사용자 인터페이스

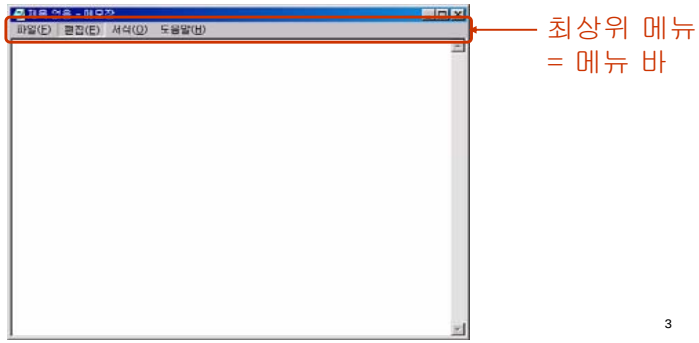
HCI Programming 2 (321190)
2007년 가을학기
11/2/2007
박경신

Overview

- 메뉴 명령을 처리하고 메뉴 항목을 적절하게 갱신 기법
- 컨텍스트 메뉴와 시스템 메뉴를 다루는 방법
- 툴바를 생성하고 사용하는 방법
- 상태바를 생성하고 사용하는 방법

메뉴 용어

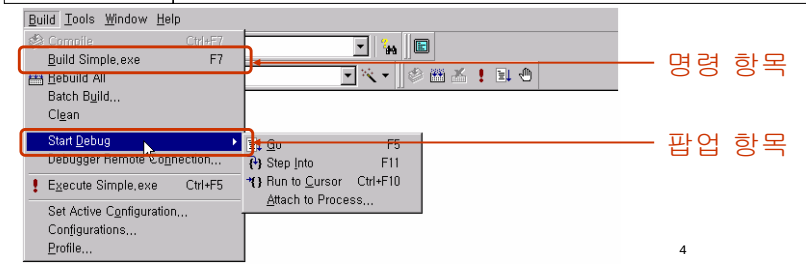
- 메뉴
  - 프로그램에서 선택할 수 있는 명령집합
  - 계층적요소로 구성된 사용자 인터페이스
- 최상위 메뉴 (top-level) = 메뉴 바 (menu bar)



메뉴 용어

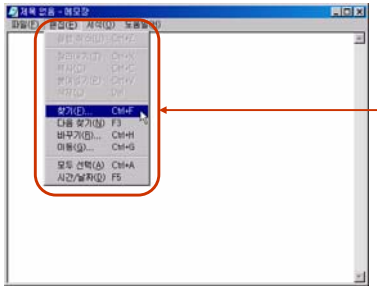
- 메뉴 항목 (menu item)

용어	의미
명령 항목	명령(Command)을 수행하는 메뉴 항목. 선택하면 WM_COMMAND 메시지가 발생한다.
팝업 항목	하위 메뉴를 화면에 표시하는 메뉴 항목. 선택해도 WM_COMMAND 메시지가 발생하지 않는다.



메뉴 용어

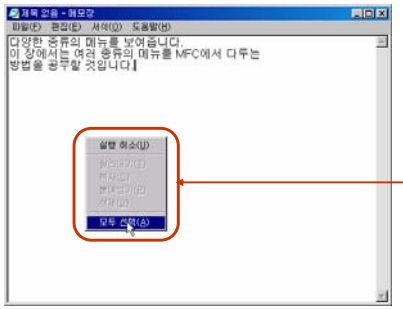
- 하위 메뉴 (submenu)
  - 팝업 항목을 선택했을 때 화면에 나타나는 메뉴
- 드랍다운 (drop-down) 메뉴
  - 최상위 메뉴 항목을 클릭했을 때 펼쳐지는 메뉴
  - 사용자가 특정항목을 선택하거나 취소하기 전까지 계속 열린 채로 유지



Drop-down menu
= 팝업 메뉴

메뉴 용어

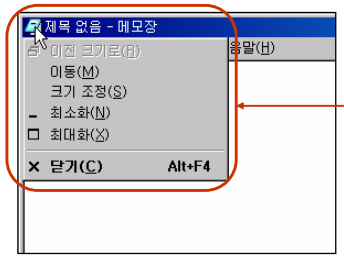
- 컨텍스트 메뉴 (context menu) = 단축 메뉴
  - 마우스 오른쪽 버튼을 누를 때 열리는 메뉴
  - 마우스 커서의 위치 또는 현재 작업하고 있는 내용에 따라 서로 다른 메뉴 항목이 표시됨



Context memu
= 단축 메뉴
= 팝업 메뉴

메뉴 용어

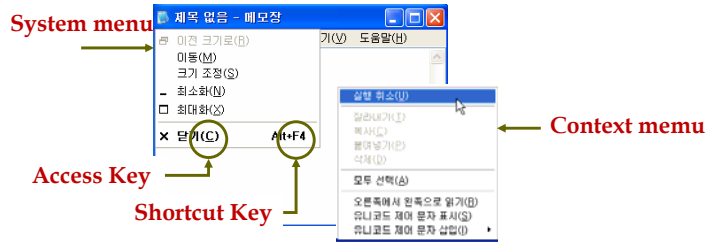
- 팝업 메뉴 (pop-up menu)
  - 사용자가 무엇인가를 선택했을 때 메뉴가 튀어나온다(Pop Up)는 뜻으로 만든 용어
  - 드랍다운 메뉴와 컨텍스트 메뉴가 여기에 속함
- 시스템 메뉴 (system menu) = 윈도우 메뉴



System menu
= 윈도우 메뉴

메뉴 용어

- 액세스키 (access key)와 단축키 (shortcut key)
  - 액세스키 - 메뉴가 열린 상태에서 특정 항목을 키보드로 빠르게 선택
  - 단축키 - 메뉴가 열리지 않은 상태에서도 키 조합으로 메뉴 항목의 기능을 곧바로 실행



메뉴 생성 및 추가방법

- 메뉴 생성 방법
  - 방법1
    - 메뉴 리소스를 만들어 메인프레임 생성시 로드하여 사용
  - 방법2
    - 프로그램 코드에서 메뉴 클래스(CMenu)를 사용하여 메뉴를 생성하고 메뉴항목을 구성하여 추가
    - CWnd::SetMenu() 메인프레임 윈도우에 메뉴 연결
- 동적 생성 메뉴를 기존메뉴에 추가
  - 메뉴 클래스(CMenu)를 사용하여 메뉴를 생성하고 메뉴의 구성요소를 설정
  - 메인프레임의 CWnd::GetMenu() 또는 메뉴리소스 ID를 사용한 CMenu::LoadMenu()를 이용하여 기존 메뉴를 얻기
  - 기존 메뉴에 CMenu::AppendMenu()를 이용하여 생성된 메뉴를 추가

메뉴 클래스

- MFC 클래스

  - CMenu
    - 메뉴를 생성하여 관리하는 클래스
    - 메뉴 생성, 메뉴항목추가 등 다양한 함수 제공
  - CCmdUI
    - CObject의 파생 클래스가 아닌 독립된 클래스
    - 사용자 인터페이스의 요소상태를 변경할 수 있는 클래스
    - 활성화 상태 변경, 체크 상태변경, 문자열 변경 등 메뉴와 툴바 및 상태바에 필요한 기능 지원

CMenu Class

메뉴: 윈도우 운영체제에서 관리하는 구조
메뉴객체: 프로그램에서 사용되는 C++ 객체

- 메뉴생성 및 항목 추가 등 메뉴관리를 위한 클래스
- 주요 메소드
  - CreateMenu() : 최상위 메뉴 생성 후 메뉴객체와 연결(attach)
  - CreatePopupMenu() : 팝업 메뉴 생성 후 메뉴객체와 연결(attach)
  - AppendMenu() : 새로운 메뉴항목을 메뉴에 추가
  - InsertMenu() : 메뉴항목 삽입
  - DeleteMenu() : 메뉴항목 삭제
  - LoadMenu() : 리소스로 존재하는 메뉴의 ID를 이용하여 메뉴를 얻기
  - GetSubMenu() : 메뉴의 인덱스를 이용하여 하위메뉴 추출
  - TrackPopupMenu() : WM_CONTEXTMENU 메시지 발생시 팝업 메뉴를 표시
  - Attach() : 메뉴와 메뉴객체를 연결
  - Detach() : 메뉴와 메뉴객체를 분리, 특정 메소드내에서 사용된 메뉴객체는 파괴되더라도 메뉴구조는 유지시켜야 할 때 사용

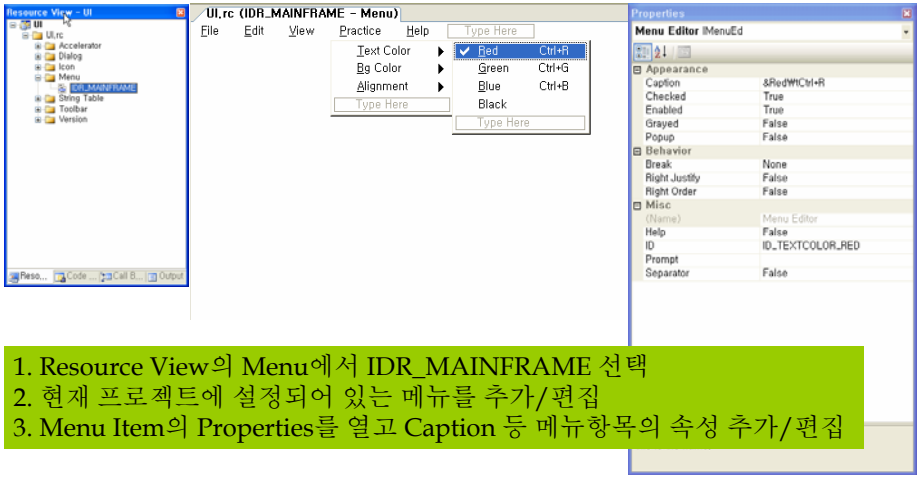
메뉴 생성

- AppWizard가 생성한 코드

```
BOOL CUIApp::InitInstance()
{
   // 생략...
   CMainFrame* pFrame = new CMainFrame;
   m_pMainWnd = pFrame;

   pFrame->LoadFrame(IDR_MAINFRAME,
      WS_OVERLAPPEDWINDOW | FWS_ADDTOTITLE, NULL,
      NULL);

   pFrame->ShowWindow(SW_SHOW);
   pFrame->UpdateWindow();

   return TRUE;
}
```

메뉴 생성

- 방법1: 메뉴 리소스를 작성하여 메뉴 생성
  - 리소스 뷰의 메뉴에서 IDR_MAINFRAME 선택



1. Resource View의 Menu에서 IDR_MAINFRAME 선택
2. 현재 프로젝트에 설정되어 있는 메뉴를 추가/편집
3. Menu Item의 Properties를 열고 Caption 등 메뉴항목의 속성 추가/편집

메뉴 생성

- 메뉴 항목 속성

속성	의미
ID	내부적으로 메뉴 항목을 구분하는 번호이며 일반적으로 ID_메뉴이름_항목이름 형태로 만든다. 예) ID_EDIT_CUT
Caption	화면에 표시되는 문자열로 액세스키를 지정하려면 **해당 문자 앞에 '&' 기호**를 사용한다. **단축키를 사용할 경우 '\t' 기호를 삽입**하여 단축키를 나타내는 문자열이 탭 위치에 정렬되도록 한다. 예) 잘라내기(&T)\tCtrl+X
Separator	메뉴 항목을 구분하는 가로줄이 표시된다.
Pop-up	설정하면 명령 항목이 아닌 팝업 항목이 된다. 최상위 메뉴는 대개 Pop-up 속성을 가진다.

메뉴 생성

- 메뉴 항목 속성

속성	의미
Inactive	메뉴 항목이 표시되지만 사용하지는 못한다.
Break	일반적으로 메뉴 항목은 하나의 열(Column)에 표시되지만 항목의 개수가 많을 경우 두 개 이상의 열에 표시되게 할 수 있다. None: 메뉴항목에 대해 열을 분리하지 않는다 Column: 다음 메뉴항목에 대해 열을 분리하여 표시한다 Bar: 다음 메뉴항목에 대해 열에 분리하고 분리선도 표시한다
Checked	메뉴 항목의 왼쪽에 체크 표시를 한다.

메뉴 생성

- 메뉴 항목 속성

속성	의미
Grayed	메뉴 항목이 흐리게 표시되어 현재 사용할 수 없음을 나타낸다.
Help	윈도우의 오른쪽 끝 위치에 메뉴가 표시되도록 한다. 주로 Help 메뉴 항목에 이 속성을 설정한다.
Prompt	MFC로 작성한 프로그램에서만 사용할 수 있는 속성으로, 툴바와 상태바에 표시될 문자열을 나타낸다. '\n'을 기준으로 앞쪽 문자열은 상태바에 표시되며 뒤쪽 문자열은 툴팁에 표시된다. 예) 선택 부분을 잘라내어 클립보드에 넣습니다\n잘라내기

메뉴 생성

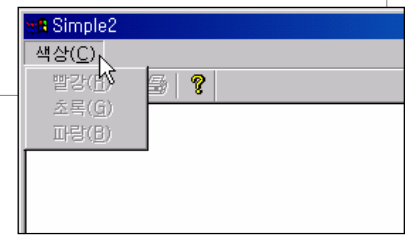
- 방법2: 프로그램 실행 중 전체 메뉴 생성하기

```
int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
   // 생략...

   CMenu menuMain;              // 메뉴 객체생성
   menuMain.CreateMenu();       // 메뉴바 생성 및 메뉴 객체에 연결
   CMenu menuPopup;
   menuPopup.CreatePopupMenu(); // 팝업메뉴 객체생성
   // 3개 메뉴항목 추가
   menuPopup.AppendMenu(MF_STRING, 201, "빨강(&R)");
   menuPopup.AppendMenu(MF_STRING, 202, "초록(&G)");
   menuPopup.AppendMenu(MF_STRING, 203, "파랑(&B)");
```


메뉴 생성

- 방법2: 프로그램 실행 중 전체 메뉴 생성하기

```
   // 메뉴바에 팝업메뉴 추가
   menuMain.AppendMenu(MF_POPUP,
      (UINT_PTR)menuPopup.Detach(), "색상(&C)");

   // 메뉴를 윈도우 메인프레임에 연결
   SetMenu(&menuMain);
   menuMain.Detach(); // 메뉴객체와 메뉴를 분리
   return 0;
}
```



AppendMenu 함수

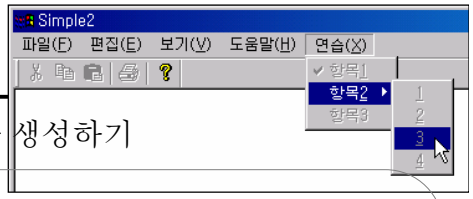
- 메뉴항목을 메뉴에 추가하는 함수

BOOL AppendMenu(UINT nFlags, UINT_PTR nIDNewItem=0,
    LPCTSTR lpszNewItem=NULL)

nFlags	의미	nIDNewItem	lpszNewItem
MF_STRING	메뉴항목이 문자열	새로운 항목의 ID	메뉴항목의 캡션 문자열
MF_POPUP	메뉴항목이 팝업메뉴를 가짐	팝업메뉴의 핸들(HWND 형식)	메뉴항목의 캡션 문자열
MF_CHECKED MF_UNCHECKED MF_ENABLED MF_DIABLED MF_GRAYED	메뉴항목 속성표시		
MF_SEPARATOR	Separator 속성 설정		

메뉴 생성

- 프로그램 실행 중 추가 메뉴 생성하기

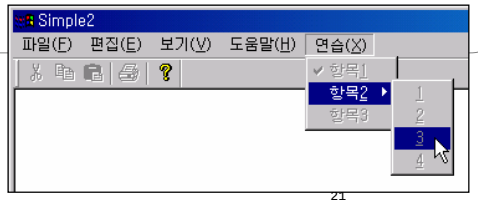



```
int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
   // 생략...

   CMenu Popup1; // Item2의 하위메뉴 생성
   Popup1.CreatePopupMenu();
   Popup1.AppendMenu(MF_STRING, 301, "&1");
   Popup1.AppendMenu(MF_STRING, 302, "&2");
   Popup1.AppendMenu(MF_STRING, 303, "&3");
   Popup1.AppendMenu(MF_STRING, 304, "&4");

   CMenu Popup2; // 메뉴항목 추가
   Popup2.CreatePopupMenu();
   Popup2.AppendMenu(MF_STRING|MF_CHECKED, 201, "Item&1");
```

메뉴 생성

- 프로그램 실행 중 추가 메뉴 생성하기

```
    Popup2.AppendMenu(MF_POPUP,
     (UINT_PTR)Popup1.Detach(), "Item&2");
    Popup2.AppendMenu(MF_STRING, 203, "Item&3");

    CMenu *pTopLevel = GetMenu(); // 연습메뉴를 최상위메뉴에 붙인다
    pTopLevel->AppendMenu(MF_POPUP,
                          (UINT_PTR)Popup2.Detach(), "연습(&X)");

    return 0;
}
```

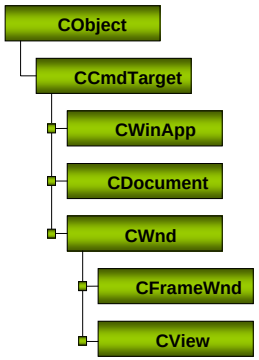


메뉴 명령 처리

- 메뉴 명령 처리 과정
  1. 명령 항목을 마우스나 키보드로 선택
  2. WM_COMMAND 메시지 발생
  3. WM_COMMAND 메시지 핸들러에서 메뉴 명령 처리
- MFC의 메뉴 명령 처리 방법
  - 각각의 메뉴 항목에 대해 함수를 따로 작성
    - 명령 핸들러(Command Handler)
  - ON_COMMAND(메뉴ID, 함수명) 매크로를 이용하여 메뉴 항목과 함수 연결
- 명령 라우팅
  - 명령 핸들러를 작성하는 위치에 관계없이 처리

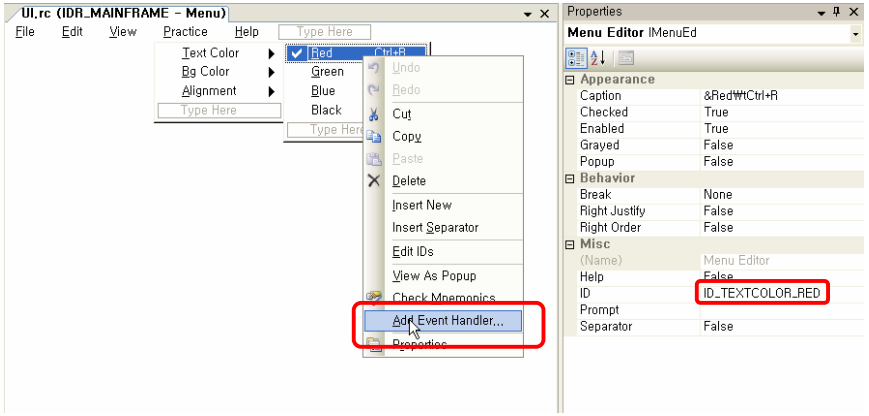
Command Routing

- WM_COMMAND 메시지는 CCmdTarget클래스로부터 상속된 하위 객체에서 처리가능
- WM_COMMAND 메시지를 처리할 메시지핸들러가 여러 객체에서 정의되어있는 경우 명령 전달 경로의 순서로 해당 메시지가 처리
- Command Handler 설치 기준
  - 메뉴명령의 목적에 따라 설치
  - 데이터 관리-Document
  - 클라이언트 영역관련 작업-View
  - 메인 프레임 윈도우 관련 작업 - FrameWnd



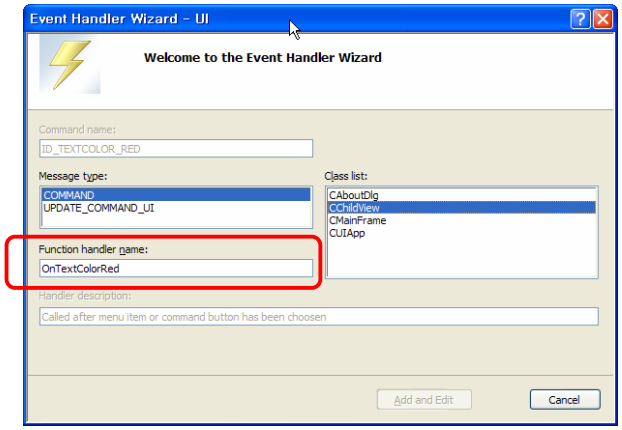
Command Message Handler

- IDR_MAINFRAME의 메뉴항목에서 Add Event Handler를 선택



Command Message Handler

- Event Handler Wizard를 이용하여 메뉴항목 선택 시 실행되는 명령 메시지 핸들러 생성



메뉴 명령 처리

- 메뉴 명령 처리 예

```
BEGIN_MESSAGE_MAP(CChildView,CWnd )
   ...
   ON_COMMAND(ID_TEXTCOLOR_RED, OnTextColorRed)
   ON_COMMAND(ID_TEXTCOLOR_GREEN, OnTextColorGreen)
   ON_COMMAND(ID_TEXTCOLOR_BLUE, OnTextColorBlue)
   ...
END_MESSAGE_MAP()

void CChildView::OnTextColorRed()
{
   m_textColor = RGB(255, 0, 0);
   Invalidate();
}
```

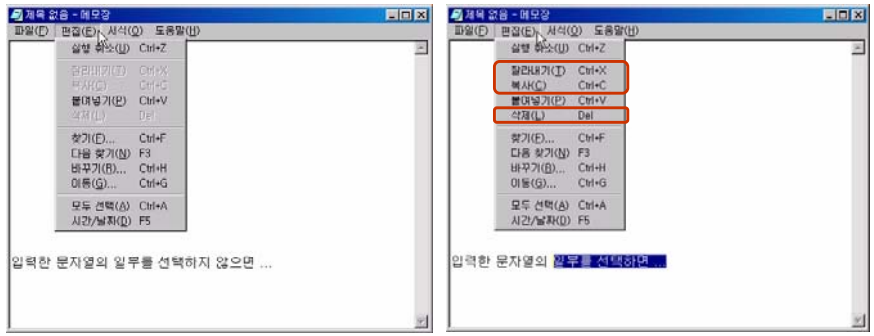

메뉴 명령 처리

- 메뉴 명령 처리 예

```
void CChildView::OnTextColorGreen()
{
   m_textColor = RGB(0, 255, 0);
   Invalidate();
}

void CChildView::OnTextColorBlue()
{
   m_textColor = RGB(0, 0, 255);
   Invalidate();
}
```


메뉴 항목 갱신

- 메뉴 항목 갱신 예

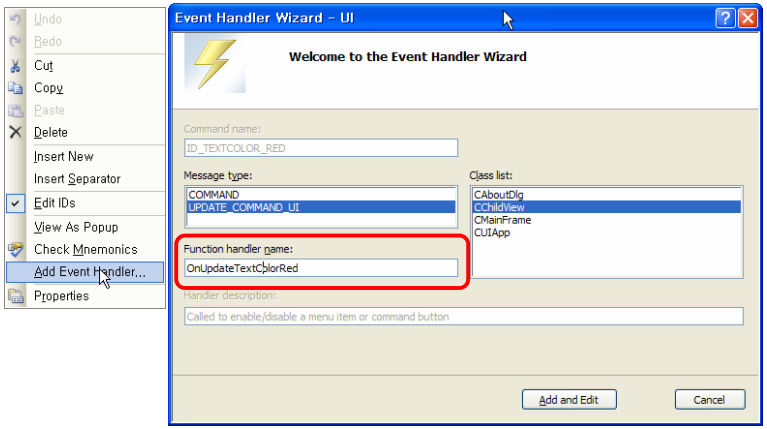


메뉴 항목 갱신

- MFC의 메뉴 항목 갱신 방법
  - 각각의 메뉴 항목에 대해 함수를 따로 작성
    - 명령 갱신 핸들러 (Command Update Handler)
  - ON_UPDATE_COMMAND_UI(메뉴ID, 함수명) 매크로를 이용하여 메뉴 항목과 함수 연결
- 명령 라우팅
  - 명령 핸들러와 마찬가지로 명령 갱신 핸들러도 작성하는 위치에 관계없이 처리

Command Update Handler

- Event Handler Wizard를 이용하여 메뉴항목 선택 시 실행되는 명령 갱신 핸들러 생성



메뉴 항목 갱신

- 메뉴 항목 갱신 예

```
BEGIN_MESSAGE_MAP(CChildView,CWnd )
   ...
   ON_UPDATE_COMMAND_UI(ID_COLOR_RED,
                                    OnUpdateTextColorRed)
   ON_UPDATE_COMMAND_UI(ID_COLOR_GREEN,
                                    OnUpdateTextColorGreen)
   ON_UPDATE_COMMAND_UI(ID_COLOR_BLUE,
                                    OnUpdateTextColorBlue)
   ...
END_MESSAGE_MAP()
//메뉴항목변경 메시지 핸들러
void CChildView::OnUpdateTextColorRed(CCmdUI* pCmdUI)
{
    pCmdUI->SetCheck(m_textColor == RGB(255, 0, 0));
}
```


메뉴 항목 갱신

- 메뉴 항목 갱신 예

```
void CChildView::OnUpdateTextColorGreen(CCmdUI* pCmdUI)
{
    pCmdUI->SetCheck(m_textColor == RGB(0, 255, 0));
}

void CChildView::OnUpdateTextColorBlue(CCmdUI* pCmdUI)
{
    pCmdUI->SetCheck(m_textColor == RGB(0, 0, 255));
}
```

CCmdUI 클래스

- 명령을 수행하는 사용자인터페이스를 변경하는 클래스
- 주요 메소드
  - Enable : 메뉴항목 활성화(TRUE)/비활성화(FALSE)
  - SetText : 메뉴항목 문자열 변경
  - SetCheck : 메뉴항목에 체크표시 보이게(1)/보이지 않게(0) 설정
  - SetRadio : 메뉴항목에 원점표시 보이게(1)/보이지 않게(0) 설정

멤버 함수	의미	사용 예
Enable( )	활성화 상태 변경	pCmdUI->Enable(b_DrawMode);
SetCheck( )	체크 상태 변경	pCmdUI->SetCheck(m_textColor == RGB(255, 0, 0));
SetRadio( )	라디오 표시 상태 변경	pCmdUI->SetRadio(m_textColor == RGB(255, 0, 0));
SetText( )	문자열 변경	pCmdUI->SetText("Light On");

컨텍스트 메뉴

- 마우스 오른쪽 버튼을 누르면 열리는 단축 메뉴
- 마우스 커서위치, 현재작업에 따라 다른 메뉴항목 표시
- WM_CONTEXTMENU 메시지 발생 상황
  - 클라이언트 영역 또는 비 클라이언트 영역에서 마우스 오른쪽 버튼을 클릭하는 경우
  - Shift + F10 키 조합을 누른 경우
  - 가상 키코드 VK_APPS에 해당하는 키를 누른 경우

- WM_CONTEXTMENU 메시지 핸들러

```
afx_msg void OnContextMenu (CWnd* pWnd, CPoint pos) ;
```

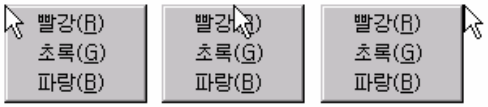
  - pWnd - 마우스 커서 아래쪽에 있는 윈도우
  - pos - 마우스 커서의 위치(스크린 좌표)

컨텍스트 메뉴

- CMenu::TrackPopupMenu() 함수
  - WM_CONTEXTMENU 메시지 발생시 팝업 메뉴를 표시하기 위한 함수

```
BOOL TrackPopupMenu (UINT nFlags, int x, int y, CWnd* pWnd, LPCRECT
    lpRect = 0) ;
```

  - nFlags: 컨텍스트 메뉴의 마우스 커서 위치 및 선택 버튼
    - TPM_LEFTALIGN, TPM_CENTERALIGN, TPM_RIGHTALIGN



    - TPM_LEFTBUTTON, TPM_RIGHTBUTTON

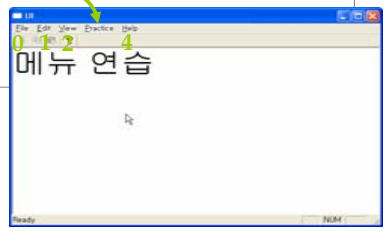
컨텍스트 메뉴

- x, y
  - 컨텍스트 메뉴가 표시될 위치(스크린 좌표)
- pWnd
  - 컨텍스트 메뉴에서 발생한 WM_COMMAND 메시지를 받을 윈도우
- lpRect
  - 마우스 버튼을 클릭하더라도 컨텍스트 메뉴가 닫히지 않는 사각형 영역(스크린 좌표)

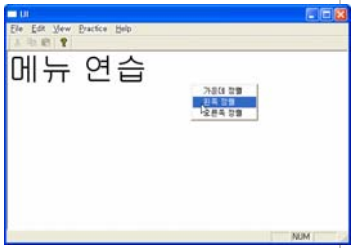
컨텍스트 메뉴

- 컨텍스트 메뉴 사용 예

```
void CChildView::OnContextMenu(CWnd* pWnd, CPoint point)
{
    CMenu menu;
    menu.LoadMenu(IDR_MAINFRAME);
    CMenu* pMenu = menu.GetSubMenu(3);
    pMenu->TrackPopupMenu(
        TPM_LEFTALIGN|TPM_RIGHTBUTTON,
        point.x, point.y, AfxGetMainWnd());
}
```



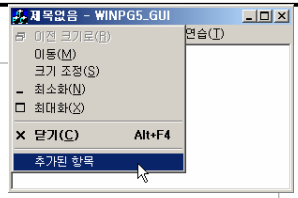

```
//새로운 메뉴를 동적으로 생성하여 컨텍스트 메뉴로 사용한 예
class CChildView : public CView
{// Generated message map functions
protected:
        //{{AFX_MSG(CUIIView)
        afx_msg void OnContextMenu(CWnd* pWnd, CPoint point);
        afx_msg void OnAlignmentCenter();   //메시지 핸들러 선언
        ...
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
...
};
BEGIN_MESSAGE_MAP(CUIView, CView)
        //{{AFX_MSG_MAP(CUIIView)
        ON_WM_CONTEXTMENU()
        ON_COMMAND(201, OnAlignmentCenter)  //메시지 맵
        ...
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()
void CChildView::OnContextMenu(CWnd* pWnd, CPoint point)
{
        CMenu menu;
        menu.CreatePopupMenu();  //팝업메뉴 생성
        menu.AppendMenu(MF_STRING, 201, "가운데 정렬");  //3개의 메뉴항목 추가
        menu.AppendMenu(MF_STRING, 202, "왼쪽 정렬");
        menu.AppendMenu(MF_STRING, 203, "오른쪽 정렬");
        //메뉴를 컨텍스트 팝업메뉴로 표시
        menu.TrackPopupMenu(TPM_LEFTALIGN, point.x, point.y, AfxGetMainWnd());
}
void CChildView::OnAlignmentCenter()
{
        m_textPoint =  DT_CENTER ;    //가운데 정렬
        Invalidate();
}
```



시스템 메뉴

- 윈도우 조작과 관련된 메뉴
  - 이동/크기조정/최대화/최소화/닫기
- CWnd::GetSystemMenu()
  - 시스템 메뉴의 CMenu 포인터를 얻기 위한 함수
  - CMenu 클래스가 제공하는 다양한 함수(AppendMenu(), InsertMenu(), DeleteMenu(), ...)를 적용
- 주의 사항
  - 시스템 메뉴를 변경하려면 GetSystemMenu(FALSE)를, 시스템 메뉴를 초기 상태로 되돌리려면 GetSystemMenu(TRUE)를 호출한다.
  - 시스템 메뉴에 새로운 메뉴 항목을 추가할 때 메뉴 ID는 반드시 16의 정수배가 되어야 한다.
  - 시스템 메뉴 항목을 선택하면 WM_COMMAND가 아닌 WM_SYSCOMMAND 메시지가 발생한다.

시스템 메뉴

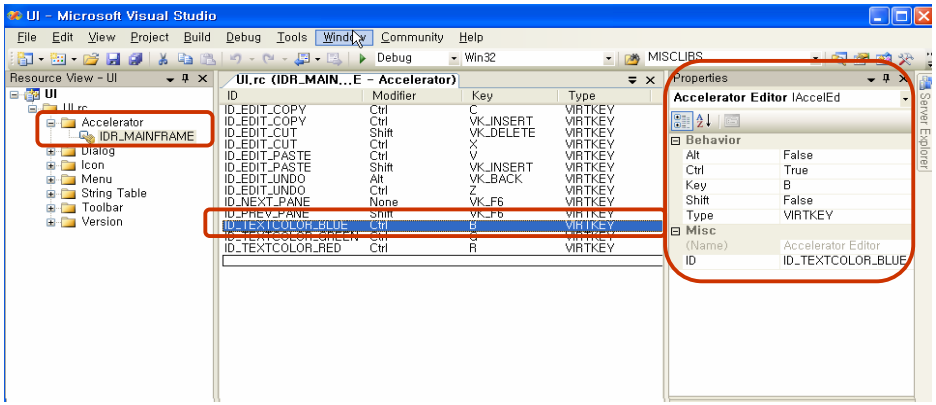



```
class CMainFrame : public CFrameWnd
{
rotected:
        //{{AFX_MSG(CMainFrame)
        ...
        afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
}
BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
        //{{AFX_MSG_MAP(CMainFrame)
        ...
        ON_WM_SYSCOMMAND()
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

```
int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
	if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
		return -1;
	//시스템 메뉴 수정
	CMenu * pSysMenu = GetSystemMenu(FALSE);
	pSysMenu->AppendMenu(MF_SEPARATOR);  //구분선 추가
	//ID가 16의 배수가 되도록 설정
	pSysMenu->AppendMenu(MF_STRING, 16, "추가된 항목");
	return 0;
}
void CMainFrame::OnSysCommand(UINT nID, LPARAM lParam)
{
// nID의 하위 4비트는 운영체제가 사용하므로 이를 무시하기 위해
// 0xFFF0과  AND 연산하여 비교
	if ((nID & 0xFFF0) == 16){
		AfxMessageBox("시스템 메뉴 연습입니다.");
		return;
	}
	CFrameWnd::OnSysCommand(nID, lParam);
}
```

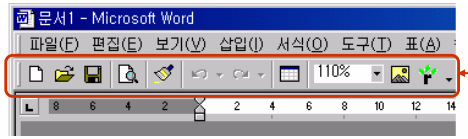
가속기

- 가속기 = 단축키
  - 메뉴 항목을 곧바로 실행할 수 있는 키 조합
- 가속기 리소스에 가속기를 만들 메뉴항목 추가
- 가속기를 누르면 WM_COMMAND 메시지가 발생



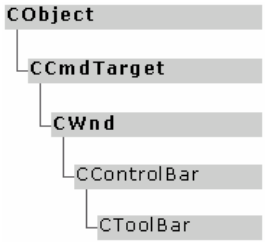
툴바 (Toolbar)

- 메뉴항목 기능을 빠르게 수행하기 위한 명령버튼
- WM_COMMAND 메시지 발생



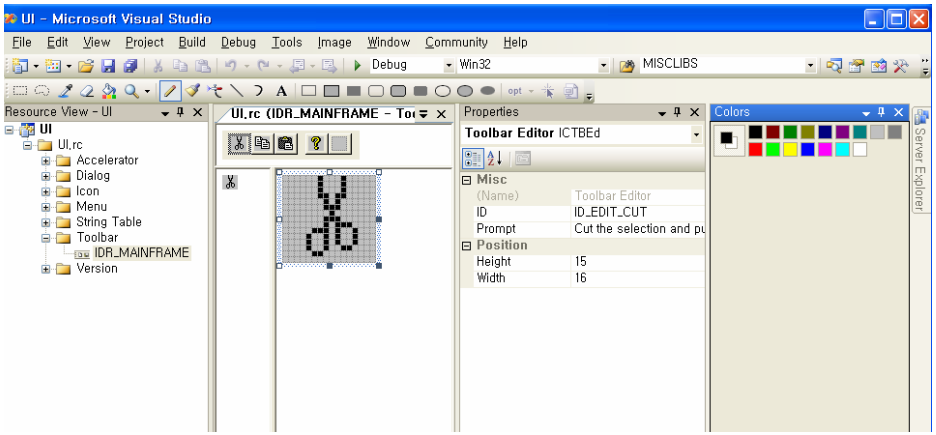
선택하면
WM_COMMAND
메시지 발생!

- MFC 클래스



툴바

- 툴바를 구성하는 버튼들이 비트맵으로 배치
- 툴바 버튼 수정/추가/이동/삭제 처리
- 툴바 버튼을 마우스로 더블 클릭하여 속성 설정



툴바

- 툴바 코드

```
class CMainFrame : public CFrameWnd
{
// 생략 ...
protected:
    CStatusBar m_wndStatusBar;
    CToolBar m_wndToolBar;
    CChildView m_wndView;
// 생략 ...
};

int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    // 생략 ...
    if (!m_wndToolBar.CreateEx(this, TBSTYLE_FLAT, WS_CHILD |
```

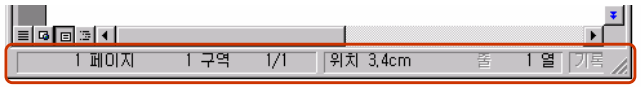

툴바

- 툴바 코드

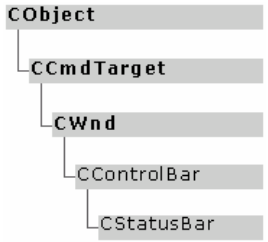
```
        WS_VISIBLE | CBRS_TOP | CBRS_GRIPPER |
    CBRS_TOOLTIPS |
        CBRS_FLYBY | CBRS_SIZE_DYNAMIC) ||
        !m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
    {
        TRACE0("Failed to create toolbar\n");
        return -1;
    }
    // 생략 ...
    m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
    EnableDocking(CBRS_ALIGN_ANY);
    DockControlBar(&m_wndToolBar);

    return 0;
}
```


상태바

- 메인 프레임 하단부에 위치하여 상태를 확인하는 윈도우
- 상태바는 팬이라 불리는 표시 영역들로 나뉜다
  - 고정되거나 변화할 수 있는 크기를 갖는다
  - 제일 왼쪽의 팬은 가변 크기 팬이다
  - 팬번호는 왼쪽으로부터 0, 1, 2, 3 부여



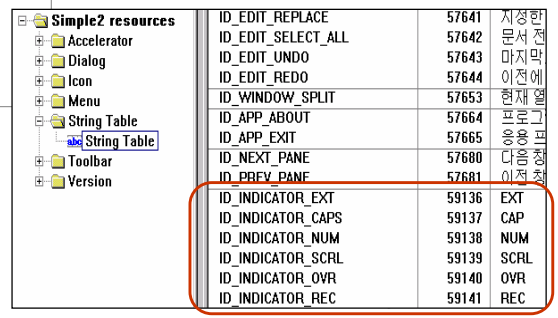
- MFC 클래스



상태바

- 상태바 리소스

```
static UINT indicators[ ] =
{
    ID_SEPARATOR,
    ID_INDICATOR_CAPS,
    ID_INDICATOR_NUM,
    ID_INDICATOR_SCRL,
};
```



상태바

- 상태바 코드

```
class CMainFrame : public CFrameWnd
{
// 생략 ...
public:
    CStatusBar m_wndStatusBar;          //상태바의 멤버변수 선언
protected:
    CToolBar m_wndToolBar;
    CChildView m_wndView;
// 생략 ...
};
```


상태바

- 상태바 코드

```
int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
   // 생략 …
   // 상태바 생성 후 상태바 객체와 연결
   // 상태바의 각 펜을 해당 문자열 리소스 값으로 설정
   if (!m_wndStatusBar.Create(this) ||
      !m_wndStatusBar.SetIndicators(indicators,
         sizeof(indicators)/sizeof(UINT)))
   {
      TRACE0("Failed to create status bar\n");
      return -1;
   }
   // 생략 ...
}
```


상태바에 새로운 팬 생성 단계

- 팬(Pane)을 생성
  - 추가할 팬의 ID 생성-[Resource Symbol]대화상자 이용
  - 생성된 팬의 초기문자열 등록-[String Table]의 빈칸에 생성한 팬의 ID, 초기문자열 입력
- 상태바에 팬을 추가
  - 상태바를 분할하는 지시자 설정에 새로운 팬을 추가

```
static UINT indicators[] =
{
    ID_SEPARATOR,
    ID_INDICATOR_POS;  // 새로운 팬 추가 nIndex=1
    ID_INDICATOR_CAPS,
    ID_INDICATOR_NUM,
    ID_INDICATOR_SCRL,
};
```

상태바 팬생성 예

- 상태바에 새로운 팬을 추가하여 마우스 좌표를 출력하는 프로그램
  - 칸 속성 지정

```
int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
 // 생략..

 // 상태바nIndex=1의칸 속성조정
 m_wndStatusBar.SetPaneInfo(1, ID_INDICATOR_POS,
                                        SBPS_NORMAL, 120);
 return 0;
}
```

상태바 팬생성 예

- 상태바에 새로운 팬을 추가하여 마우스 좌표를 출력하는 프로그램
  - 칸을 활성화

```
BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
        ON_WM_CREATE()
        ON_WM_SETFOCUS()
        ON_UPDATE_COMMAND_UI(ID_INDICATOR_POS,
                                        OnUpdateIndicatorPos)
END_MESSAGE_MAP()

// 상태바 nIndex=1의칸을활성화
void CMainFrame::OnUpdateIndicatorPos(CCmdUI* pCmdUI)
{
        pCmdUI->Enable();
}
```

상태바 팬생성 예

- 상태바에 새로운 팬을 추가하여 마우스 좌표를 출력하는 프로그램
  - 상태바에 새로운 문자열 출력 : SetPanelText()함수 이용

```
void CChildView::OnMouseMove(UINT nFlags, CPoint point)
{
  //View에서 MainFrame객체의 포인터 얻기
  CMainFrame *pFrame = (CMainFrame *)AfxGetMainWnd();
  CString strStatus;
  strStatus.Format("마우스 X : %d, Y : %d ", point.x, point.y);
  //메인프레임 상태바의 nIndex=1인 팬에 문자열 설정
  pFrame->m_wndStatusBar.SetPaneText(1, strStatus);

  CWnd::OnMouseMove(nFlags, point);
}
```