select] = 1 - flag[select];
            if (flag[select])
                SetTimer(hwnd, select, 100, NULL);
            else
                KillTimer(hwnd, select);
            break;
    }
    return DefWindowProc (hwnd, iMsg, wParam, lParam);
}
```

윈도우 분할: 윈도우 다루기 함수들

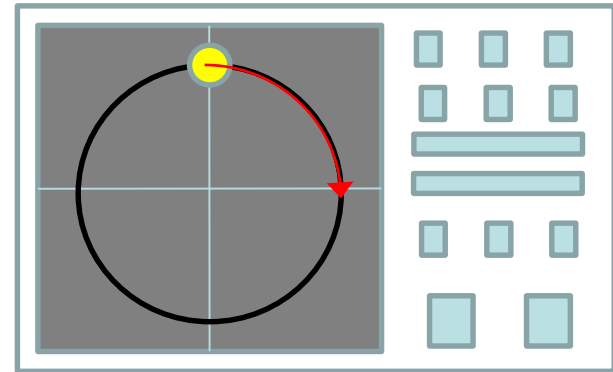
- 윈도우 다루기 함수

함수 원형	함수 소개
<code>BOOL MoveWindow (HWND hWnd, int x, int y, int nWidth, int nHeight, BOOL bRepaint);</code>	윈도우의 위치와 크기를 변경하는 함수
<code>HWND SetCapture (HWND hWnd) / HWND ReleaseCapture ();</code>	마우스를 윈도우 내에 캡처하는 함수 / 마우스 캡처를 해제하는 함수
<code>HWND SetFocus (HWND hWnd); / HWND GetFocus ();</code>	키보드 포커스를 설정하여 윈도우를 활성화 해주는 함수 / 키보드 포커스를 가진 윈도우 핸들 반환 함수
<code>BOOL IsChild (HWND hWndParent, HWND hWnd);</code>	hWnd 윈도우가 차일드 윈도우인지 확인
<code>HWND GetWindow (HWND hWnd, UINT uCmd);</code>	uCmd 관계를 가지고 있는 윈도우 핸들을 얻는 함수
<code>HWND GetParent (HWND hWnd);</code>	부모 윈도우 핸들을 얻는 함수
<code>HWND FindWindow (LPCSTR lpClassName, LPCSTR lpWindowName);</code>	윈도우 클래스 이름을 가진 윈도우를 찾는 함수

실습 7-1

• 공전하는 원 만들기

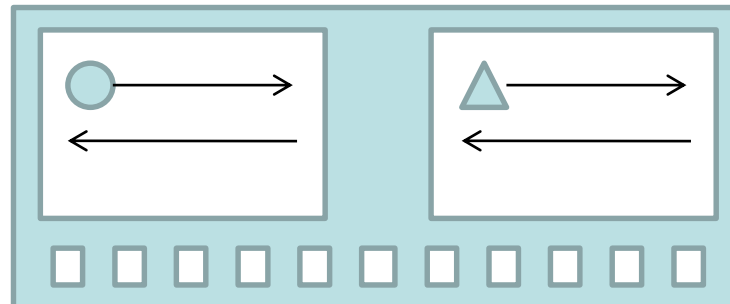
- 좌측에 차일드 윈도우를 만들고 중앙을 원점으로 x축과 y축 좌표계를 그린다.
 - 좌표계를 그리고 선택된 형태의 도형으로 궤도를 그린다.
 - 원이 궤도를 따라 이동한다.
- 우측에 차일드 윈도우를 사용하여 컨트롤을 놓는다.
 - 라디오 버튼: 공전 궤도 종류 (원 / 사각형 / 삼각형)
 - 라디오 버튼2: 궤도의 크기 대 / 중 / 소
 - 스크롤: 이동 원의 이동 속도를 증가 / 감소
 - 스크롤2: 이동 원의 크기를 크게 / 작게 조절
 - 체크박스: 이동 원의 색을 Cyan/Magenta/Yellow
 - 버튼: 원의 이동을 반대 방향으로 한다.
 - 버튼2: 종료 버튼



실습 7-2

• 2개의 차일드 윈도우 컨트롤하기

- 2개의 차일드 윈도우를 좌, 우에 만든다.
- 각각의 차일드 윈도우에 각각 1개의 도형이 그려진다.
- 좌측 차일드 윈도우
 - 버튼 1: 1번 윈도우의 도형이 좌/우로 지그재그 이동
 - 버튼 2: 1번 윈도우의 도형이 위/아래로 지그재그 이동
 - 버튼 3: 1번 윈도우의 도형이 제자리에서 점프
 - 버튼 4: 1번 윈도우의 도형이 멈추고 제자리에서 커졌다 작아졌다 반복함. (다시 누르면 이전에 이동하던 방향으로 다시 이동)
- 우측 차일드 윈도우
 - 버튼 5: 2번 윈도우의 도형이 좌/우로 지그재그 이동 (1번 도형과 다른 속도)
 - 버튼 6: 2번 윈도우의 도형이 위/아래로 지그재그 이동
 - 버튼 7: 2번 윈도우의 도형이 제자리에서 점프
 - 버튼 8: 2번 윈도우의 도형이 멈추고 제자리에서 커졌다 작아졌다 반복함. (다시 누르면 이전에 이동하던 방향으로 다시 이동)
- 컨트롤 차일드 윈도우
 - 버튼 9: 두 도형이 정지
 - 버튼 10: 두 도형이 바뀐다. (모양, 위치, 색 등)
 - 버튼 11: 프로그램 종료



실습 7-3