쉽게 배우는 MFC 윈도우 프로그래밍

Visual C++ 2019

2015·2017 버전 사용 가능

6장. 사용자 인터페이스

목차

메뉴 기초

메뉴 용어

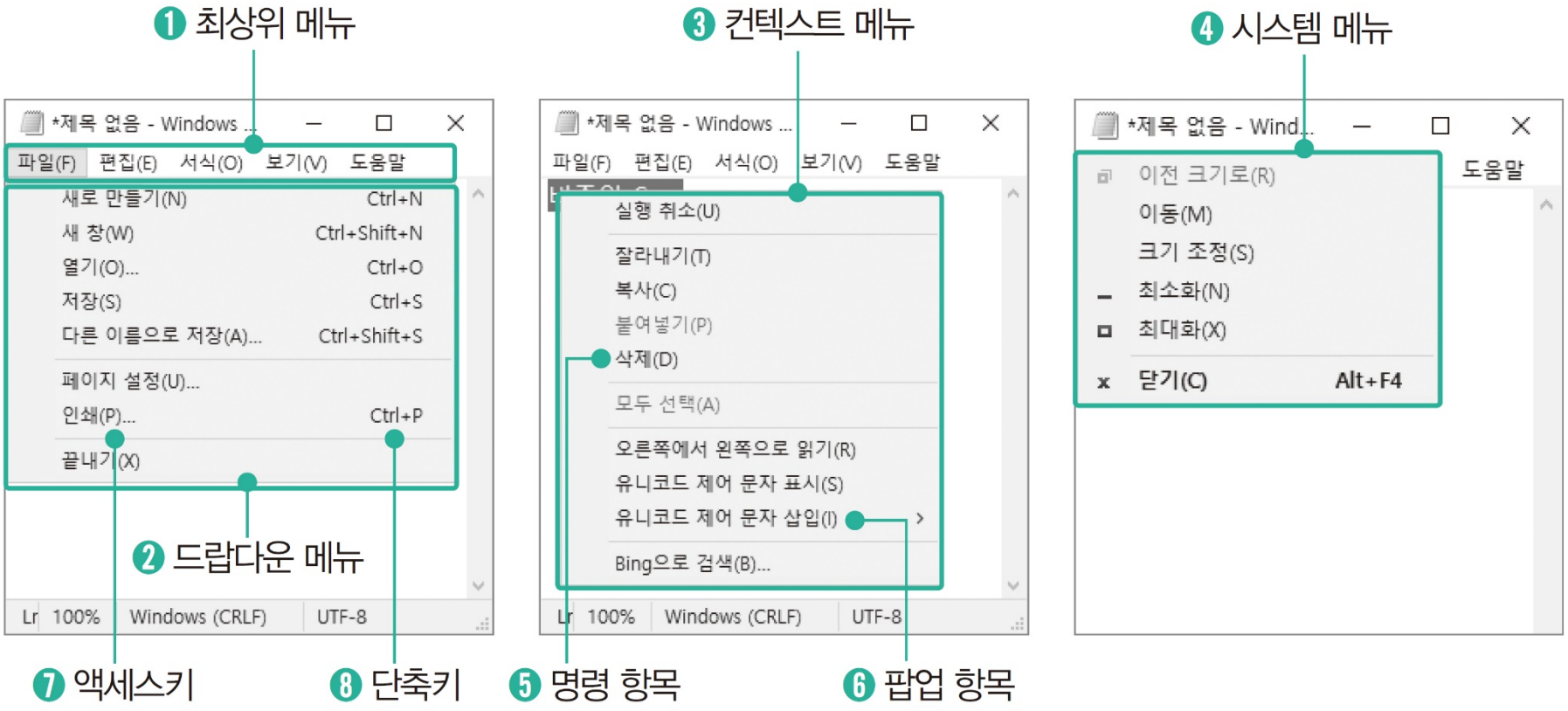


그림 6-1 메뉴 관련 용어

메뉴 기초

표 6-1 메뉴 항목의 종류

용어	의미
명령 항목(❺)	• 명령을 수행하는 메뉴 항목 • 선택 시 WM_COMMAND 메시지 발생
팝업 항목(❻)	• 하위 메뉴를 표시하는 메뉴 항목 • 선택 시 WM_COMMAND 메시지 발생하지 않음

[실습 6-1] MFC 응용 프로그램 생성하기

그림 6-2 프로젝트 종류 선택

[실습 6-1] MFC 응용 프로그램 생성하기

그림 6-3 프로젝트 이름과 위치 지정

[실습 6-1] MFC 응용 프로그램 생성하기

표 6-2 응용 프로그램 마법사 단계별 변경 사항

단계	변경 사항
애플리케이션 종류	애플리케이션 종류로 '단일 문서' 선택 '문서/뷰 아키텍처 지원' 체크 해제
문서 템플릿 속성	변경 없음 – 모든 기능 비활성화되어 있음
사용자 인터페이스 기능	변경 없음 – '초기 상태 표시줄'과 '클래식 도킹 도구 모음 사용' 선택되어 있음
고급 기능	모든 옵션 해제

[실습 6-1] MFC 응용 프로그램 생성하기

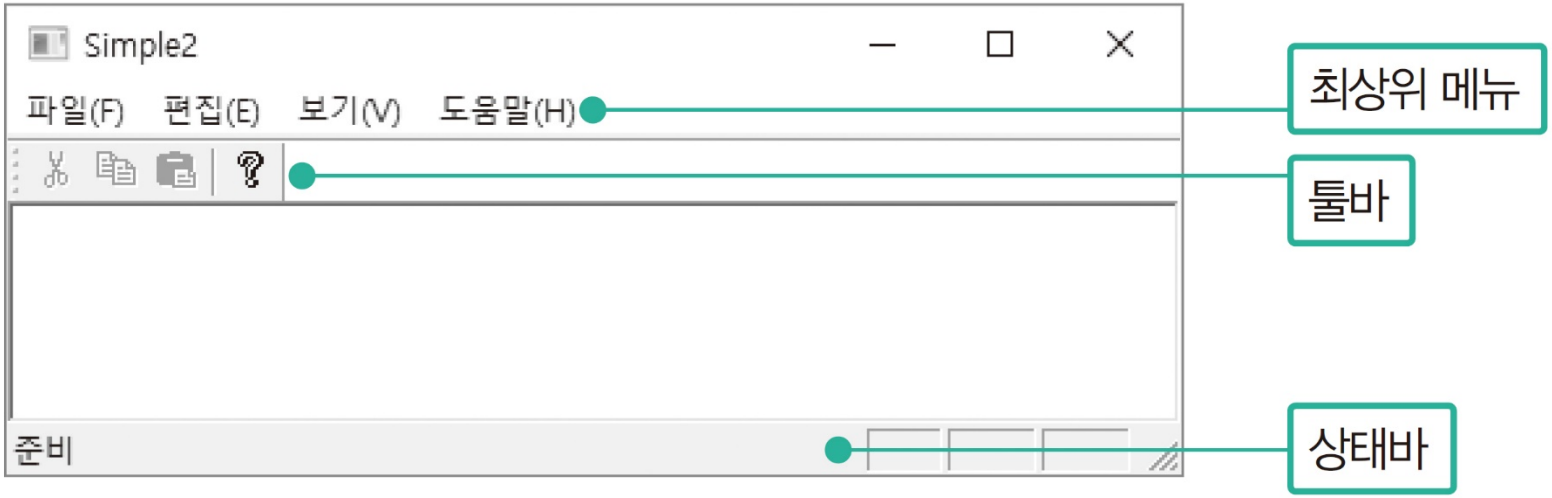


그림 6-4 실행 결과

메뉴 생성

- 윈도우 응용 프로그램에서 메뉴 생성 방법
  - 리소스를 이용한 메뉴 생성 : 메뉴 리소스 정의 → 실행 파일에 포함 → 프로그램 실행 중에 불러옴
  - 코드를 이용한 메뉴 생성 : 코드를 실행하여 메뉴 생성 → 윈도우에 붙여서 사용

리소스를 이용한 메뉴 생성

- 메뉴 리소스 추가

그림 6-5 메뉴 리소스 추가

리소스를 이용한 메뉴 생성

- 메뉴 리소스 추가

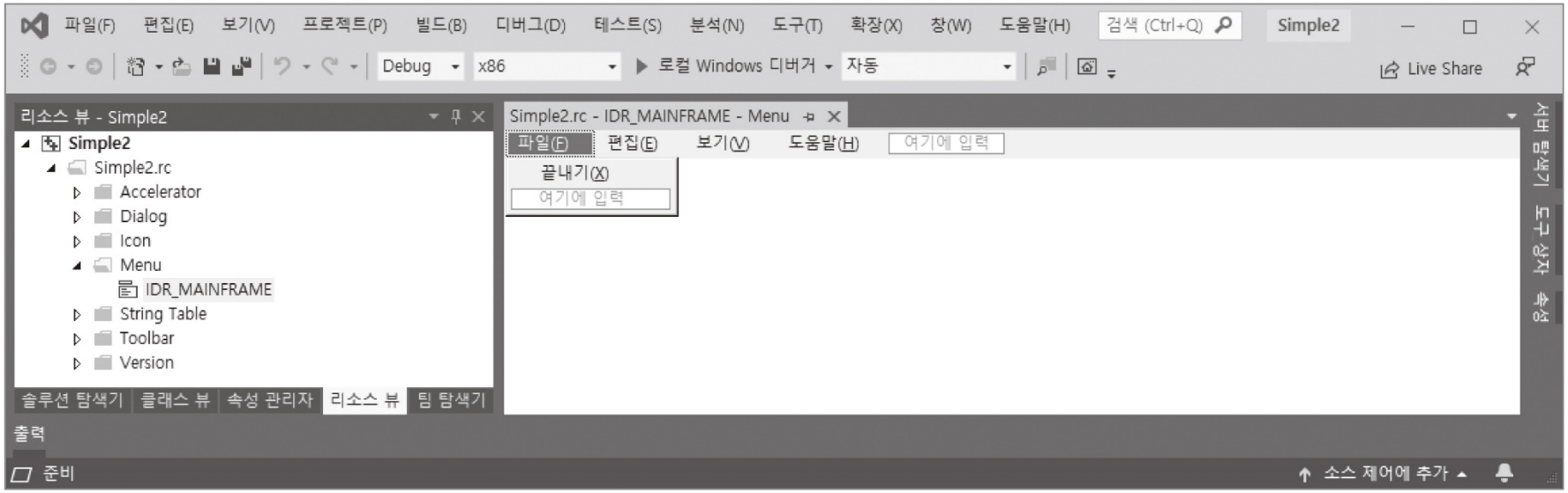


그림 6-6 메뉴 리소스

리소스를 이용한 메뉴 생성

- 응용 프로그램 마법사가 생성한 코드

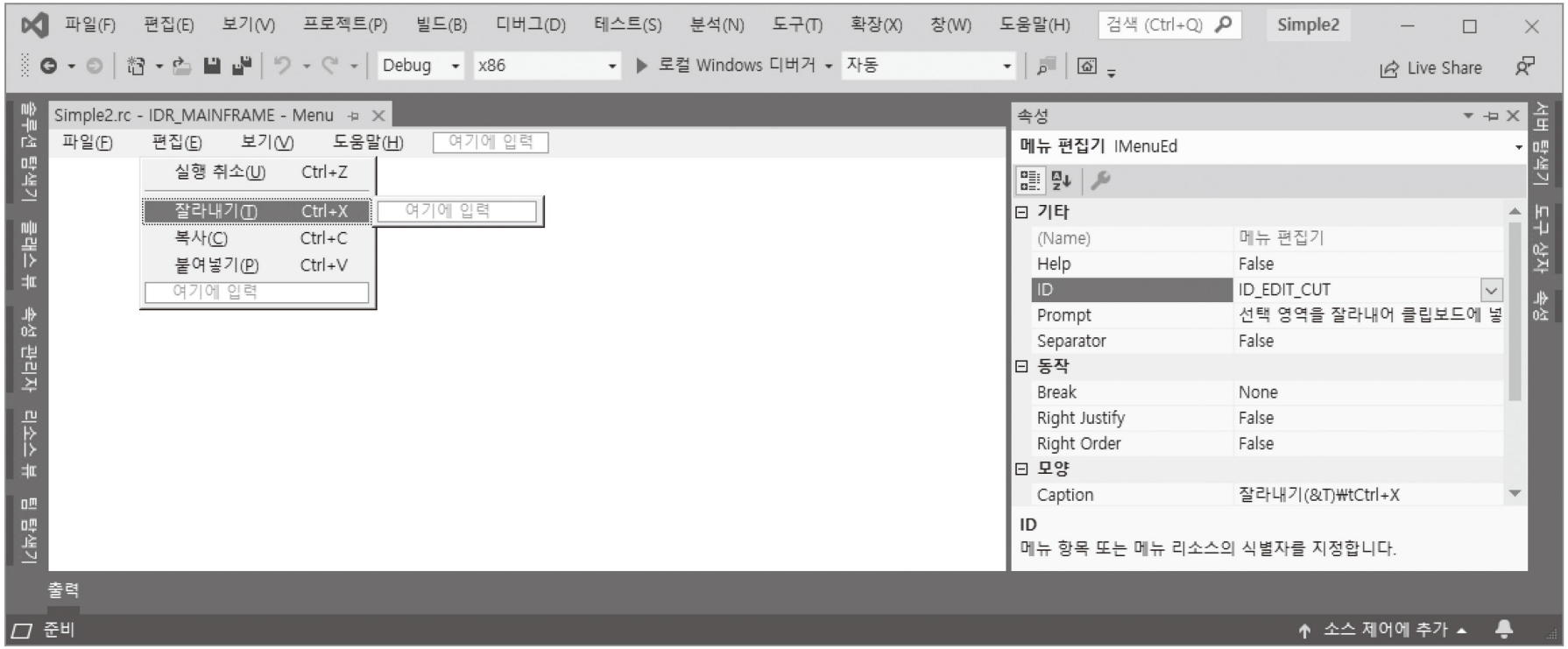
```
BOOL CSimple2App::InitInstance()
{
    ...
    CFrameWnd* pFrame = new CMainFrame;
    if (!pFrame)
        return FALSE;
    m_pMainWnd = pFrame;

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

    pFrame->ShowWindow(SW_SHOW);
    pFrame->UpdateWindow();
    return TRUE;
}
```

리소스를 이용한 메뉴 생성

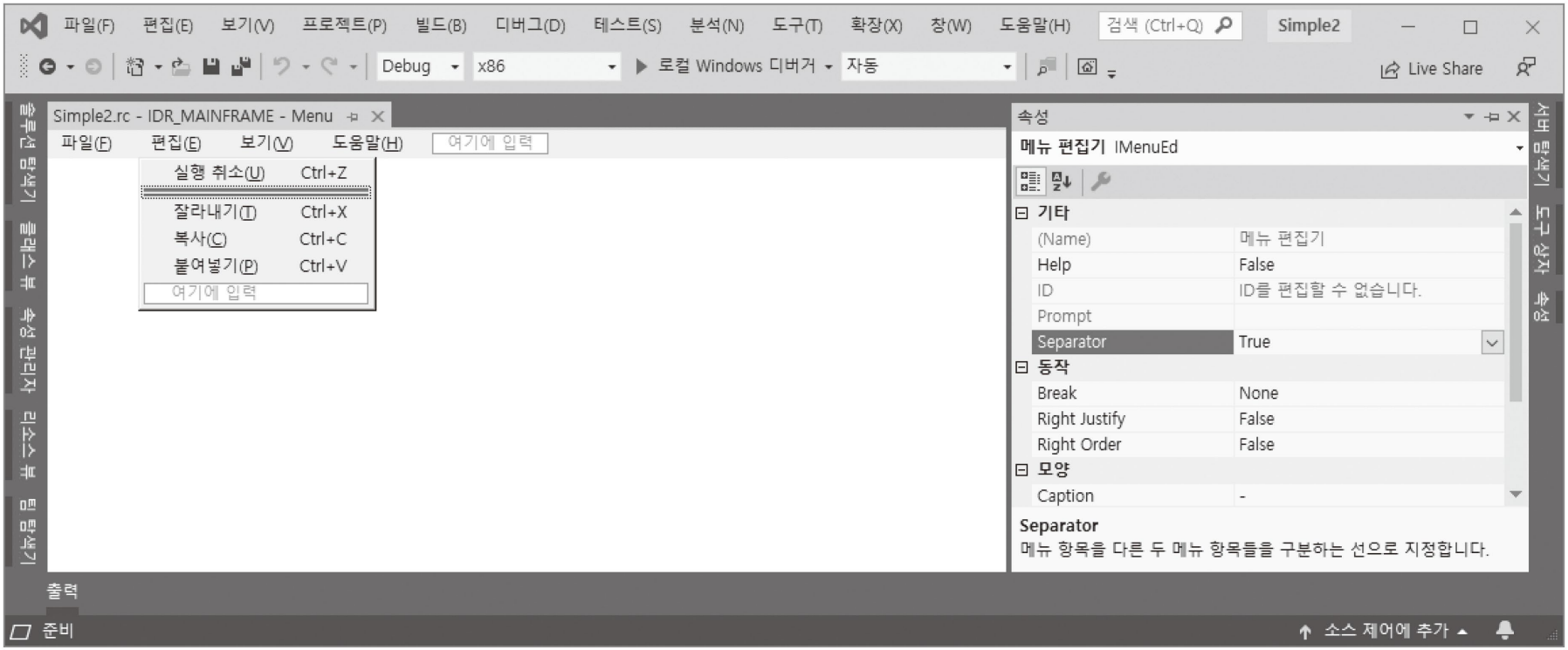
- 응용 프로그램 마법사가 생성한 코드



(a) 메뉴 항목

리소스를 이용한 메뉴 생성

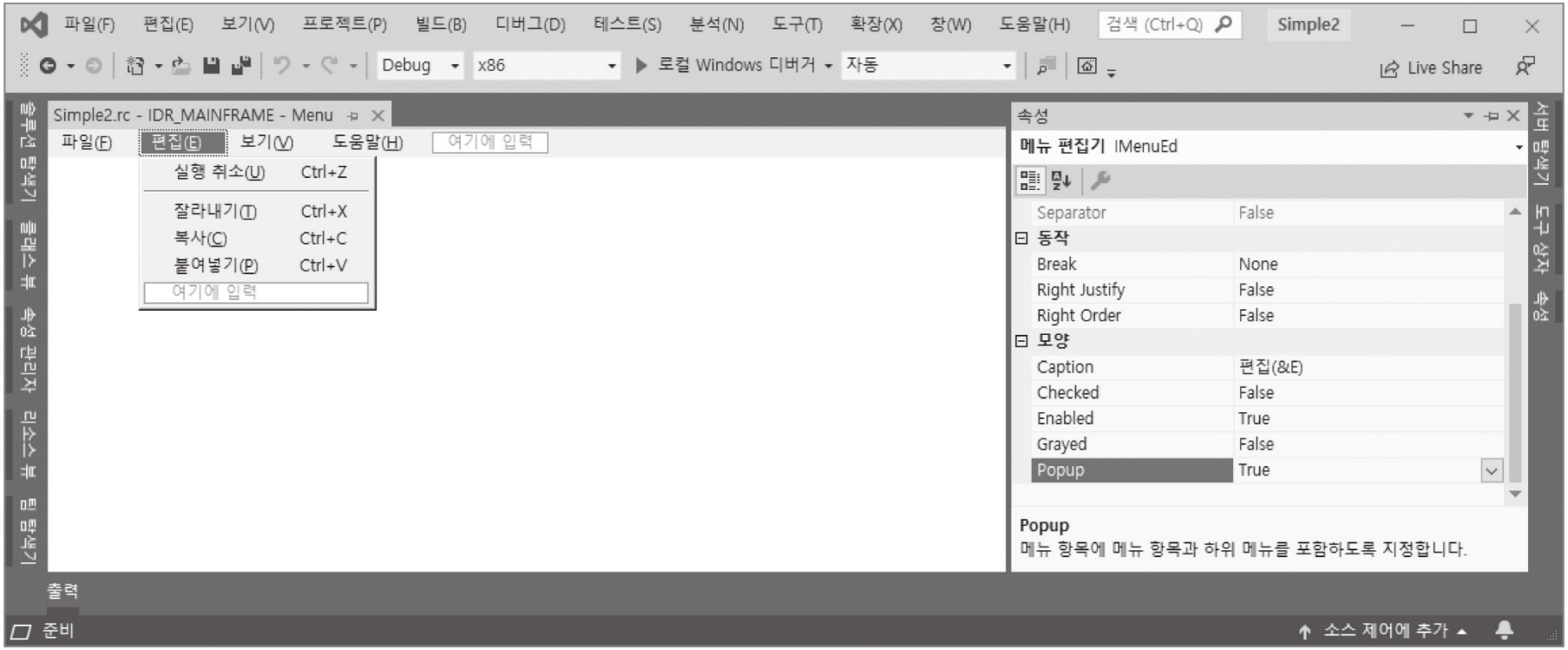
- 메뉴 리소스 편집



(b) 분리자

리소스를 이용한 메뉴 생성

■ 메뉴 리소스 편집



(c) 최상위 메뉴

그림 6-7 메뉴 리소스 편집

리소스를 이용한 메뉴 생성

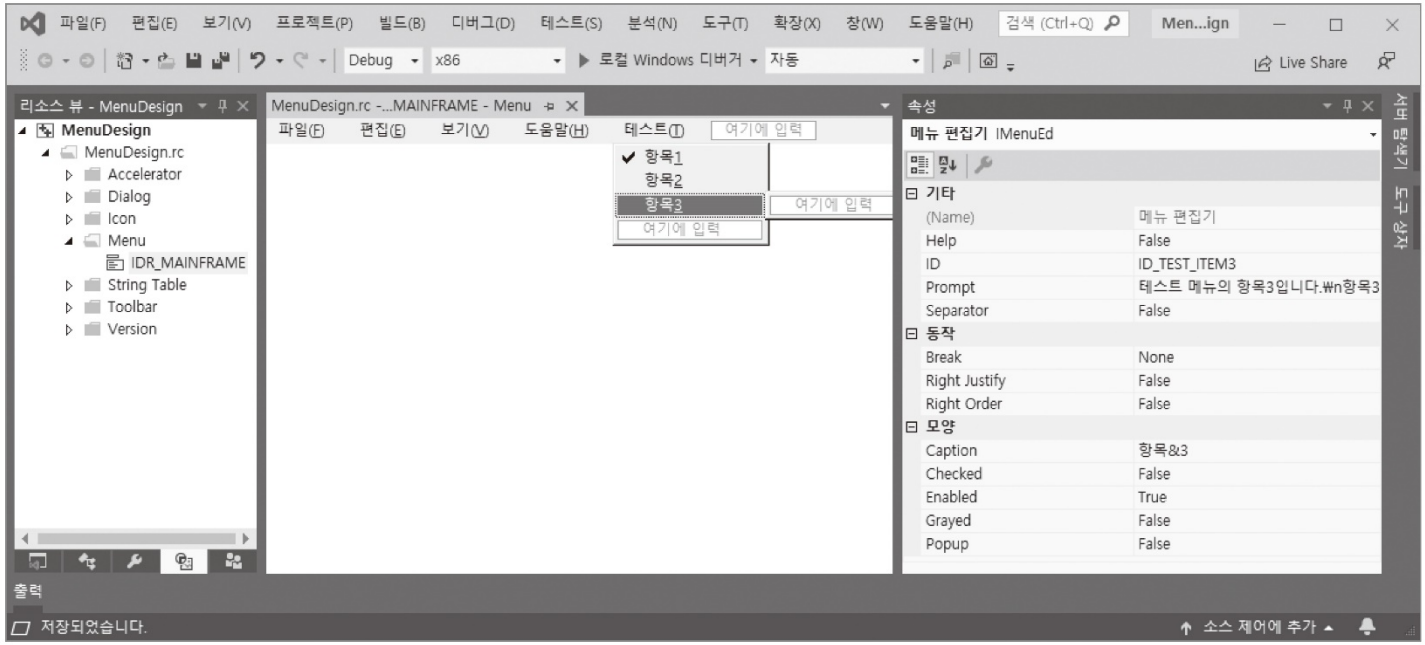
표 6-3 메뉴 항목 속성

속성	의미
Help	최상위 메뉴 항목에만 설정할 수 있다. 이 속성을 설정하면 윈도우의 오른쪽 끝에 메뉴 항목이 표시된다. 요즘은 잘 쓰이지 않으나 과거에 Help 메뉴 항목에 주로 설정했다.
ID	메뉴 항목을 구분하는 번호이다. 일반적으로 ID_메뉴이름_항목이름 형태로 만든다. 예 ID_EDIT_CUT
Prompt	MFC로 작성한 프로그램에서만 사용할 수 있다. 현재 선택된 메뉴 항목에 관한 설명을 상태바나 툴바에 표시하기 위한 속성이다. '\n'을 기준으로 앞쪽 문자열은 상태바에 표시되고, 뒤쪽 문자열은 동일한 기능을 가진 툴바 항목 위에 툴팁(Tooltip)으로 표시된다. 예 선택 영역을 잘라내어 클립보드에 넣습니다.\n잘라내기
Separator	메뉴 항목을 구분하는 가로줄이 표시된다.
Break	일반적으로 메뉴 항목은 하나의 열(Column)에 표시되지만 항목 수가 많으면 두 개 이상의 열에 표시되게 할 수 있다. Break 속성으로 Column 또는 Bar를 선택하면 다음 열에 메뉴 항목이 표시된다. Column과 Bar 속성은 기본적으로 기능이 같지만 Bar 속성을 선택하면 열 구분선(세로줄)이 보인다는 차이가 있다.

리소스를 이용한 메뉴 생성

Right Justify	Help 속성과 기능이 동일하다. 최상위 메뉴 항목에만 설정할 수 있으며, 이 속성을 설정하면 윈도우의 오른쪽 끝에 메뉴 항목이 표시된다.
Right Order	Caption 문자열이 오른쪽에서 왼쪽 방향으로 표시된다. 아랍어나 히브리어를 위한 기능이다.
Caption	화면에 표시되는 문자열이다. 액세스키를 지정하려면 해당 문자 앞에 '&' 기호를 사용한다. 단축키는 '\t' 기호를 삽입하여 단축키를 나타내는 문자열이 탭 위치에 정렬되게 하는 것이 좋다. 예 잘라내기(&T)\tCtrl+X
Checked	메뉴 항목의 왼쪽에 체크 표시를 한다.
Enabled	이 값이 False이면 메뉴 항목이 표시되지만 사용하지는 못한다.
Grayed	메뉴 항목이 흐리게 표시되어 현재 사용할 수 없음을 나타낸다.
Pop-up	이 속성을 설정하면 명령 항목이 아닌 팝업 항목이 된다(즉, 항목 선택 시 하위 메뉴가 펼쳐진다). 최상위 메뉴 항목은 대개 Pop-up 속성을 가진다.

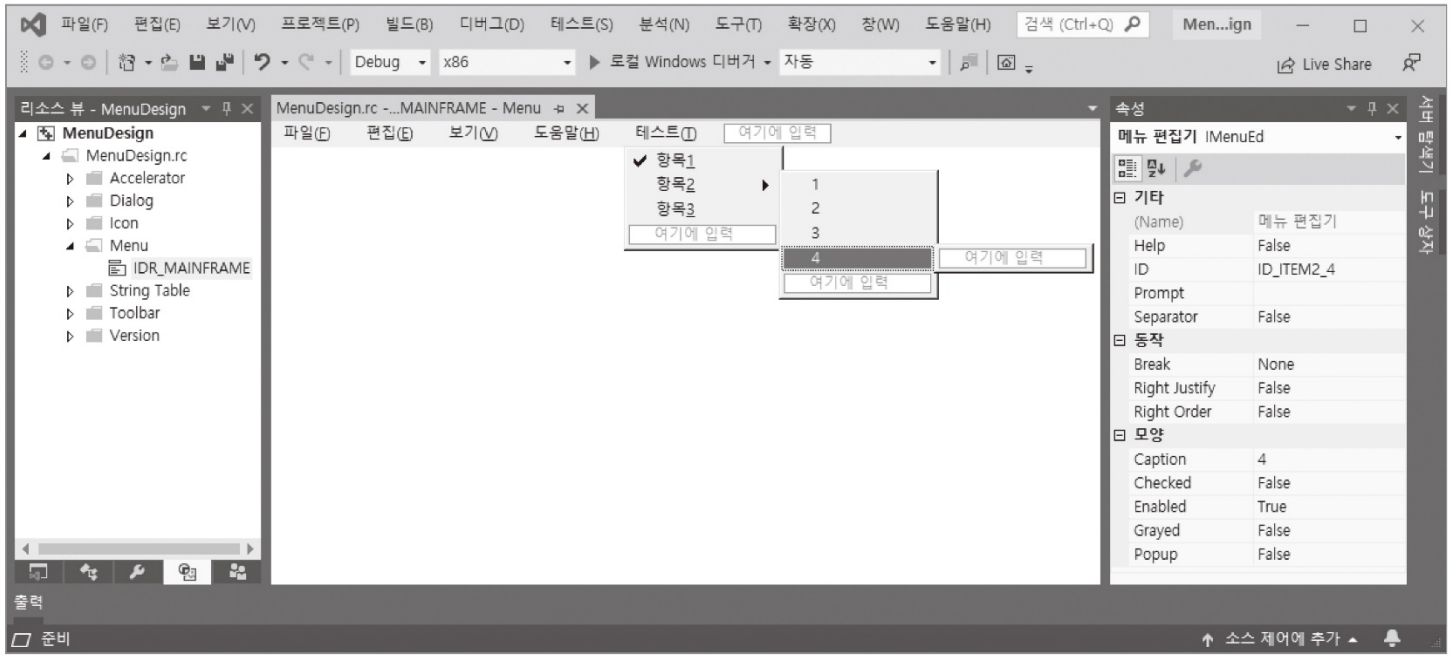
[실습 6-2] 리소스를 이용하여 메뉴 만들기



Caption	ID	나머지 속성
테스트(&T)	없음	변경 없음
항목&1	ID_TEST_ITEM1	Checked 속성 True로 변경
항목&2	없음	Pop-up 속성 True로 변경
항목&3	ID_TEST_ITEM3	Prompt 속성에 '테스트 메뉴의 항목3입니다.\n항목3' 입력

그림 6-8 '테스트' 메뉴 추가와 속성 설정

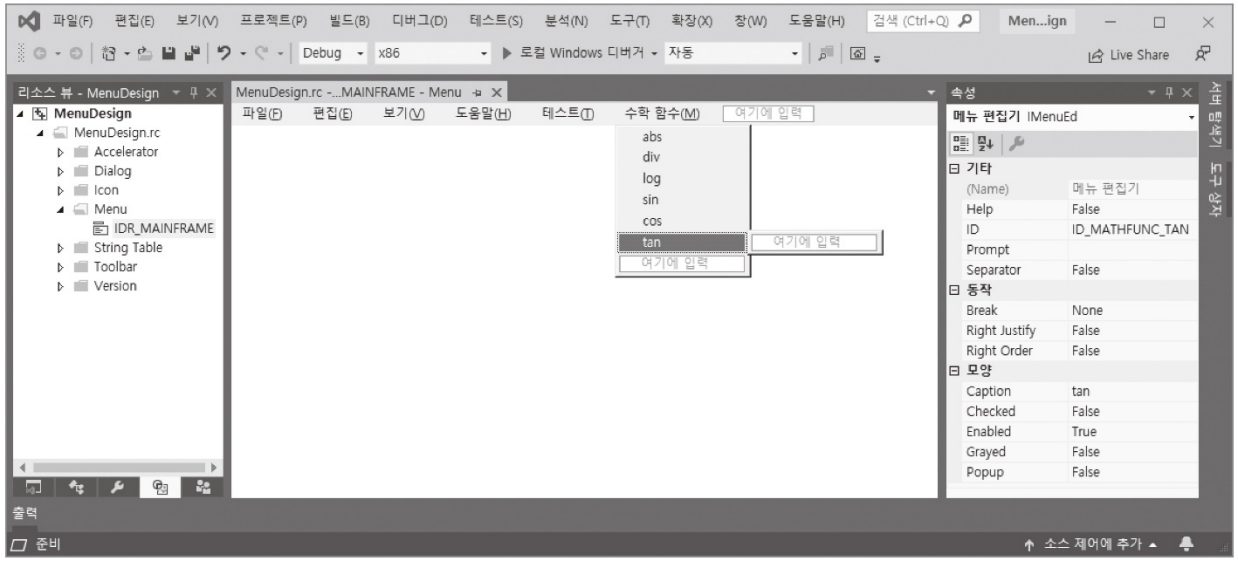
[실습 6-2] 리소스를 이용하여 메뉴 만들기



Caption	ID	나머지 속성
1	ID_ITEM2_1	변경 없음
2	ID_ITEM2_2	변경 없음
3	ID_ITEM2_3	변경 없음
4	ID_ITEM2_4	변경 없음

그림 6-9 '항목2'의 하위 메뉴 추가와 속성 설정

[실습 6-2] 리소스를 이용하여 메뉴 만들기



Caption	ID	나머지 속성
수학 함수(&M)	없음	변경 없음
abs	ID_MATHFUNC_ABS	변경 없음
div	ID_MATHFUNC_DIV	변경 없음
log	ID_MATHFUNC_LOG	변경 없음
sin	ID_MATHFUNC_SIN	Break: Bar 선택
cos	ID_MATHFUNC_COS	변경 없음
tan	ID_MATHFUNC_TAN	변경 없음

그림 6-10 '수학 함수' 메뉴 추가와 속성 설정

[실습 6-2] 리소스를 이용하여 메뉴 만들기

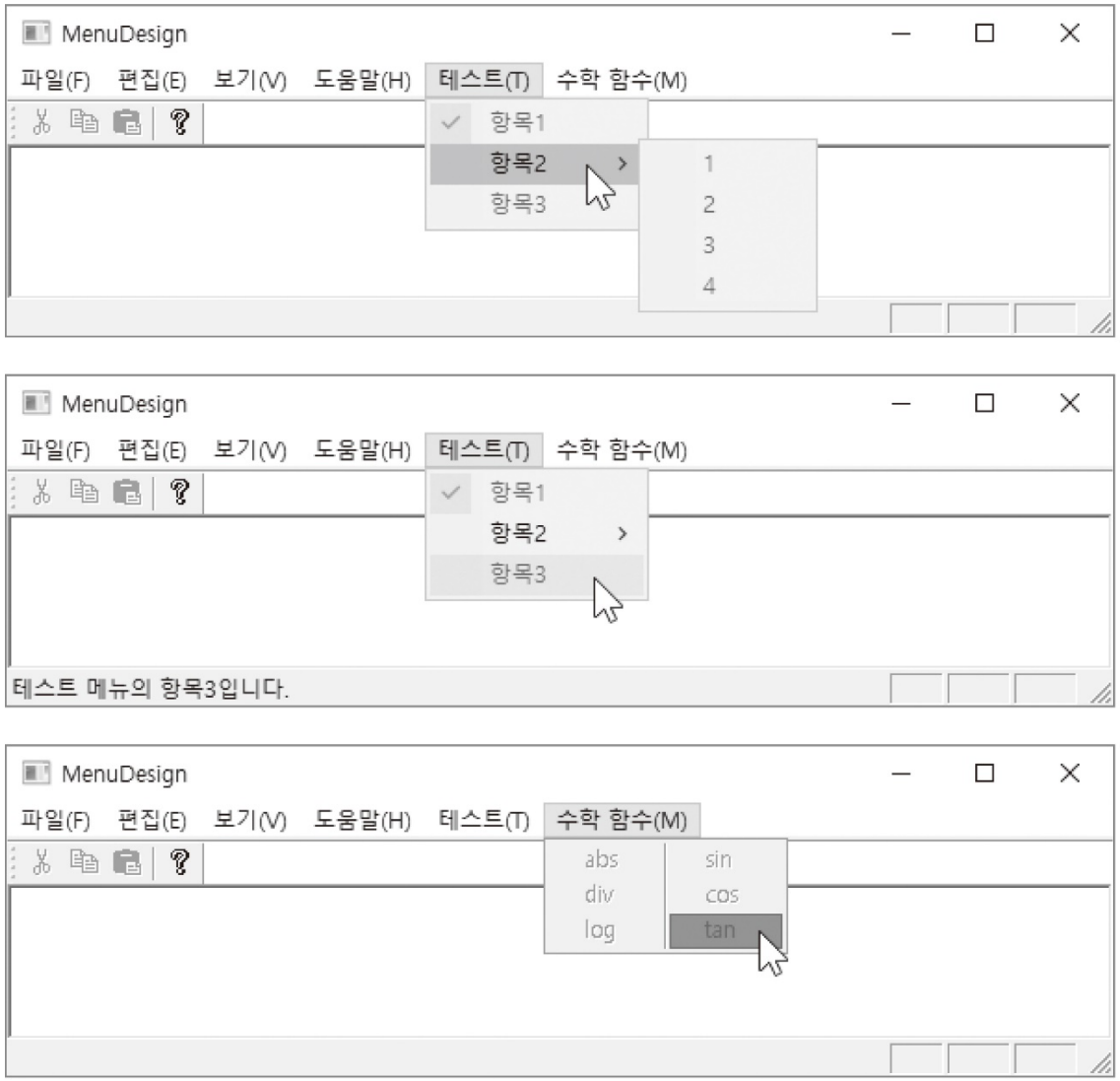


그림 6-11 실행 결과

리소스를 이용한 메뉴 생성

표 6-3 메뉴 항목 속성

속성	의미
Help	최상위 메뉴 항목에만 설정할 수 있다. 이 속성을 설정하면 윈도우의 오른쪽 끝에 메뉴 항목이 표시된다. 요즘은 잘 쓰이지 않으나 과거에 Help 메뉴 항목에 주로 설정했다.
ID	메뉴 항목을 구분하는 번호이다. 일반적으로 ID_메뉴이름_항목이름 형태로 만든다. 예 ID_EDIT_CUT
Prompt	MFC로 작성한 프로그램에서만 사용할 수 있다. 현재 선택된 메뉴 항목에 관한 설명을 상태바나 툴바에 표시하기 위한 속성이다. '\n'을 기준으로 앞쪽 문자열은 상태바에 표시되고, 뒤쪽 문자열은 동일한 기능을 가진 툴바 항목 위에 툴팁(Tooltip)으로 표시된다. 예 선택 영역을 잘라내어 클립보드에 넣습니다.\n잘라내기
Separator	메뉴 항목을 구분하는 가로줄이 표시된다.
Break	일반적으로 메뉴 항목은 하나의 열(Column)에 표시되지만 항목 수가 많으면 두 개 이상의 열에 표시되게 할 수 있다. Break 속성으로 Column 또는 Bar를 선택하면 다음 열에 메뉴 항목이 표시된다. Column과 Bar 속성은 기본적으로 기능이 같지만 Bar 속성을 선택하면 열 구분선(세로줄)이 보인다는 차이가 있다.
Right Justify	Help 속성과 기능이 동일하다. 최상위 메뉴 항목에만 설정할 수 있으며, 이 속성을 설정하면 윈도우의 오른쪽 끝에 메뉴 항목이 표시된다.

코드를 이용한 메뉴 생성

- MFC 클래스 계층도

그림 6-12 MFC 클래스 계층도

코드를 이용한 메뉴 생성

- 메뉴 전체를 만드는 방법

그림 6-13 실행 화면

코드를 이용한 메뉴 생성

- 메뉴 전체를 만드는 방법

```
int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    ...
    CMenu menuMain; // 메뉴 객체 생성
    menuMain.CreateMenu(); // 최상위 메뉴 생성

    CMenu menuPopup; // 메뉴 객체 생성
    menuPopup.CreatePopupMenu(); // 팝업 메뉴 생성

    // 팝업 메뉴에 메뉴 항목 세 개 추가
    menuPopup.AppendMenu(MF_STRING, 201, _T("빨간색(&R)"));
    menuPopup.AppendMenu(MF_STRING, 202, _T("초록색(&G)"));
    menuPopup.AppendMenu(MF_STRING, 203, _T("파란색(&B)"));
```

코드를 이용한 메뉴 생성

- 메뉴 전체를 만드는 방법

```
    // 최상위 메뉴에 팝업 메뉴 추가
    menuMain.AppendMenu(MF_POPUP, (UINT)menuPopup.Detach(), _T("색상(&C)"));
    SetMenu(&menuMain); // 메뉴를 윈도우에 붙임

    menuMain.Detach(); // 메뉴 객체와 메뉴를 분리

    return 0;
}
```

- CMenu::CreateMenu( ) : 비어 있는 최상위 메뉴를 만들고 메뉴 객체와 연결
- CMenu::CreatePopupMenu( ) : 비어 있는 팝업 메뉴를 만들고 메뉴 객체와 연결
- CMenu::AppendMenu( ) : 새로운 메뉴 항목을 메뉴에 추가

코드를 이용한 메뉴 생성

■ 메뉴 전체를 만드는 방법

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

표 6-4 AppendMenu() 함수의 인자

❶ nFlags		❷ nIDNewItem	❸ lpszNewItem
MF_STRING	메뉴 항목이 문자열이다.	새로운 메뉴 항목의 ID	메뉴 항목의 캡션 문자열
MF_POPUP	메뉴 항목이 팝업 메뉴를 가진다.	팝업 메뉴를 가리키는 핸들 (HWND 타입)	메뉴 항목의 캡션 문자열
MF_CHECKED MF_UNCHECKED MF_ENABLED MF_DIABLED MF_GRAYED	메뉴 항목의 속성을 나타내는 값이다. [표 6-3]에서 Checked, Enabled, Grayed 속성을 설정하거나 해제한 것과 의미가 동일하다.		
MF_SEPARATOR	[표 6-3]에서 Separator 속성을 설정한 것과 의미가 동일하다.	사용되지 않음	사용되지 않음

코드를 이용한 메뉴 생성

- 기존 메뉴에 추가하는 방법

```
int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    ...
    // (1) '항목2'의 하위 메뉴를 생성한다.
    CMenu Popup1; // 메뉴 객체 생성
    Popup1.CreatePopupMenu(); // 팝업 메뉴 생성
    Popup1.AppendMenu(MF_STRING, 301, _T("1"));
    Popup1.AppendMenu(MF_STRING, 302, _T("2"));
    Popup1.AppendMenu(MF_STRING, 303, _T("3"));
    Popup1.AppendMenu(MF_STRING, 304, _T("4"));
```

코드를 이용한 메뉴 생성

- 기존 메뉴에 추가하는 방법

```
    // (2) '테스트' 메뉴를 생성한다.
    CMenu Popup2; // 메뉴 객체 생성
    Popup2.CreatePopupMenu(); // 팝업 메뉴 생성
    Popup2.AppendMenu(MF_STRING|MF_CHECKED, 201, _T("항목&1"));
    Popup2.AppendMenu(MF_POPUP, (UINT)Popup1.Detach(), _T("항목&2"));
    Popup2.AppendMenu(MF_STRING, 203, _T("항목&3"));

    // (3) '테스트' 메뉴를 최상위 메뉴에 붙인다.
    CMenu *pTopLevel = GetMenu(); // 최상위 메뉴의 포인터를 얻는다.
    pTopLevel->AppendMenu(MF_POPUP, (UINT)Popup2.Detach(), _T("테스트(&T)"));

    return 0;
}
```

메뉴 명령 처리

- 메뉴 명령 처리 과정
  - 명령 항목을 마우스나 키보드로 선택
  - WM_COMMAND 메시지 발생
  - WM_COMMAND 메시지 핸들러에서 메뉴 명령 처리
- MFC에서는 각각의 명령 항목별로 처리 함수(명령 핸들러)를 따로 작성할 수 있기 해줌
- 명령 라우팅
  - 명령 핸들러를 작성하는 위치에 관계없이 처리됨

[실습 6-3] 메뉴 명령 처리하기

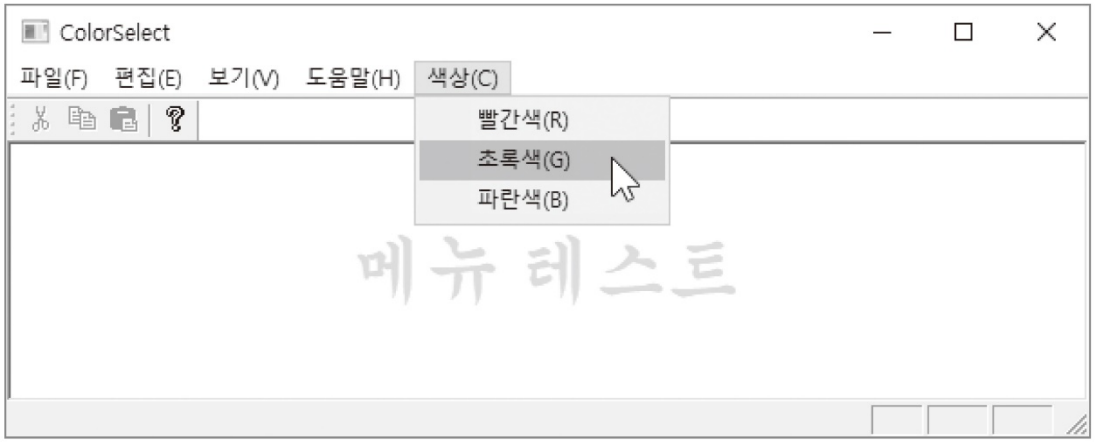


그림 6-14 실행 결과

[실습 6-3] 메뉴 명령 처리하기

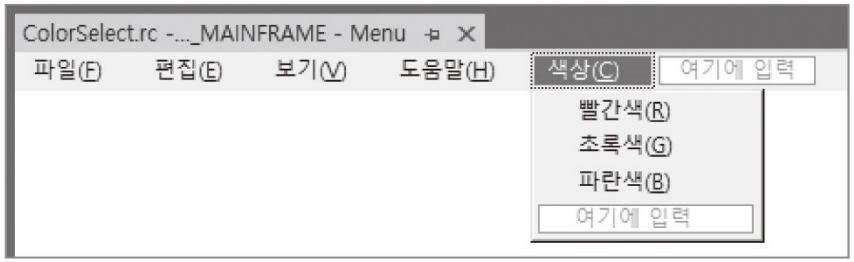


그림 6-15 '색상' 메뉴 추가와 속성 변경

Caption	ID	나머지 속성
색상(&C)	없음	변경 없음
빨간색(&R)	ID_COLOR_RED	변경 없음
초록색(&G)	ID_COLOR_GREEN	변경 없음
파란색(&B)	ID_COLOR_BLUE	변경 없음

[실습 6-3] 메뉴 명령 처리하기

```
class CChildView : public CWnd
{
// 생성입니다.
public:
   CChildView();
// 특성입니다.
public:
   COLORREF m_color;
CChildView::CChildView()
{
   m_color = RGB(255, 0, 0);
}
```

[실습 6-3] 메뉴 명령 처리하기

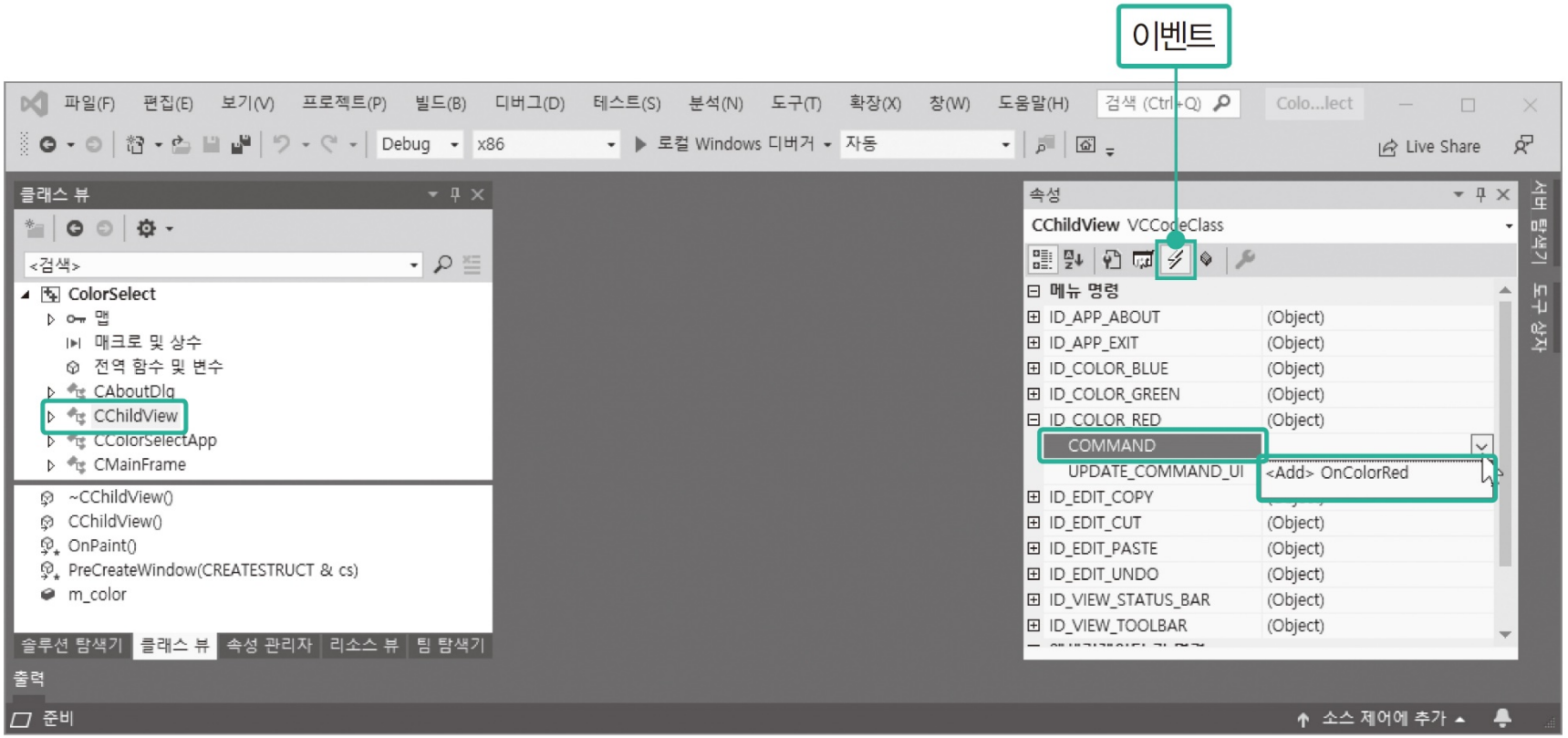


그림 6-16 메뉴 명령 핸들러 추가

[실습 6-3] 메뉴 명령 처리하기

```
void CChildView::OnColorRed()
{
   m_color = RGB(255, 0, 0);
   Invalidate();
}

void CChildView::OnColorGreen()
{
   m_color = RGB(0, 255, 0);
   Invalidate();
}

void CChildView::OnColorBlue()
{
   m_color = RGB(0, 0, 255);
   Invalidate();
}
```

[실습 6-3] 메뉴 명령 처리하기

```
void CChildView::OnPaint()
{
   CPaintDC dc(this);

   CFont font;
   font.CreatePointFont(300, _T("궁서"));
   dc.SelectObject(&font);
   dc.SetTextColor(m_color);

   CRect rect;
   GetClientRect(&rect);
   CString str = _T("메뉴 테스트");
   dc.DrawText(str, &rect, DT_CENTER|DT_VCENTER|DT_SINGLELINE);
}
```

메뉴 항목 갱신

- 메뉴 항목 갱신의 예

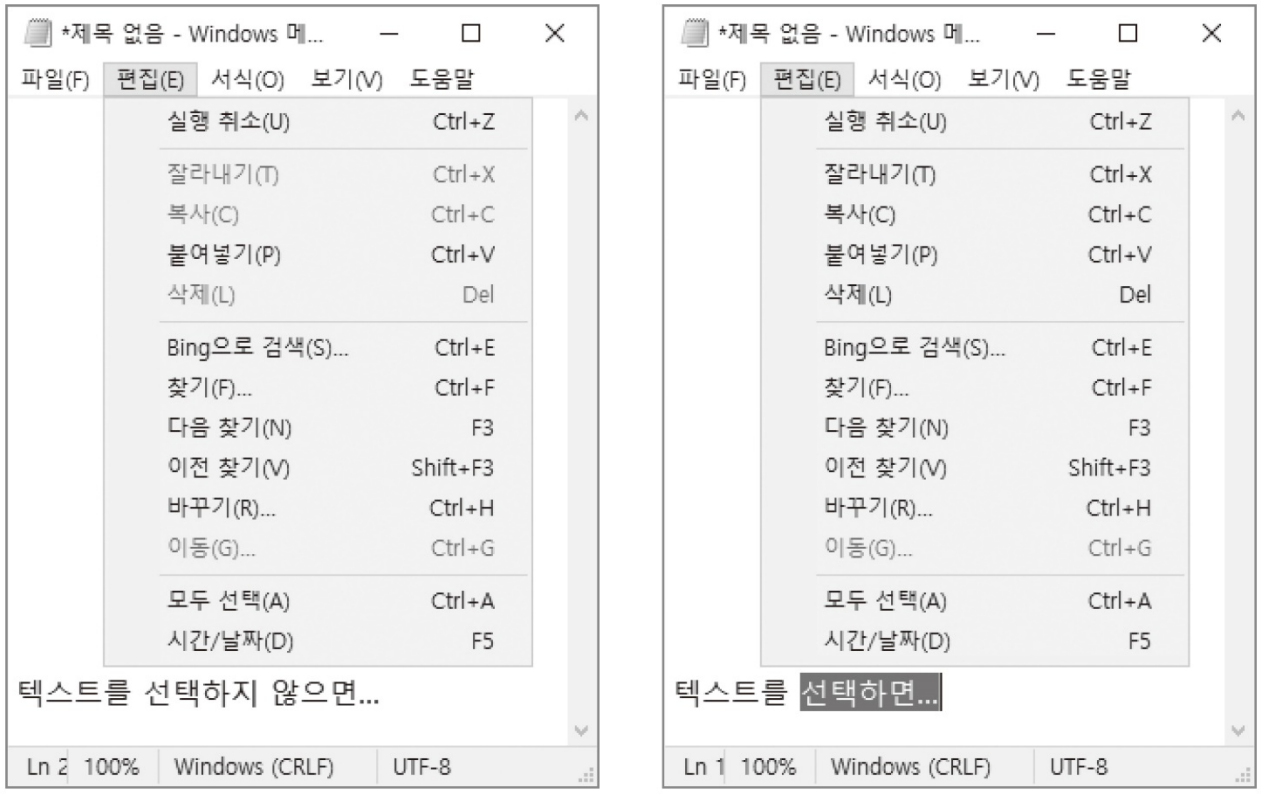


그림 6-17 메뉴 항목 갱신의 예

메뉴 항목 갱신

- 명령 갱신 핸들러
  - 명령 핸들러와 마찬가지로 각 메뉴 항목의 상태를 갱신하는 함수(명령 갱신 핸들러)를 정의함
  - 명령 갱신 핸들러 작성시, 메뉴가 열리기 전에 MFC에서 자동으로 명령 갱신 핸들러를 호출하여 메뉴 항목의 상태를 바꾸어 줌

- 명령 갱신 핸들러 작성

```
void CChildView::OnUpdateColorRed(CCmdUI* pCmdUI)
{
}
```

메뉴 항목 갱신

표 6-5 CCmdUI 클래스 멤버 함수

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

[실습 6-4] 메뉴 항목 갱신하기

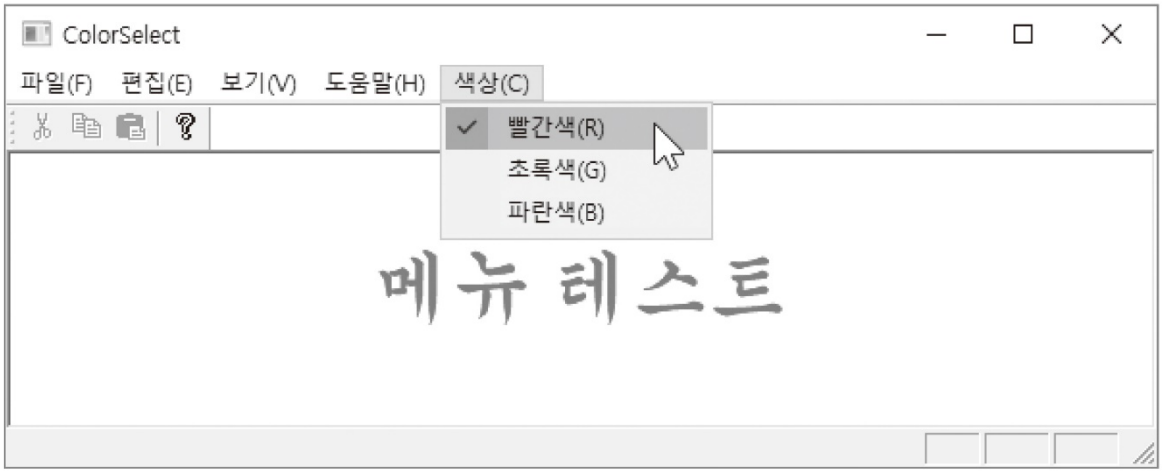


그림 6-18 실행 결과

[실습 6-4] 메뉴 항목 갱신하기

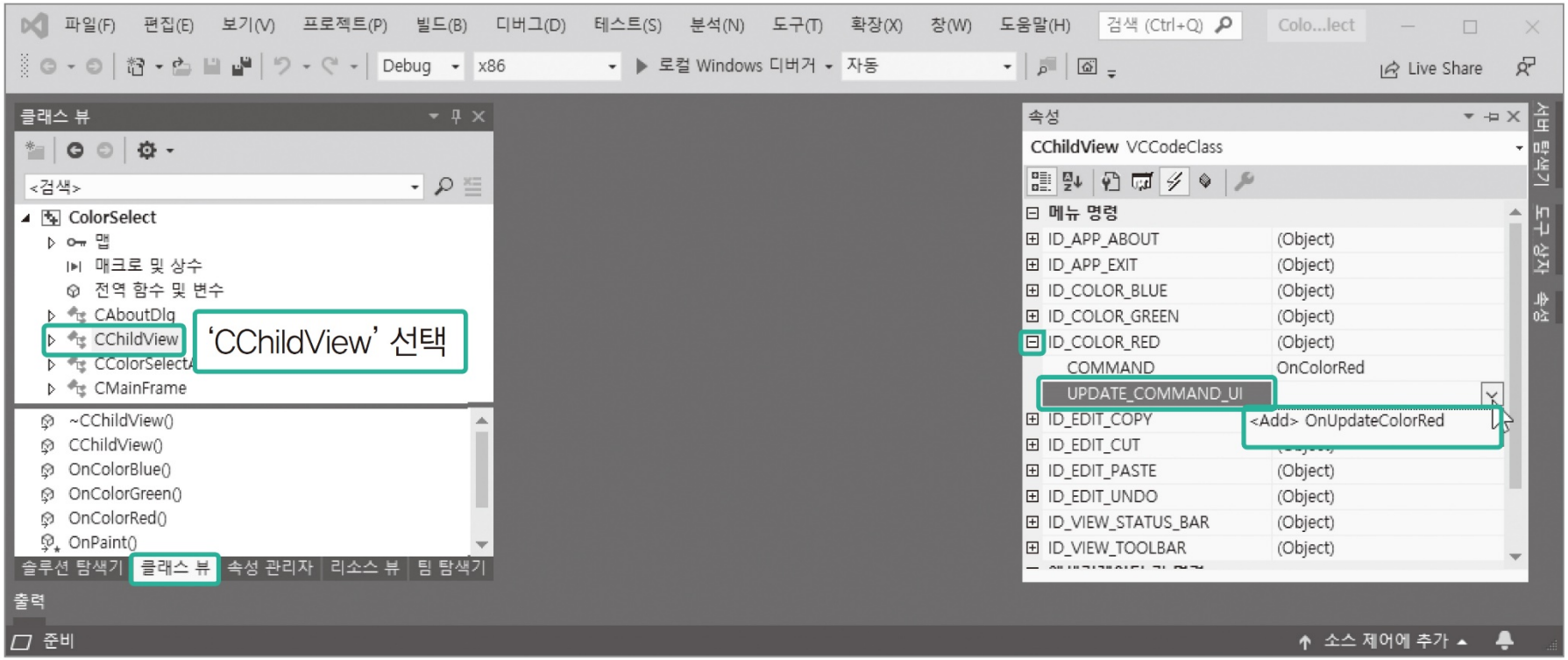


그림 6-19 메뉴 명령 갱신 핸들러 추가

[실습 6-4] 메뉴 항목 갱신하기

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

void CChildView::OnUpdateColorGreen(CCmdUI *pCmdUI)
{
   pCmdUI->SetCheck(m_color == RGB(0, 255, 0));
}

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

컨텍스트 메뉴

- WM_CONTEXTMENU 메시지 발생 상황
  - 클라이언트 영역이나 비클라이언트 영역에서 마우스 오른쪽 버튼을 클릭한 경우
  - [Shift] + [F10] 키를 누른 경우
  - 가상 키코드 VK_APPS에 해당하는 키를 누른 경우

- WM_CONTEXTMENU 메시지 핸들러

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

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

컨텍스트 메뉴

CMenu::TrackPopupMenu() 함수

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

- nFlags
  - TPM_LEFTALIGN, TPM_CENTERALIGN, TPM_RIGHTALIGN

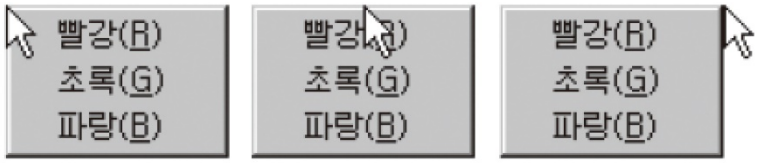


그림 6-20 TPM_*ALIGN 값에 따른 컨텍스트 메뉴의 위치

  - TPM_LEFTBUTTON, TPM_RIGHTBUTTON
- x, y : 컨텍스트 메뉴가 표시될 위치(스크린 좌표)
- pWnd : 컨텍스트 메뉴에서 발생한 WM_COMMAND 메시지를 처리할 윈도우
- lpRect : 스크린 좌표로 정의된 직사각형 좌표

컨텍스트 메뉴

- 컨텍스트 메뉴 구현

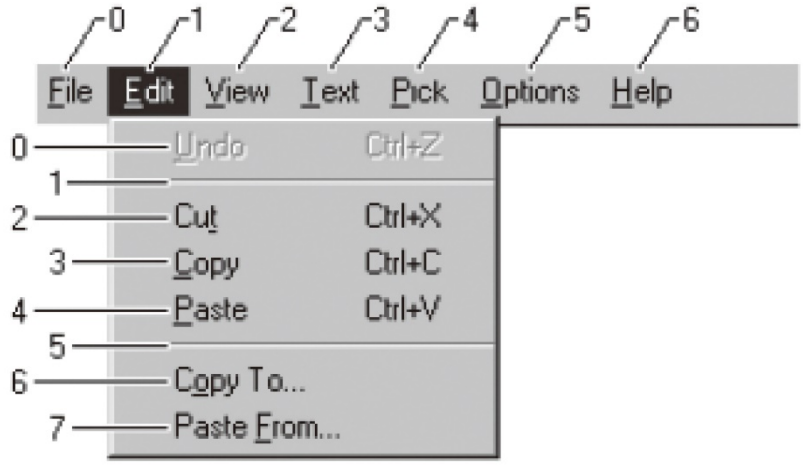


그림 6-21 메뉴 항목의 위치

```
CMenu menu; // 메뉴 객체를 생성한다.
menu.LoadMenu(IDR_MAINFRAME); // 메뉴 리소스를 로드한다.
CMenu* pMenu = menu.GetSubMenu(1); // [Edit] 메뉴를 가리키는 CMenu 포인터를 얻는다.
pMenu->TrackPopupMenu(...); // [Edit] 메뉴를 컨텍스트 메뉴로 표시한다.
```

[실습 6-5] 컨텍스트 메뉴 구현하기

그림 6-22 실행 결과

[실습 6-5] 컨텍스트 메뉴 구현하기

```
void CChildView::OnContextMenu(CWnd* /*pWnd*/, CPoint point)
{
   CMenu menu;
   menu.LoadMenu(IDR_MAINFRAME);
   CMenu* pMenu = menu.GetSubMenu(4);
   pMenu->TrackPopupMenu(
      TPM_LEFTALIGN | TPM_RIGHTBUTTON,
      point.x, point.y, AfxGetMainWnd());
}
```

시스템 메뉴

- 시스템 메뉴 조작하기
  - CWnd::GetSystemMenu() 함수를 이용하여 시스템 메뉴를 가리키는 CMenu 포인터를 얻은 후, CMenu 클래스가 제공하는 다양한 멤버 함수(AppendMenu(), InsertMenu(), DeleteMenu(), ...)를 적용
- 시스템 메뉴 조작 시 주의 사항
  - 시스템 메뉴를 변경하려면 GetSystemMenu(FALSE) 함수를 호출, 시스템 메뉴를 초기 상태로 되돌리려면 GetSystemMenu(TRUE) 함수를 호출
  - 시스템 메뉴에 새로운 메뉴 항목을 추가할 때 메뉴 ID는 반드시 16의 정수배여야 하고 0xF000보다 작은 값이어야 함
  - 시스템 메뉴 항목을 선택하면 WM_COMMAND가 아닌 WM_SYSCOMMAND 메시지가 발생

[실습 6-6] 시스템 메뉴 변경하기

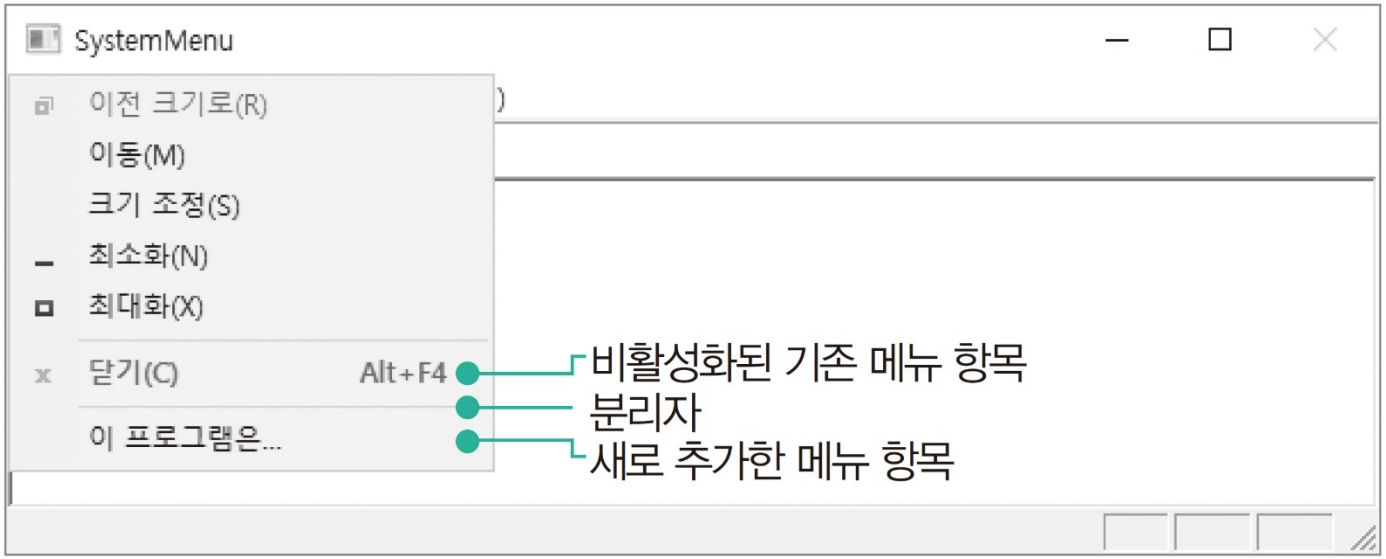


그림 6-23 실행 결과

[실습 6-6] 시스템 메뉴 변경하기

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

   CMenu* pSysMenu = GetSystemMenu(FALSE);
   // '닫기' 메뉴 항목을 비활성화한다.
   pSysMenu->EnableMenuItem(SC_CLOSE, MF_GRAYED);
   // 분리자(Separator) 항목을 추가한다.
   pSysMenu->AppendMenu(MF_SEPARATOR);
   // ID가 16인 메뉴 항목을 추가한다.
   pSysMenu->AppendMenu(MF_STRING, 16, _T("이 프로그램은..."));

   return 0;
}
```

[실습 6-6] 시스템 메뉴 변경하기

```
void CMainFrame::OnSysCommand(UINT nID, LPARAM lParam)
{
   if ((nID & 0xFFF0) == 16) {
      MessageBox(_T("시스템 메뉴를 테스트합니다."), _T("이 프로그램은..."));
      return;
   }
   CFrameWnd::OnSysCommand(nID, lParam);
}
```

가속기

- 가속기 = 단축키
  - 메뉴 항목을 곧바로 실행할 수 있는 키 조합

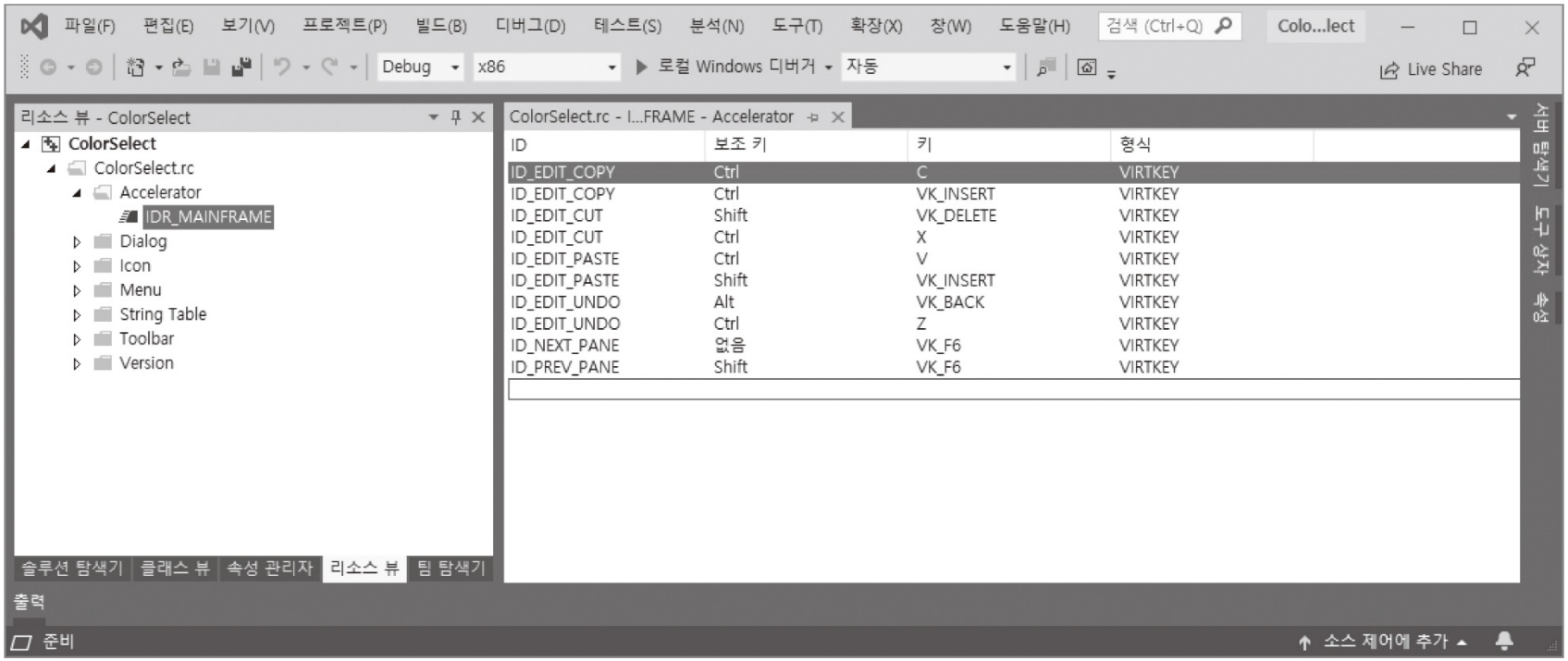


그림 6-24 가속기 리소스

가속기

■ 가속기 리소스 추가

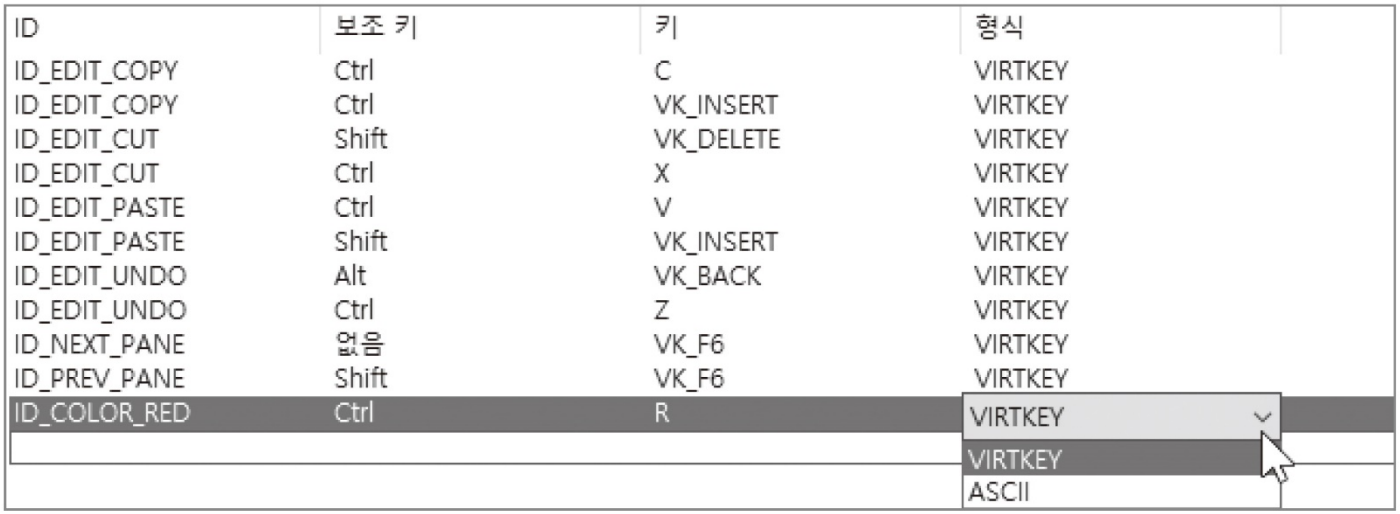


그림 6-25 가속기 리소스 추가

가속기

- 메뉴 항목에 가속기 표시

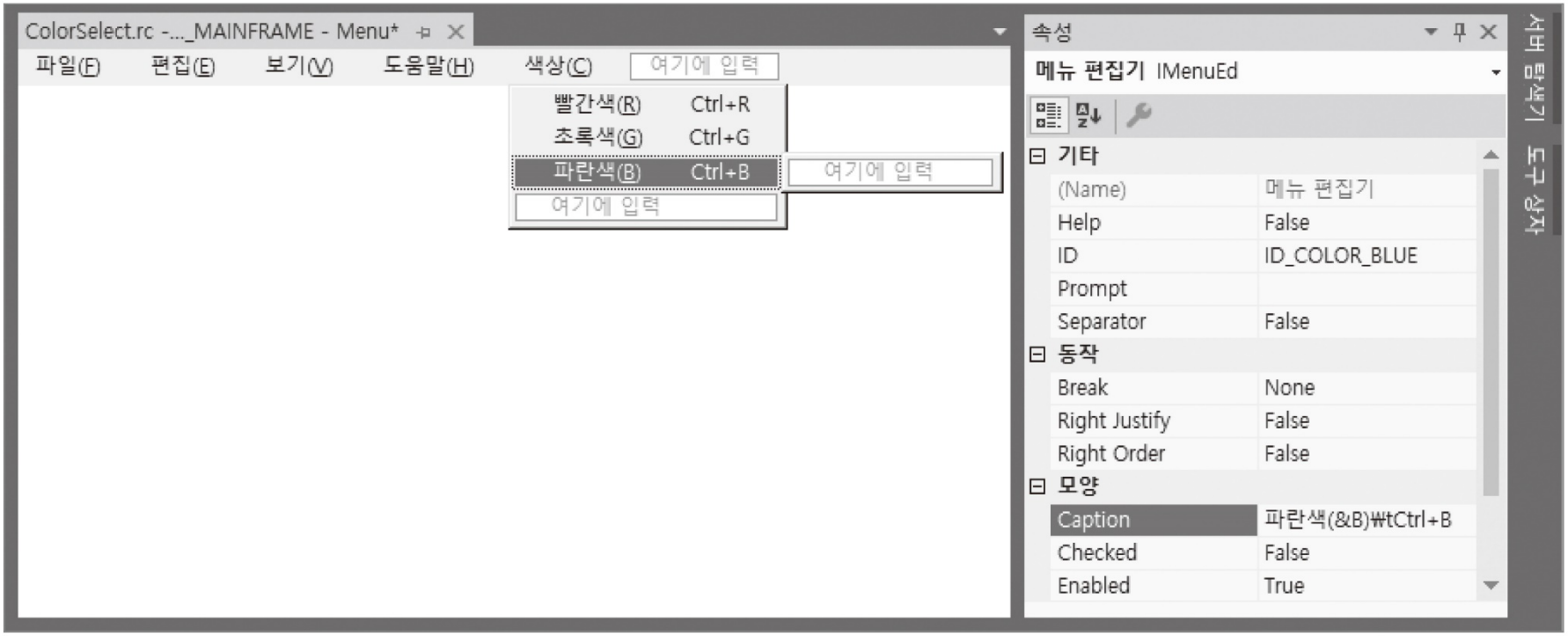


그림 6-26 메뉴 항목에 가속기 표시하기

툴바

- 툴바

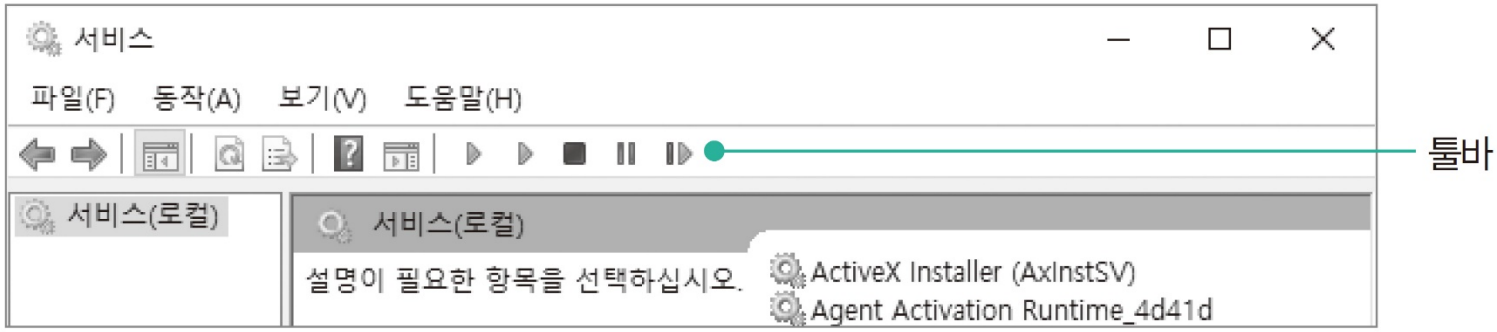


그림 6-27 툴바

- MFC 클래스 계층도

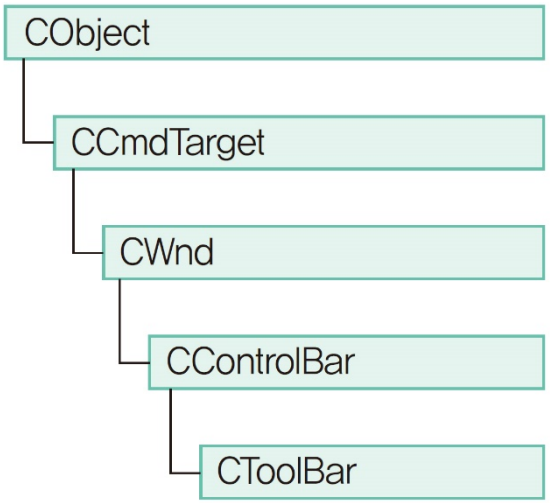


그림 6-28 MFC 클래스 계층도

툴바 생성

툴바 리소스 편집

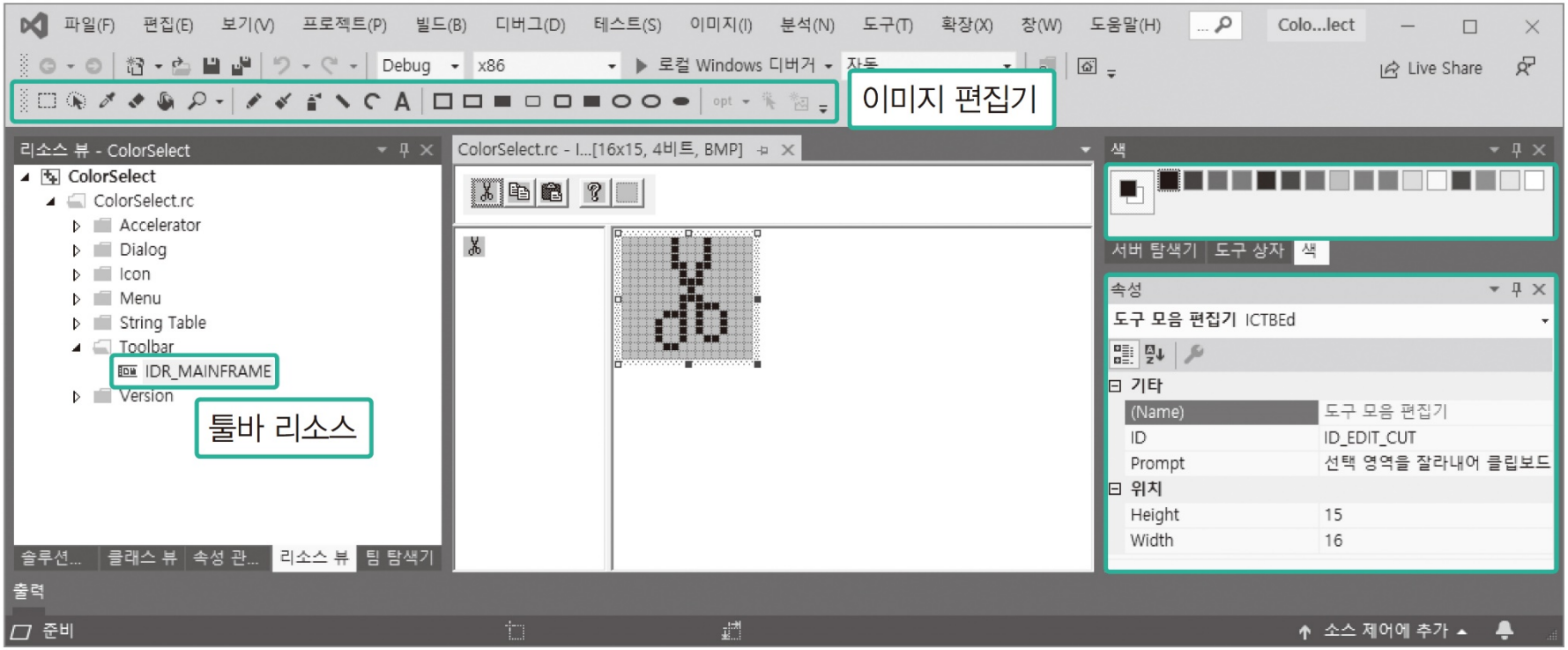


그림 6-29 툴바 리소스 편집

툴바 생성

- 툴바 리소스 속성
  - ID : 메뉴 항목과 같은 ID를 주거나 새로운 ID를 부여
  - Prompt : 메뉴 항목의 Prompt 속성처럼 여기에 입력한 내용이 상태바와 툴팁에 표시
  - Height, Width : 툴바 버튼의 크기를 폭과 높이로 나타내는데, 버튼 한 개의 값을 변경하면 전체 툴바의 크기가 바뀜

툴바 생성

툴바 리소스 편집 요령

- 추가 : 빈칸을 클릭해서 새로운 버튼을 편집함. 편집하는 버튼 오른쪽에는 새로운 빈칸이 자동으로 생김
- 이동 : 마우스로 끌어서 원하는 곳에 떨어뜨림
- 간격 조정 : 마우스로 끌어서 붙어 있는 버튼을 떼면 자동으로 간격이 생김. 프로그램을 실행하면 버튼 사이에 세로줄이 표시됨

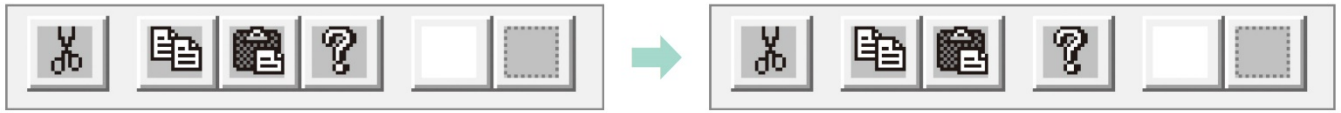


그림 6-30 툴바 버튼 간격 조정

- 삭제 : 마우스로 끌어서 툴바 바깥쪽에 떨어뜨림. Delete를 사용하면 버튼의 그림은 지워지지만 버튼 자체는 삭제되지 않음

툴바 생성

- 툴바 코드

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

툴바 생성

- 툴바 코드

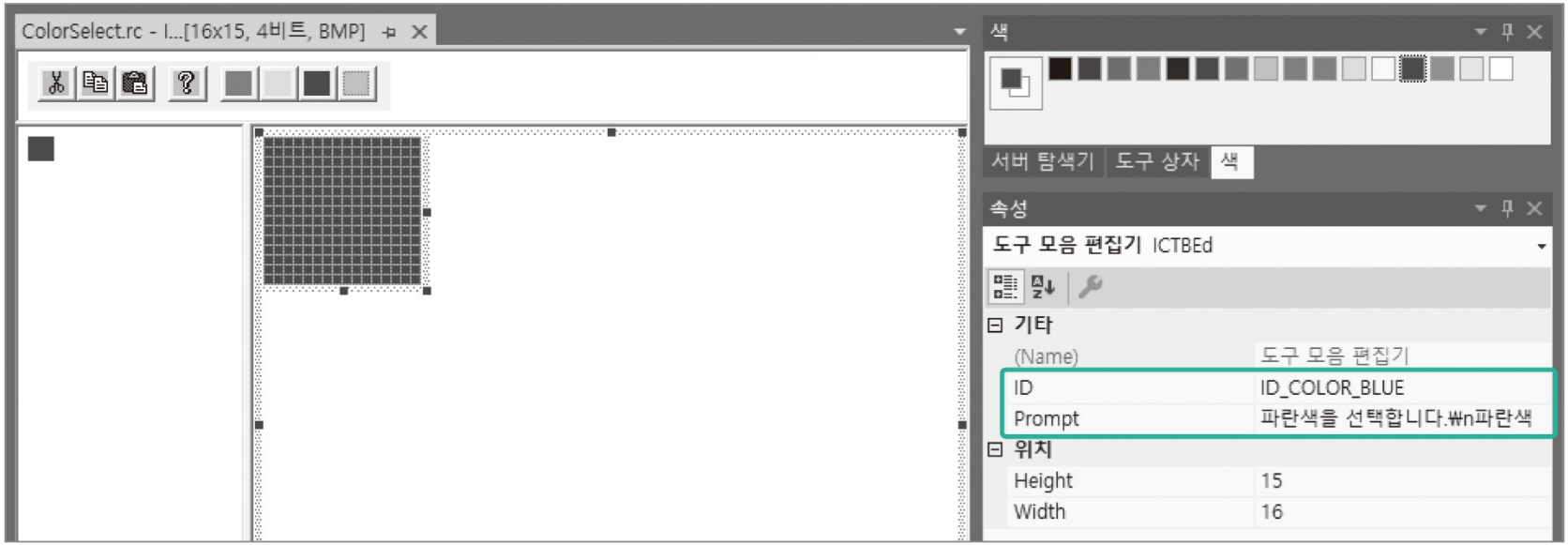
```
int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    ...
    if (!m_wndToolBar.CreateEx(this, TBSTYLE_FLAT,
          WS_CHILD | WS_VISIBLE | CBRS_TOP | CBRS_GRIPPER |
          CBRS_TOOLTIPS | CBRS_FLYBY | CBRS_SIZE_DYNAMIC) ||
       !m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
    {
       TRACE0("도구 모음을 만들지 못했습니다.\n");
       return -1;
    }
    ...
    m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
    EnableDocking(CBRS_ALIGN_ANY);
    DockControlBar(&m_wndToolBar);

    return 0;
}
```

[실습 6-7] 툴바 구현하기

그림 6-31 실행 결과

[실습 6-7] 툴바 구현하기



ID	Prompt
ID_COLOR_RED	빨간색을 선택합니다.\n빨간색
ID_COLOR_GREEN	초록색을 선택합니다.\n초록색
ID_COLOR_BLUE	파란색을 선택합니다.\n파란색

그림 6-32 툴바 버튼 추가, 속성 변경

상태바

상태바

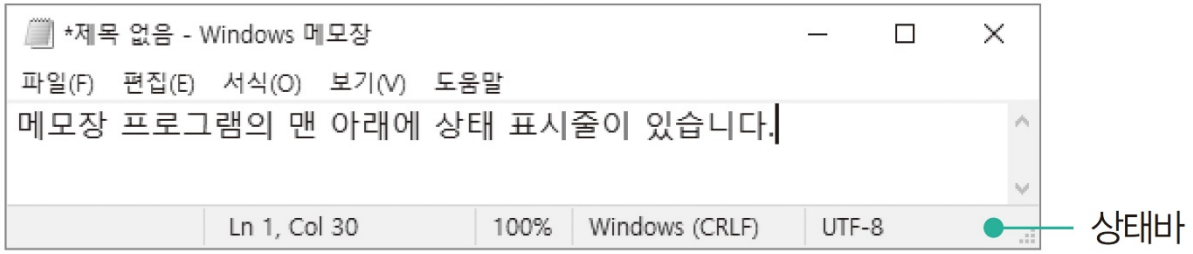


그림 6-33 상태바

MFC 클래스 계층도

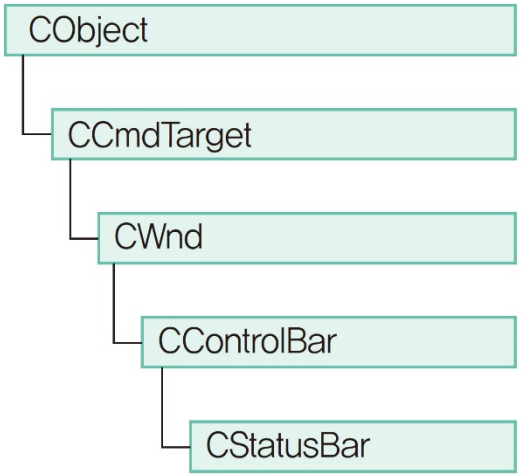


그림 6-34 MFC 클래스 계층도

상태바 생성

그림 6-35 Simple2 프로그램의 상태바 구성

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

상태바 생성

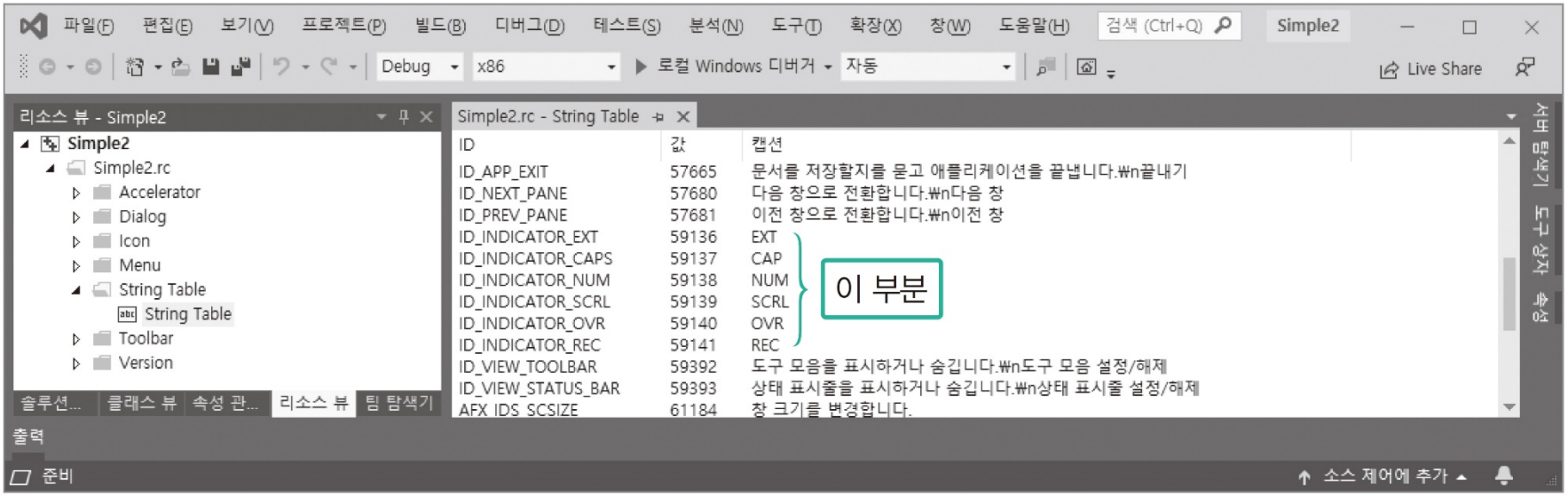


그림 6-36 상태바에 사용되는 문자열 리소스

상태바 생성

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

```
int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
	...
	if (!m_wndStatusBar.Create(this))
	{
		TRACE0("상태 표시줄을 만들지 못했습니다.\n");
		return -1;
	}
	m_wndStatusBar.SetIndicators(indicators, sizeof(indicators)/sizeof(UINT));
	...
}
```

[실습 6-8] 상태바 구현하기

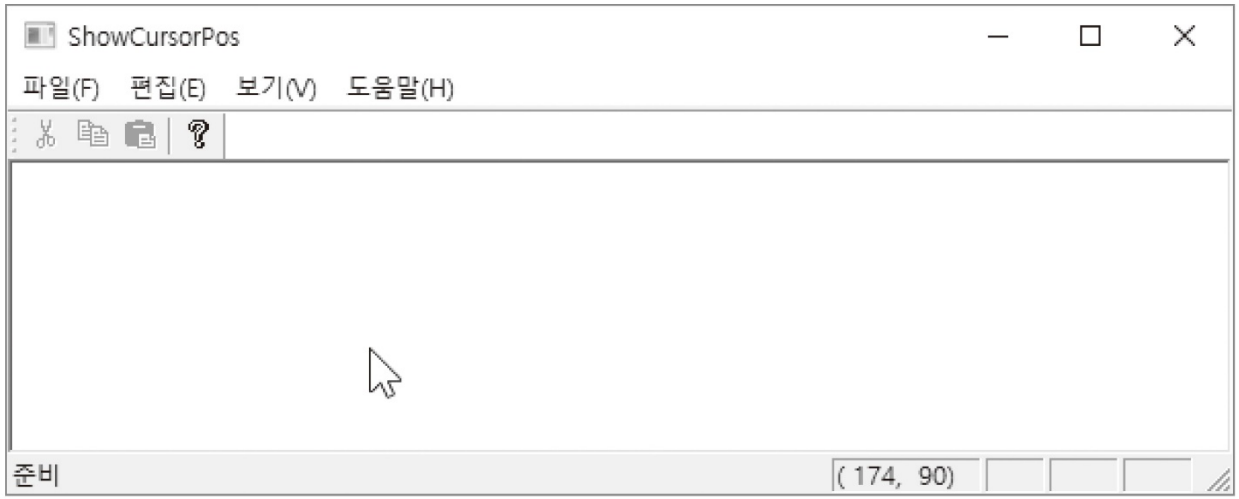


그림 6-37 실행 결과

[실습 6-8] 상태바 구현하기

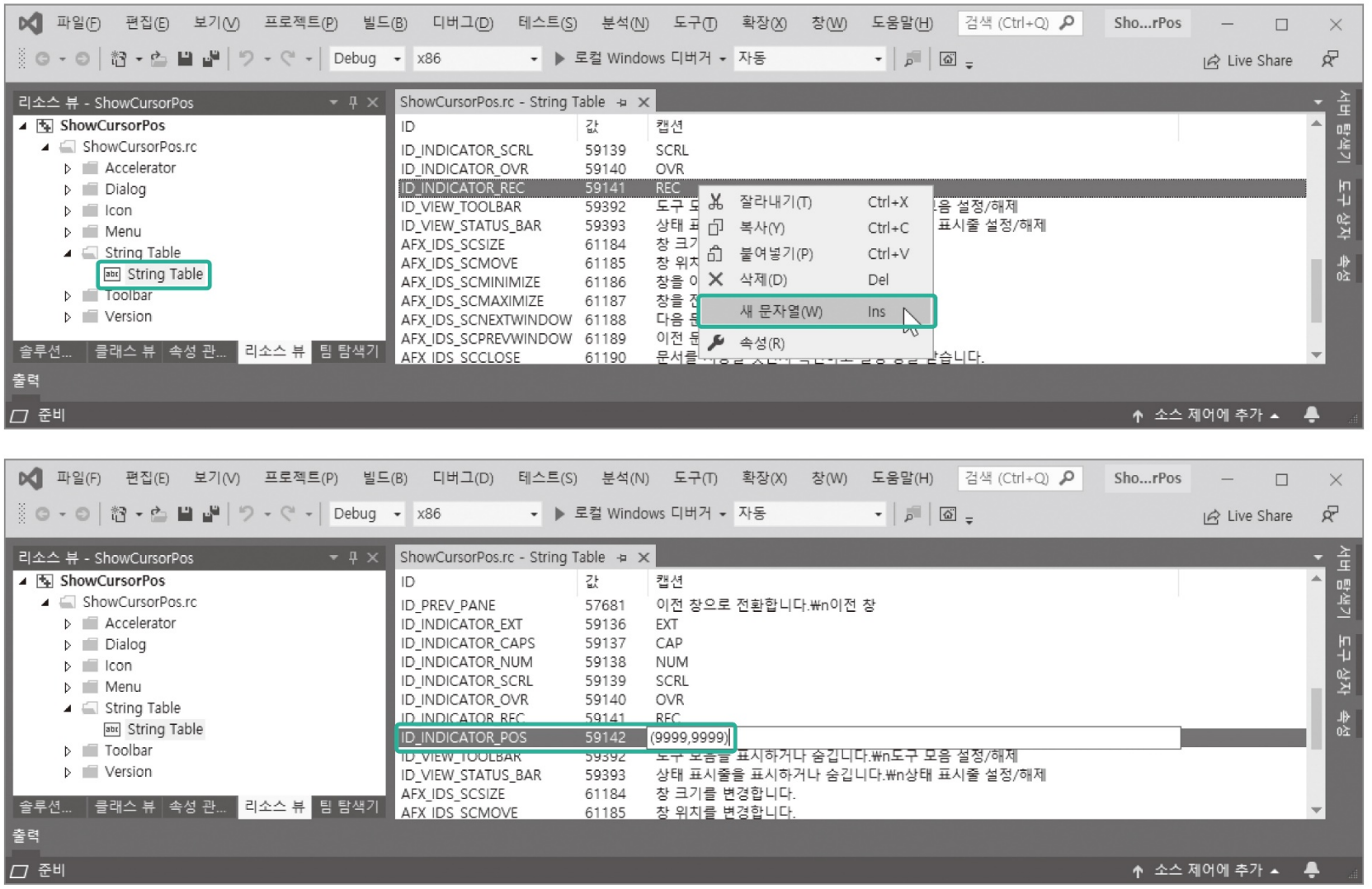


그림 6-38 문자열 리소스 추가, ID/값/캡션 입력

[실습 6-8] 상태바 구현하기

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

[실습 6-8] 상태바 구현하기

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

[실습 6-8] 상태바 구현하기

```
void CChildView::OnMouseMove(UINT nFlags, CPoint point)
{
   CString str;
   str.Format(_T("(%4d,%4d)"), point.x, point.y);
   CMainFrame *pMainFrame = (CMainFrame *)AfxGetMainWnd();
   pMainFrame->m_wndStatusBar.SetPaneText(1, str);
}
```

```
#include "stdafx.h"
#include "ShowCursorPos.h"
#include "ChildView.h"
#include "MainFrm.h"
```

[실습 6-8] 상태바 구현하기

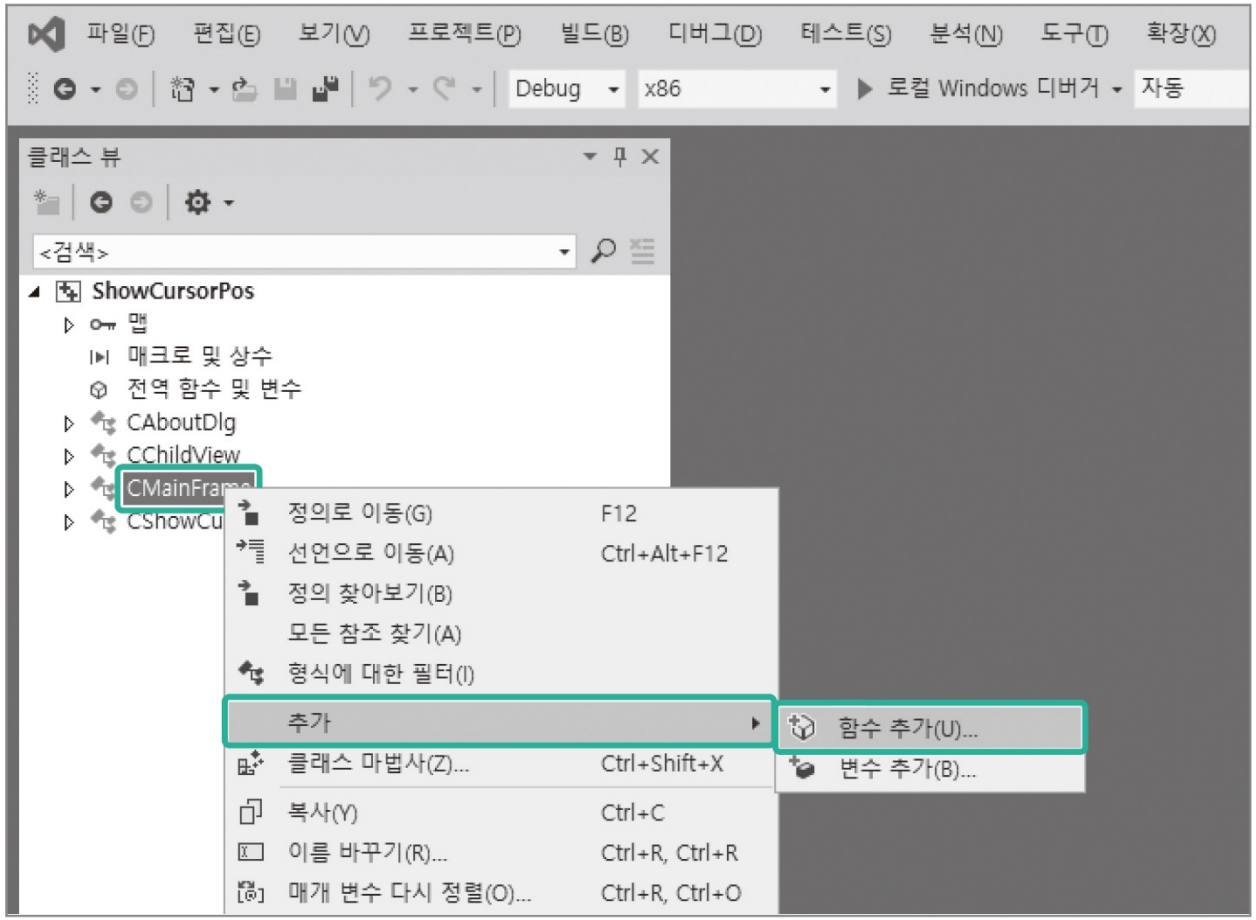


그림 6-39 멤버 함수 추가

[실습 6-8] 상태바 구현하기

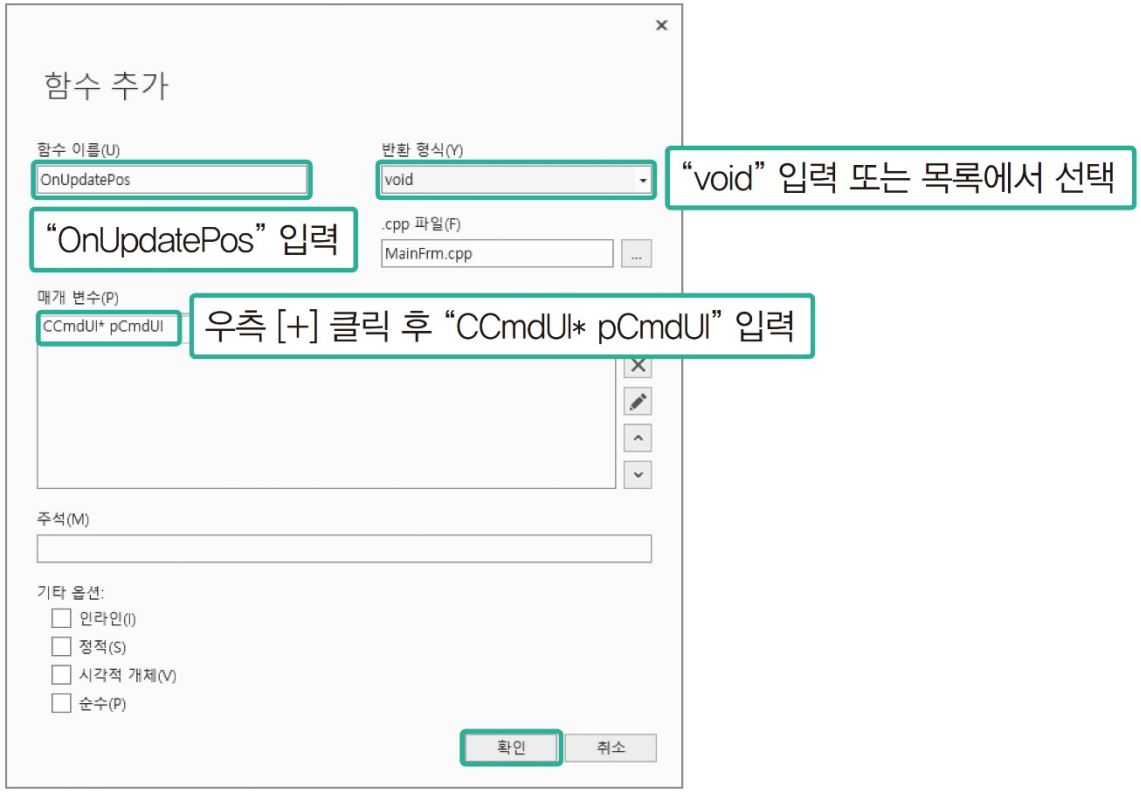


그림 6-40 상태바 갱신 핸들러 추가

[실습 6-8] 상태바 구현하기

```
void CMainFrame::OnUpdatePos(CCmdUI * pCmdUI)
{
   pCmdUI->Enable();
}
```

[실습 6-8] 상태바 구현하기

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