7장 차일드 윈도우

2018년도 1학기 윈도우 프로그래밍

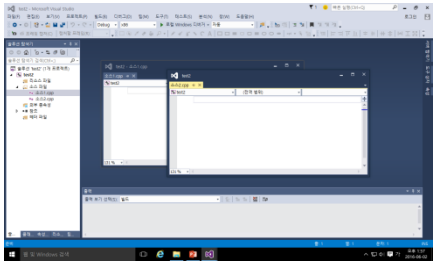
학습 목표

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

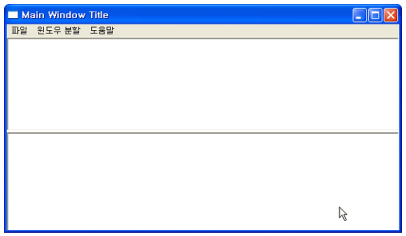
- **내용**
  - 컨트롤 윈도우 활용하기

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
  - 클래스 이름으로 구분한다.
  - 각자의 윈도우 프로시저를 가진다.
  - 차일드 윈도우 안에 또 다른 차일드 윈도우를 가질 수 있다.
  - 윈도우 스타일을 WS_CHILD | WS_VISIBLE 형태로 설정한다.
  - 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;                    // 클래스 스타일
wc.lpfnWndProc = WndProc;                              // 윈도우 프로시저 지정
wc.cbClsExtra = 0;                                      // 윈도우클래스 데이터 영역
wc.cbWndExtra = 0;                                     // 윈도우의 데이터 영역
wc.hInstance = hInstance;                              // 인스턴스 핸들
wc.hIcon = LoadIcon(NULL, IDI_APPLICATION);            // 아이콘 핸들
wc.hCursor = LoadCursor(NULL, IDC_ARROW);              // 사용할 커서 핸들
wc.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH); // 바탕색 브러쉬 핸들
wc.lpszMenuName = NULL;                                // 메뉴 이름
wc.lpszClassName = "ParentClass;                       // 윈도우 클래스 이름
wc.hIconSm = LoadIcon(NULL, IDI_APPLICATION);
RegisterClassEx(&wc);                                  // 윈도우 클래스를 등록

// 차일드 윈도우 클래스를 등록한다.
wc.hCursor = LoadCursor(NULL, IDC_HELP);
wc.hbrBackground = (HBRUSH) GetStockObject (GRAY_BRUSH);
wc.lpszClassName = "ChildClass";                       // 차일드 윈도우 클래스 이름
wc.lpfnWndProc = ChildProc;                            // 차일드 윈도우 프로시저 지정
RegisterClassEx(&wc);                                  // 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 클래스: 등록한 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 쑥 들어간 음각 모양으로 만든다.
    - WS_EX_WINDOWEDGE: 양각 모양의 경계선을 가진 윈도우를 만든다.
    - WS_EX_DLGMODALFRAME: 이중 경계선을 가진 윈도우를 만든다.

차일드 윈도우 만들기

- **차일드 윈도우 프로시저 만들기**

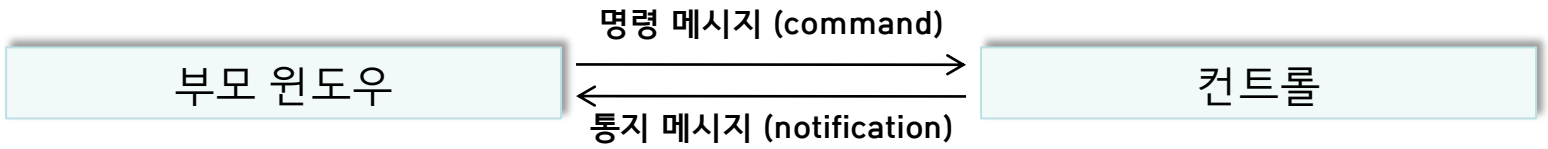
```
LRESULT CALLBACK ChildProc (HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
  switch (uMsg) {
        case WM_LBUTTONDOWN: // 마우스 좌측 버튼을 누른 경우
              MessageBox(hWnd, "자식 윈도우에서 마우스 왼쪽 버튼을 눌렀습니다", "마우스 테스트 ", MB_OK);
              break;
        case WM_DESTROY:
              PostQuitMessage(0);
              break;
  }
   return DefWindowProc(hWnd, uMsg, wParam, lParam);
}
```

- 기존의 윈도우 프로시저와 동일한 형태의 차일드 윈도우의 윈도우 프로시저를 포함한다.

컨트롤 차일드 윈도우

- **컨트롤도 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"

컨트롤	윈도우 클래스명	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);
  - 차일드 윈도우의 클래스 이름을 해당 컨트롤 윈도우 클래스로 설정한다.
    - button, edit, combobox 등
  - **윈도우의 스타일에서** 차일드 윈도우 인것을 표시한다.
    - dwStyle: Window style + Child Style (**WS_CHILD가 항상 포함된다.**)
      - 예) WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON
  - **부모 윈도우의 핸들을** 설정한다.
  - **차일드 윈도우의 아이디**를 메뉴값에 설정한다.
  - 차일드 윈도우의 아이디는 define하여 설정한다.
  - 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

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 윈도우 생성

- 버튼 생성: CreateWindow 함수로 버튼을 만든다.
- 버튼의 종류

윈도우 스타일	버튼 내용
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

- CreateWindow 함수에서
  - 클래스 이름: button
  - 버튼의 스타일:
    - WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON
  - 메뉴: 컨트롤의 id로 사용한다.

버튼 컨트롤 윈도우 생성

- **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 컨트롤 윈도우 생성

```
#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",                    // 버튼의 윈도우 클래스 이름은 "button"
                     WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON,      // 차일드 윈도우이고 누르는 형태의 버튼 스타일
                     200, 0, 100, 25, hwnd,
                     (HMENU) IDC_BUTTON, hInst, NULL);
       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;
    }
}
```

버튼 (체크 박스) 컨트롤 윈도우 생성

- **체크 박스 클래스: "button"**
- **체크 박스 스타일: WS_CHILD | WS_VISIBLE | BS_CHECKBOX**

- **체크박스 통지 메시지:**
  - 차일드 윈도우 → 부모 윈도우: BN_CLICKED 메시지를 보낸다.

  - 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
      - BM_GETCHECK: 체크 박스가 현재 체크되어 있는 상태인지를 조사하며 추가정보는 없다.
      - BM_SETCHECK: 체크 박스의 체크 상태를 변경하며 wParam에 변경할 체크 상태를 보내준다.

    - BM_GETCHECK에 의해 리턴 되는 값, 또는 BM_SETCHECK에 의해 설정되는 체크 박스의 상태
      - BST_CHECKED: 현재 체크되어 있다.
      - BST_UNCHECKED: 현재 체크되어 있지 않다.
      - BST_INDETERMINATE: 체크도 아니고 비 체크도 아닌 상태

버튼 (체크 박스) 컨트롤 윈도우 생성

- 체크박스 버튼 컨트롤

```
#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",                                   // 윈도우 클래스 이름은 button
                                        WS_CHILD | WS_VISIBLE | BS_CHECKBOX,                // 차일드 윈도우, 체크 박스
                                        10, 210, 180, 40, hwnd, (HMENU) IDC_BUTTON5,
                                        hInst, NULL);
                 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);
}
```

버튼 (라디오 버튼) 컨트롤 윈도우 생성

- **라디오 버튼 클래스: "button"**
- **라디오 버튼 스타일: WS_CHILD|WS_VISIBLE|BS_AUTORADIOBUTTON**

- **라디오 버튼 그룹**
  - 그룹 박스로 만들기
    - 그룹 박스 클래스: "button"
    - 그룹 박스 컨트롤 스타일: BS_GROUPBOX

  - 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

버튼 (라디오 버튼) 컨트롤 윈도우 생성

```
#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;
             }
        }

}
```

에디트 컨트롤 윈도우 생성

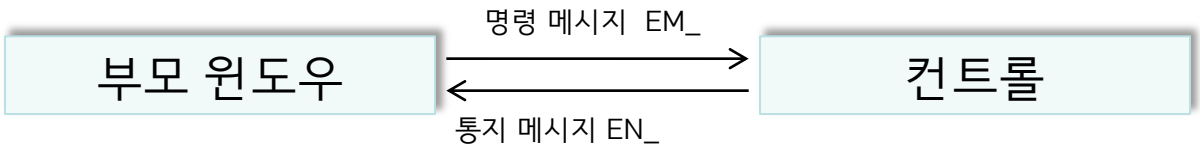
- **에디트 컨트롤 윈도우 스타일**

스타일	의미
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 윈도우 생성

- **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 윈도우 생성

```
#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:
            hEdit = CreateWindow ("edit", "에디팅", WS_CHILD | WS_VISIBLE | WS_BORDER,
                                    // 에디트 윈도우 클래스 이름은 "edit"
                                   // 차일드 윈도우이고 에디트 박스 주위에 테두리가 있는 스타일
                                   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 윈도우 생성

- **콤보 박스 스타일**
  - 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 종류

- **부모 윈도우가 리스트 박스에 보내는 메시지**
  - CB_ADDSTRING: 스트링 추가 메시지로
    - wParam: 사용하지 않음
    - lParam: 스트링의 시작주소
  - CB_DELETESTRING: 스트링을 삭제하는 메시지
    - wParam: 삭제할 스트링의 인덱스 번호
    - lParam: 사용하지 않음
  - CB_GETCOUNT: 저장되어 있는 스트링 아이템의 개수
    - wParam, lParam: 사용하지 않음
    - SendMessage()의 반환 값: 개수
  - CB_GETCURSEL: 선택된 스트링이 리스트에서 몇 번째 것인지
    - wParam, lParam: 사용하지 않음
    - SendMessage()의 반환 값: 개수

콤보 박스 윈도우 생성

```
#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 윈도우 생성

- **리스트 박스 스타일**
  - 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**
  - LB_ADDSTRING: 리스트 박스에 항목을 추가한다.
    - lParam: 추가하고자 하는 문자열의 번지를 넘겨주면 된다.
  - LB_DELETESTRING: 항목을 삭제한다.
    - wParam: 항목의 번호를 넘겨주며 남은 문자열수를 리턴한다.
  - LB_GETCURSEL:현재 선택된 항목의 번호(Index)를 조사해준다.
  - LB_GETTEXT: 지정한 항목의 문자열을 읽는다.
    - wParam: 항목 번호
    - lParam: 문자열 버퍼의 번지를 넘겨주면 버퍼에 문자열을 채워준다.
  - LB_GETCOUNT: 항목의 개수를 조사한다.
  - LB_SETCURSEL: wParam이 지정한 항목을 선택하도록 한다.

리스트 박스 생성

```
#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 생성

- **스크롤 바**
  - 스크롤 바는 "scrollbar"윈도우 클래스로 생성.
  - 수평 스크롤 바일 경우 **SBS_HORZ** 스타일, 수직 스크롤 바일 경우는 **SBS_VERT** 스타일을 지정.
  - 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 지정하는데 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 생성

- **스크롤 바의 현재 값을 지정**

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

  nPos: 스크롤 바의 현재 위치

- **다른 컨트롤들은 자신에게 변화가 있을 때 부모 윈도우로 통지 메시지를 보내는데 비해 스크롤 바는 WM_HSCROLL(수평일 경우), WM_VSCROLL(수직일 경우)이라는 별도의 메시지를 부모 윈도우로 보내며 추가 정보는 다음과 같다.**
  - 인수 정보

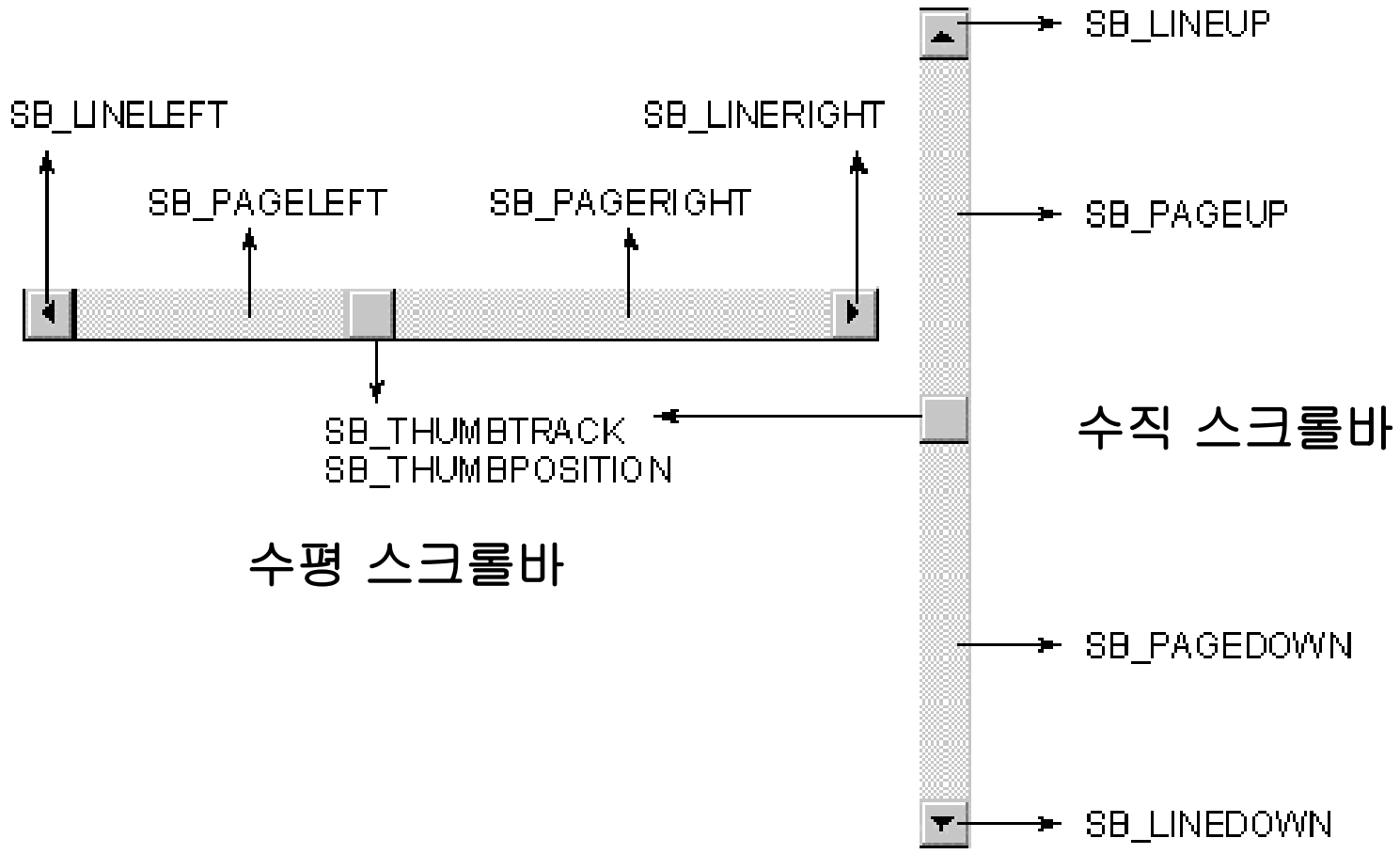
LOWORD(wParam)	스크롤 바 내의 어디를 눌렀는가?
HIWORD(wParam)	현재 위치
lParam	스크롤 바의 윈도우 핸들

스크롤 바 생성

- 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 때까지 계속 전달된다.

스크롤 바 생성

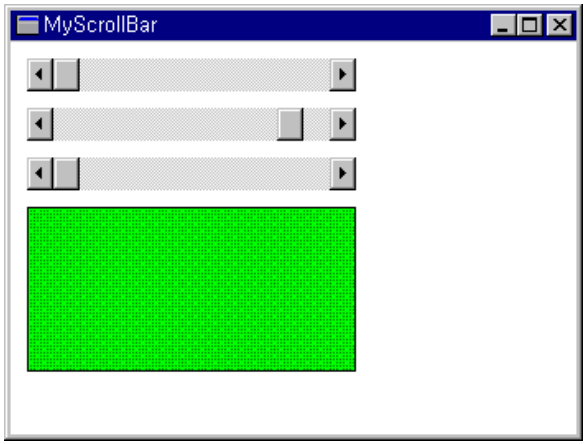

SB_LINEUP
SB_LINELEFT
SB_LINERIGHT
SB_PAGELEFT
SB_PAGERIGHT
SB_PAGEUP
SB_THUMBTRACK
SB_THUMBPOSITION
수직 스크롤바
수평 스크롤바
SB_PAGEDOWN
SB_LINEDOWN

스크롤 바 생성

```
#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_SCRGREEN, 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 생성

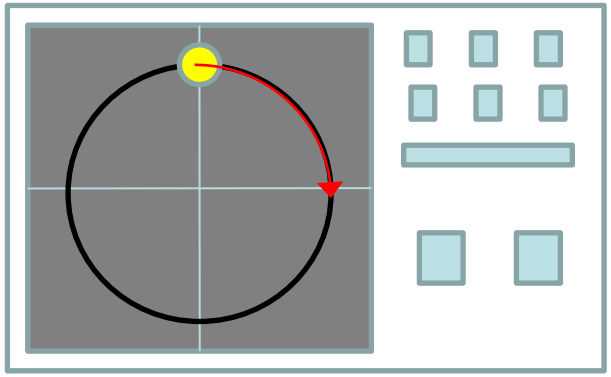
```
case WM_HSCROLL:
    if ((HWND)lParam == hRed)                   TempPos = Red;
    if ((HWND)lParam == hGreen)                 TempPos = Green;
    if ((HWND)lParam == hBlue)                  TempPos = Blue;

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

    SetScrollPos((HWND)lParam,  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;
```



실습 7-1

- **제목**
  - 공전하는 원 만들기

- **내용**
  - 좌측에 차일드 윈도우를 만들고 중앙을 원점으로 x축과 y축 좌표계를 그린다.
    - 좌표계에 선택된 형태의 곡선을 그린다.
    - 원이 곡선을 따라 이동한다.
  - 우측에 컨트롤을 놓는다.
    - 라디오 버튼: 곡선 종류 (원 / 사인곡선 / 스프링) - (실습 6-3 참고)
    - 라디오 버튼: 좌표계의 원의 크기 대 / 중 /소
    - 스크롤: 원의 이동 속도를 증가 / 감소
    - 버튼: 원의 이동을 반대 방향으로 한다.
    - 버튼2: 종료 버튼



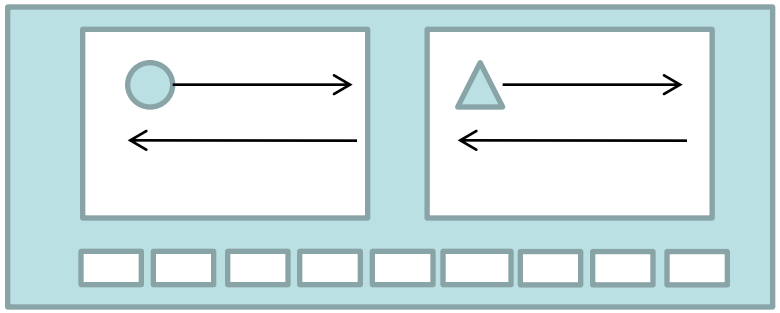
실습 7-2

- **제목**
  - 2개의 차일드 윈도우 컨트롤하기

- **내용**
  - 2개의 차일드 윈도우를 좌, 우에 만든다.
  - 각각의 차일드 윈도우에 각각 1개의 도형이 그려진다.
  - 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: 2번 윈도우의 도형이 좌/우로 지그재그 이동 (1번 도형과 다른 속도)
  - 버튼 5: 2번 윈도우의 도형이 위/아래로 지그재그 이동
  - 버튼 6: 2번 윈도우의 도형이 제자리에서 점프

  - 버튼 7: 두 도형이 정지
  - 버튼 8: 두 도형이 바뀐다. (모양, 위치, 색 등)
  - 버튼 9: 프로그램 종료



실습 7-3