

WEEK

1

DAY 6

애플리케이션에 메뉴를 넣자

대부분의 윈도우 애플리케이션이 사용자들이 원하는 기능을 선택할 수 있도록 윈도우 상에 버튼을 두는 대신에 가지고 있는 것이 풀다운 메뉴이다. 풀다운 메뉴는 여러분이 공들여 만든 멋진 윈도우 화면을 보존하면서 다양한 기능을 제공해 주기 때문에 유용하게 쓰인다.

이번 장에서 여러분이 배울 내용은 다음과 같다.

- ❖ 비주얼 C++ 애플리케이션을 위한 메뉴 생성 방법
- ❖ 메인 디자인로그 윈도우에 메뉴를 부착하는 방법
- ❖ 메뉴를 통해 애플리케이션의 함수를 호출하는 방법
- ❖ 오른쪽 버튼을 눌렀을 때 또는 팝업 메뉴를 만드는 방법
- ❖ 메뉴에 대한 단축키로 사용되는 키보드 가속기를 설정하는 방법

메뉴에 대하여

최초로 컴퓨터 단말기가 출현하고 사용자가 컴퓨터 소프트웨어를 사용하기 시작할 무렵, 대형 메인 프레임에서 소프트웨어를 개발하던 사람조차도 컴퓨터가 수행할 수 있는 기능을 어떠한 형태로든지 간에 메뉴로 정리해서 사용자에게 보여줄 필요를 느꼈다. 초기의 메뉴는 오늘날에 비하면 매우 구닥다리였으며, 이동이나 사용이 매우 어려웠다. 메뉴 시스템은 그 이후 계속적으로 발전해왔으며, 쓰고 배우기 쉬운 방향으로 표준화되었다.

그래픽 유저 인터페이스(GUI)의 아이디어를 처음 생각해낸 소프트웨어 디자이너들은 컴퓨터 시스템과 애플리케이션의 작동을 일관된 방식으로 운영함으로써 학습과 응용에 편리하도록 하자는데 의견을 모았으며, 애플리케이션의 기능을 선택하는데 사용되는 메뉴는 그 GUI 디자인 중 하나였다. 결과적으로 여러 가지의 표준적인 메뉴 스타일이 고안되었다.

메뉴 스타일

첫번째로 생각나는 표준화된 메뉴 스타일은 풀다운(pull-down) 메뉴이다. 이 메뉴는 애플리케이션 윈도우의 상단에 여러 개의 카테고리가 일렬로 늘어서 있는 형태를 취하고 있으며, 사용자가 카테고리 하나를 선택하면, 선택된 카테고리 아래로 메뉴가 뚝 떨어지면서 애플리케이션의 기능을 작동시킬 수 있는 메뉴 항목들이 나타난다.

이 메뉴는 변형된 스타일이 계단식(cascading) 메뉴로서, 메뉴 항목의 오른쪽에 또 하나의 서브 메뉴가 나타나는 것을 말한다. 이 서브 메뉴는 풀 다운 메뉴와 비슷하게 애플리케이션의 기능을 작동시킬 수 있는 메뉴 항목을 가지고 있다. 계단식 메뉴가 몇 개나 이어질지에 대한 제한은 없지만, 대부분의 개발자들에게 재빨리 퍼진 관습은 “두 개 이상 두면 조금 어지럽다”라는 것이다.

새로 고안된 메뉴 스타일은 팝업(pop-up) 메뉴 혹은 문맥(context) 메뉴라고 불리는 것으로서, 애플리케이션 영역의 중간에서 자유롭게 튀어나오는 메뉴이다. 커서나 마우스 포인터로 선택된 개체 혹은 작업 영역에 따라 다른 팝업 메뉴를 띄울 수 있기 때문에 문맥 메뉴라고도 불리는 것이다.

■ 키보드 단축키를 가능하게 하는 메뉴

워드프로세서와 같은 키보드를 많이 사용하는 애플리케이션을 사용할 때는, 키보드를 사용하다가 어떤 기능을 사용하기 위해 마우스로 손을 옮기는 그 자체가 얼마나 생산성을 떨어뜨리는지 알만한 사람은 다 알 것이다. 이에 소프트웨어 디자이너들은 자주 쓰이는 메뉴 항목에 대한 키보드 단축키(keyboard shortcuts)를 추가할 필요성을 느꼈고, 이런 이유로 키보드 단축키(加速기 : accelerator)와 핫키(hotkey)란 개념이 추가되었다.

핫키는 메뉴 항목에 밑줄이 그어진 문자를 말한다. Alt 키와 밑줄이 그어진 문자 키를 같이 누르면, 해당되는 문자를 가진 메뉴 항목을 선택할 수 있게 된다. 이 방법을 사용하면 키보드를 손에서 떼지 않고도 애플리케이션 메뉴를 모두 돌아다닐 수 있다.

조금 더 숙련된 사용자를 위해, 애플리케이션 디자이너들은 키보드 단축키, 즉 키보드 가속기란 것을 추가시켰는데, 키보드 가속기란 애플리케이션 메뉴를 돌아다닐 필요 없이 어떤 애플리케이션 기능을 작동시키기 위해 누를 수 있는 단일 키 조합을 말한다. 자주 쓰이는 애플리케이션 기능에 대해서 메뉴를 계속 불러낼 필요가 없기 때문에 숙련된 사용자에게는 그만이다. 애플리케이션 안에는 사용자들에게 어떤 키보드 가속기를 사용할 수 있는지 알려주기 위해 메뉴 항목의 오른쪽에 키 조합을 표시해 두는 것이 상례이다.

■ 메뉴 사용에 통용되는 표준과 규칙들

메뉴를 어떻게 설계할 것인가에 대한 기준은 없지만, 지금까지 애플리케이션을 만들어온 개발자들의 경험을 통해 내려오는 규칙은 몇 가지가 있다. 상세하게 알고 싶은 분은 마이크로소프트에서 발간한 “Windows Interface Guidelines for Software Design”이란 책을 참고하기 바란다. 이 책을 발간한 목적은 애플리케이션이 일관된 동작을 하도록 함으로써 GUI 시스템의 개발의 이면에 숨겨진 기본적인 목표를 따르는데 도움을 주기 위한 것이니 거부감을 가질 필요는 없을 것이다. 몇 가지를 추려보면 다음과 같다.

- ◆ 메뉴 바에는 한 단어로 된 메뉴 카테고리를 사용할 것. 두 단어로 된 카테고리는 한 단어로 된 카테고리가 두 개 있는 것처럼 보이기 쉽다.

- ❖ File(파일) 메뉴는 왼쪽의 첫번째에 오도록 한다. 이 메뉴는 모든 파일 관련 기능(New, Open, Save, Print 등)을 담고 있으며, Exit(끝내기) 옵션도 여기에 둔다. Exit 옵션은 이 메뉴의 최하단에 두며, 경계선을 사용해서 다른 옵션과 구분한다.
- ❖ Edit(편집) 메뉴는 File 메뉴의 다음에 둔다. 이 메뉴는 Copy, Cut, Paste, Undo, Redo 등의 모든 편집 기능을 가지게 한다.
- ❖ View(보기) 메뉴는 애플리케이션의 작업 영역의 외향을 조정하는 기능을 가지게 한다.
- ❖ Window(창) 메뉴는 다중 도큐먼트 인터페이스(MDI) 스타일의 애플리케이션에 쓰인다. 이 메뉴에는 자식 윈도우의 제어, 현재 윈도우의 선택, 윈도우 레이아웃의 변경 등에 대한 기능을 넣어 두며, 최우측에서 두번째에 두는 것이 상례이다.
- ❖ Help(도움말) 메뉴는 메뉴바의 가장 끝에 있는 메뉴이며, 애플리케이션의 설명이나 문서를 제공하는 메뉴 항목을 둔다. 만일 어떤 애플리케이션의 저작권 정보나 법인 정보를 보여야 할 필요가 있다면 “About <애플리케이션 이름>”으로 된 메뉴 항목을 마지막 항목으로 넣어두도록 한다.

메뉴를 디자인하자

메뉴는 윈도우 애플리케이션에서는 리소스로 정의된다. 따라서 메뉴를 디자인하려면 프로젝트 워크스페이스의 리소스뷰를 사용해야 한다. 여러분이 처음 다이얼로그 스

타일의 애플리케이션을 만들었을 때는 resources 트리 구조에 메뉴에 관한 폴더가

없을 테지만, 걱정할 필요는 없다.

Note

윈도우 애플리케이션은 리소스, 즉 윈도우 레이아웃, 메뉴, 도구바, 이미지, 텍스트 문자열, 가속기 등의 측면에서 보면 새롭게 관찰할 수 있다. 방금 말한 모든 리소스는 리소스 스크립트 파일이라고 알려진 것으로 묶이며, 비주얼 C++ 컴파일러는 이 파일에 정의된 사항에 따라 해당 리소스를 생성하도록 만든다. 리소스 스크립트 파일은 .rc 파일 확장자를 가진 텍스트 파일이며, 리소스의 ID, 캡션, 크기와 위치 등의 정보를 텍스트 형식으로 가지고 있다.

이미지나 사운드 따위의 리소스의 경우에는 텍스트로 표현할 수 없으며, 바이너리 형식으로 저장되어야 한다. 이들 리소스는 별도의 파일로 저장되며, 이 파일의 이름과 저장 위치가 리소스 스크립트 파일에 들어간다.

■ 메뉴를 생성하자

메뉴 생성은 매우 쉽다. 딱 네 가지 단계만 따르면 된다.

1. 메뉴를 담을 애플리케이션을 만든다.
2. 메뉴 리소스를 프로젝트에 추가한다.
3. 메뉴 리소스를 잘 손보고 고쳐서 메뉴 항목을 가지게 한다.
4. 여러분이 넣은 메뉴 항목에 함수를 연결하여 작동되게 한다.

■ 애플리케이션을 만들자

이 장에서 만들 예제는 하나의 메뉴와 하나의 버튼을 가진 다이얼로그 스타일의 간단한 애플리케이션이다. 다음의 단계를 따르도록 하자.

1. MFC AppWizard 애플리케이션을 새로 만들고, 이름을 Menus라고 짓자.
2. 디폴트 설정을 그대로 받아들이며, 다이얼로그의 제목은 Menus로 입력한다.
3. 애플리케이션 위저드가 여러분의 애플리케이션 골격을 다 만들고 난 다음, 다이얼로그에서 모든 컨트롤을 삭제한다.
4. 다이얼로그에 버튼을 하나 놓는다. 이 버튼의 ID는 IDC_EXIT로 하고, 캡션은 E&xit로 입력한다.
5. 클래스위저드를 사용해서 이 버튼에 함수를 연결하는데, 이 함수에서는 OnOK()를 호출하도록 만든다. OnOK()가 애플리케이션을 종료하는 구실을 한다는 것 정도는 이젠 알고 있을 것이다.

Note

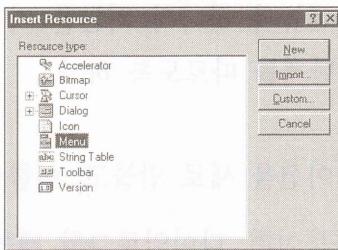
아직 OnOK() 함수를 넣는 방법이 생각나지 않는다면 2장, “애플리케이션 컨트롤 사용하기”의 “애플리케이션 종료하기” 절로 되돌아 가자.

■ 메뉴를 추가하고 직접 고치자

이제 애플리케이션의 바탕은 만들어졌고, 이 애플리케이션의 메뉴를 만들 순서가 남았다. 메뉴를 생성하는 두번째 단계이므로, 메뉴 리소스를 프로젝트에 추가하는 일이 필요하겠다. 여러분이 리소스를 추가할 때 비주얼 C++가 자동으로 메뉴 에디터를 실

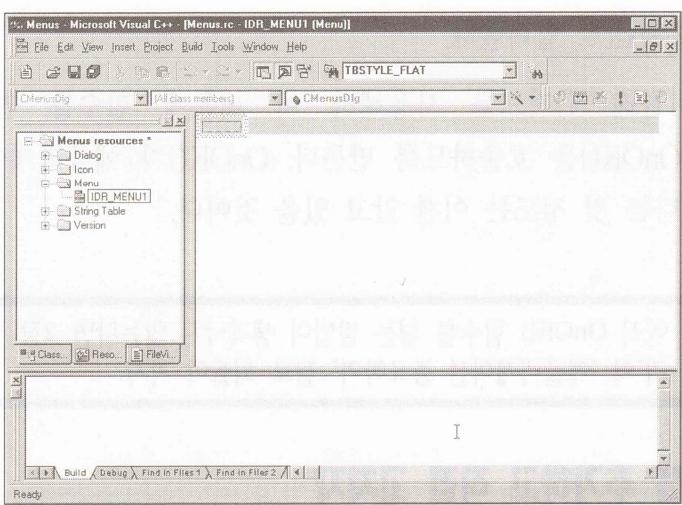
행시기므로 아주 쉽게 메뉴를 편집할 수 있을 것이다. 다음의 단계를 따라서 메뉴를 추가하고 편집해 보도록 하자.

1. 프로젝트 워크스페이스의 리소스뷰를 선택한다.
2. 트리 구조의 최상단 항목을 선택한다.
3. 마우스를 오른쪽 클릭해서 팝업 메뉴를 띄운다.
4. 팝업 메뉴에서 Insert를 선택한다.
5. [그림 6.1]과 같이 Insert Resourceダイ얼로그에서 Menu를 선택하고, New 버튼을 클릭한다.

그림 6.1 → 

Insert Resource
다이얼로그

6. 메뉴 에디터가 디벨로퍼 스튜디오의 에디터 영역에 열리고, [그림 6.2]와 같이 첫번째 메뉴 지점이 테두리로 둘러싸여 나타나 있을 것이다.

그림 6.2 → 

빈 메뉴

이 시점을 정리해 보면, 여러분은 메뉴 리소스를 생성했으며 메뉴 항목을 추가해 나가면서 편집할 준비가 모두 된 상태이다. 이제 다음의 단계를 따라 메뉴 항목을 추가하자.

1. 테두리로 둘러싸인 부분을 오른쪽 클릭한 다음, 팝업 문맥 메뉴에서 Properties를 선택한다.
2. 메뉴 항목의 캡션을 입력하는데, 이 예제에서는 &File이라고 입력하고 프로퍼티 다이얼로그를 닫는다.



현재의 프로퍼티 다이얼로그에서는 애플리케이션이 실행되고 있을 때 메뉴바에 나타날 텍스트를 설정해 주는 것이다. Pop-up 체크 박스가 체크되어 있기 때문에(최상위 메뉴 바 상의 메뉴 항목에 대해선 기본적으로 이렇게 설정된다), 이 메뉴는 애플리케이션의 기능을 작동시키는 구실을 하지 않으며, 따라서 ID를 가질 필요도 없다.

3. 첫번째 드롭다운 메뉴가 놓일 위치가 테두리로 둘러싸여 있다. 이 부분에 메뉴 항목을 추가하기 위해서 다시 한번 마우스를 오른쪽 클릭한 다음, 팝업 문맥 메뉴에서 Properties를 선택한다.
4. 메뉴 항목에 ID와 캡션을 입력해야 한다. 이 예제에서는 IDM_FILE_HELLO와 &Hello를 각각 입력하고 프로퍼티 다이얼로그를 닫자.

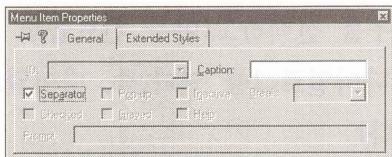


이번의 경우에는 프로퍼티 다이얼로그에 메뉴바에서 메뉴가 열렸을 때 사용자가 보게 될 텍스트 뿐만 아니라, 이 항목이 선택되었을 때 작동될 이벤트 메시지 핸들러에 사용될 ID도 설정해 주는 것이다.

이제 여러분은 하나의 메뉴 항목을 가진 메뉴를 만들었다. 앞의 3번 단계와 4번 단계를 반복하면 테두리로 둘러싸인 빈 부분에 계속해서 메뉴 항목을 추가할 수 있으며, 분리자도 넣을 수 있을 것이다. 분리자(Separator)란 풀다운 메뉴 영역을 구분해 주는 수평선으로, 이 분리자를 넣으려면 다음의 단계를 따르면 된다.

그림 6.3 →

메뉴 분리자를 설정한다.



1. 분리자를 놓을 부분에 테두리가 놓이게 한다. 여러분이 만들고 있는 현재의 예제에서는 두번째 드롭다운 메뉴의 위치가 적합할 것이다. 앞서 보았던 세번째 단계와 같이 프로퍼티 다이얼로그를 열자. 분리자를 넣으려면 단순히 Separator 체크 박스만 체크해 주면 된다([그림 6.3] 참조). 이제 다이얼로그를 닫는다.

여러분의 예제 프로그램을 마무리 짓기 위해, 필자가 지금까지 설명한 똑같은 단계를 따라 Exit 항목을 File 메뉴에 추가하고, About이란 서브 항목을 가진 Help란 두 번째 메뉴를 File 메뉴 옆에 추가하자. 아래의 세 단계가 우왕좌왕하는 여러분에게 도움이 될지도 모르겠다.

1. 세번째 드롭다운 메뉴 항목의 위치에서 프로퍼티ダイ얼로그를 열고 ID를 IDM_FILE_EXIT로 정한 다음, 캡션에는 E&xit를 입력한다.ダイ얼로그를 닫는다.
2. 두번째 최상위 메뉴의 위치에 테두리가 놓이게 한 다음, 프로퍼티ダイ얼로그를 연다. 캡션을 &Help로 하고ダイ얼로그를 닫자.
3. 두번째 최상위 메뉴의 첫번째 드롭다운 메뉴 위치에서 프로퍼티ダイ얼로그를 연 다음, ID를 ID_HELP_ABOUT으로 입력하고 캡션을 &About으로 입력한다.ダイ얼로그를 닫는다.

이제 여러분의 멋진 메뉴가 완성되었다. 하지만 아직 애플리케이션에 부착하지는 않은 상태이다.

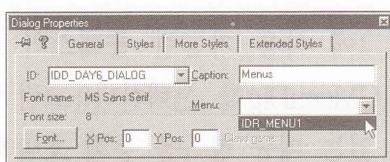
■ 메뉴를ダイ얼로그 윈도우에 부착하자

여러분의 애플리케이션에서 사용할 수 있는 메뉴는 생겼지만, 이 상태에서 애플리케이션을 컴파일하고 실행시키면 절대로 이 메뉴가 나타나지 않는다. 즉, 이 메뉴를ダイ얼로그 윈도우에 부착시킬 수 있는 방법을 강구해야 하는데, 강구할 필요도 없이 다음 단락을 읽으면 되니 얼마나 편한가?

1. 프로젝트 워크스페이스의 리소스류에 있는 Dialog 폴더에서 첫번째 애플리케이션ダイ얼로그(이 경우 IDD_MENUS_DIALOG)를 더블클릭한다.
2. 전체ダイ얼로그 윈도우를 선택하고(아무 컨트롤도 선택되지 않게 하자), 이 디자인뷰의 프로퍼티ダイ얼로그를 연다(즉, 디자인뷰 자체의 프로퍼티ダイ얼로그를 열라는 뜻이다).
3. [그림 6.4]와 같이 Menu 드롭다운 리스트 박스에서 여러분이 만든 메뉴의 ID를 선택한다.

그림 6.4 →

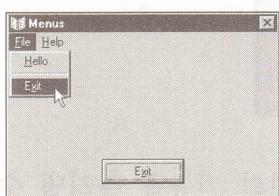
다이얼로그 윈도우에 메뉴를
부착하는 순간



이제 애플리케이션을 컴파일하고 실행하면, 메뉴가 붙어있음을 확인할 수 있을 것이다([그림 6.5] 참조). 여느 애플리케이션과 같이 메뉴 항목을 선택할 수 있다(물론 한 가지 차이는 있지만). 현재는 메뉴 항목을 선택했을 때 아무 일도 일어나지 않는다. 이제 메뉴에 어떤 기능을 연결해야겠다는 생각이 들 것이다.

그림 6.5 →

이제 메뉴가 애플리케이션
다이얼로그의 한 부분이
되었다.

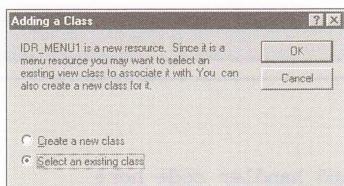


■ 메뉴 항목에 애플리케이션 기능을 연결하자

이제는 메뉴 항목을 선택했을 때 무엇인가가 작동되었으면 금상첨화일 것이다. 그런데, 메뉴가 여러분에게 무엇을 해주길 바라기 전에 여러분이 메뉴에게 무엇을 해주어야 하나를 고민해야 한다(비주얼 C++ 애플리케이션의 어느 곳이 안 그렇겠는가만). 다음의 단계를 따르도록 하자.

그림 6.6 →

메뉴 리소스에 대한 처리
클래스를 설정한다.



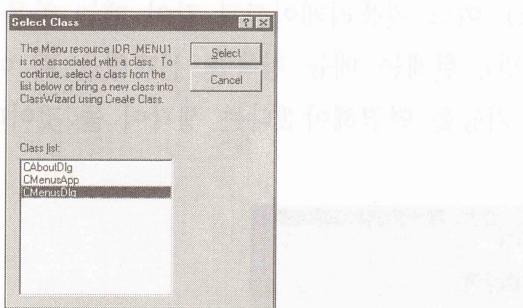
1. 메뉴 에디터로 방금 만든 메뉴를 연다.
2. View 메뉴에서 ClassWizard를 선택한다.
3. Adding a Class 다이얼로그가 여러분 앞에 인사를 드릴 것이다. 5장에서 두 번째 다이얼로그를 추가하려 할 때 나왔듯이 말이다. Select an Existing Class가 선택된 채로 놔두고 OK를 클릭한다([그림 6.6] 참조).

5장에서는 두 번째 다이얼로그를 추가하려고 했을 때 이 다이얼로그 윈도우에 대한 C++ 클래스가 새로 필요했었지만, 이번 장의 메뉴의 경우 메뉴가 부착된 다이얼로그 윈도우의 C++ 클래스를 사용해도 충분하다.

4. Select Class 다이얼로그에 사용 가능한 클래스의 리스트가 나오는데, 여기서 다이얼로그 클래스 즉, CMenuDlg를 선택한다([그림 6.7] 참조). 즉, 비주얼 C++에게 이 메뉴가 가지고 있는 메뉴 항목을 통해 호출되는 모든 함수는 이 메뉴가 부착된 다이얼로그 윈도우를 처리하는 클래스에 속해 있다고 알려주는 것이다.

그림 6.7 →

Select Class 다이얼로그



이젠 메뉴 항목을 통해 애플리케이션의 기능을 불러낼 수 있도록 하기 위해서 다이얼로그 윈도우의 컨트롤에 대해 해준 것과 같이, 클래스위저드를 사용해서 이벤트 핸들러를 추가할 수 있을 것이다.

이번 장의 예제에서는 IDM_FILE_HELLO(Hello 메뉴의) 항목에 대해 COMMAND 이벤트 메시지 처리 함수를 추가해 보도록 하자. 이 함수의 이름은 OnHello이며, [리스트 6.1]이 그 코드이다.

리스트 6.1 OnHello() 함수

```

1 :void CMenuDlg::OnHello()
2 :{
3 :    // TODO: Add your command handler code here
4 :
5 :    /////////////////////
6 :    // 새로 넣을 코드가 여기서부터 시작된다
7 :    ///////////////////
8 :
9 :    // 사용자에게 메시지를 띄운다
10:   MessageBox("Hello there", "Hello");
11:
12:   ///////////////////
13:   // 새로 넣을 코드는 여기서 끝난다
14:   ///////////////////
15: }
```

Note

COMMAND 이벤트 메시지는 메뉴 항목이 선택되었을 때 애플리케이션 윈도우로 전달되는 메시지로서, 이 이벤트 메시지에 함수를 놓는 일은 선택된 메뉴 항목에 대해 함수를 놓는 것과 동일한 결과이다.

메뉴 항목의 COMMAND 이벤트에 이미 존재하는 어떤 함수를 연결해 두면, 그 메뉴 항목을 선택했을 때 연결된 그 함수가 호출될 수 있는데, 메뉴 항목 ID에 함수를 연결하고, 제시된 이름 대신에 이미 존재하는 함수의 이름을 설정하면 그만이다.

예를 들어, Exit 메뉴 항목에 대해 OnExit() 함수를 다시 사용하려면 메뉴 에디터를 다시 열고 클래스위저드를 연다. 클래스위저드가 열렸을 때 IDM_FILE_EXIT의 COMMAND 이벤트 메시지에 대한 함수를 추가하고, 클래스위저드가 제시하는 이름 대신 OnExit라고 써넣는다. 이렇게 함으로써 Exit 버튼에 대해 만들어둔 OnExit() 함수와 IDM_FILE_EXIT 메뉴가 자동적으로 연결되는 것이다.

여러분이 만든 예제 프로그램을 마무리 짓기 위해서는 ID_HELP_ABOUT 메뉴의 COMMAND 이벤트 메시지에 대해 함수를 추가하고, [리스트 6.2]에 나온 코드를 써 넣도록 하자.

리스트 6.2 OnHelpAbout() 함수

```

1: void CMenusDlg::OnHelpAbout()
2: {
3:     // TODO: Add your command handler code here
4:
5:     /////////////////
6:     // 새로 넣을 코드가 여기서부터 시작된다
7:     /////////////////
8:
9:     // Aboutダイアログ에 대한 인스턴스를 선언한다
10:    CAaboutDlg dlgAbout;
11:
12:    // Aboutダイアログ를 띄운다
13:    dlgAbout.DoModal();
14:
15:    /////////////////
16:    // 새로 넣을 코드는 여기서 끝난다
17:    ///////////////
18: }
```

여러분은 File | Exit 메뉴 항목과 애플리케이션을 종료하는 함수를 연결하였고, File | Hello 메뉴 항목에 대해서는 사용자에게 간단한 메시지를 띄우는 MessageBox() 함수를 호출하게 하였으며, Help | About 메뉴 항목의 경우 About 디아얼로그 윈도우의 인스턴스를 선언하고 DoModal() 함수를 호출하도록 하였다.

이제 여러분의 애플리케이션을 컴파일하고 실행시키면, 메뉴 항목이 작동되는 것을 확인할 수 있을 것이다. [그림 6.8]과 같이 Help | About을 선택하면 About 디아얼로 그([그림 6.9] 참조)가 뜨고, File | Hello를 선택하면 “Hello there”를 띄우는 메시지 박스를 볼 수 있으며([그림 6.10] 참조), File | Exit를 선택하면 바로 애플리케이션이 종료된다.

그림 6.8 →
Help | About 메뉴 항목

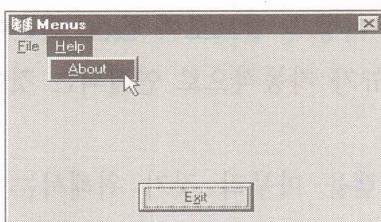


그림 6.9 →
About 디아얼로그

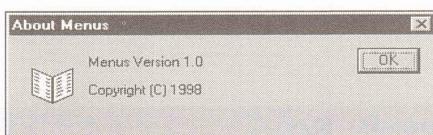


그림 6.10 →
Hello there를 표시하는
메시지 박스



팝업 메뉴를 생성하자

대부분의 윈도우 애플리케이션은 소위 “팝업(pop-up)” 혹은 “문맥(context)” 메뉴란 것을 가지고 있어서, 오른쪽 마우스 버튼을 클릭해서 작업 내용에 맞는 애플리케이션 기능을 즉시 수행할 수 있도록 되어 있다. 팝업 메뉴는 애플리케이션의 작업 영역에서 나타나며, 메뉴바나 윈도우 프레임 어니에도 붙어 있지 않다. 또한, 메뉴가 열린 지점이 어디냐(마우스 포인터의 위치 혹은 선택된 개체에 따라)에 따라 메뉴 항목이 다르게 나타날 수 있으므로 문맥 메뉴라고도 불린다.

팝업 메뉴를 여러분의 애플리케이션에서 제공하기 위해서는 두 가지 접근 방식을 생각해 볼 수 있다. 즉, 팝업 메뉴로 특별히 사용될 메뉴를 디자인하거나 이미 디자인한 메뉴를 가진 풀다운 메뉴들 중 하나를 사용하면 되는데, 전자의 접근 방식을 택한다면 최상위의 메뉴 바 요소를 없애야 한다(캡션에다가 공백을 넣거나 아무 텍스트나 넣으면 된다). 팝업 메뉴로 사용될 커스텀 메뉴를 만들 때 메뉴가 어떻게 작동되는지에 대해서는 11장, “다중 도큐먼트 인터페이스 애플리케이션 만들기”의 “문맥 메뉴를 추가하자” 절에서 공부할 수 있을 것이다.

후자의 접근 방식을 택할 경우 메뉴 내의 풀다운 항목의 모든 부분을 팝업 메뉴로 사용한다. 이렇게 하려면 일단 서브 메뉴(풀다운 메뉴)의 핸들을 얻어낸 다음 TrackPopupMenu() 함수를 호출하도록 한다. 팝업 메뉴의 나머지 기능은 이미 이 메뉴를 만들어 두었을 당시의 것을 그대로 사용하기 때문에 더 이상 건드릴 것이 없다. 자, 그럼 다음의 단계를 따르자.

1. 클래스위저드를 사용해서 WM_CONTEXTMENU 이벤트 메시지에 대한 함수를 여러분의 다이얼로그 윈도우에 추가한다.

Note

문맥 메뉴를 띄우는데 사용할 수 있는 이벤트 메시지에는 두 가지가 있다. 바로 머리에 떠오르는 것은 WM_RBUTTONDOWN 이벤트로서, 오른쪽 버튼을 눌렀을 때 발생되는 메시지이다. 또 하나는 WM_CONTEXTMENU 이벤트로서 문맥 메뉴를 띄울 때만 특별히 사용되도록 준비된 메시지인데, 두 가지 사건 중 하나에 의해 발생된다. 한 사건은 오른쪽 마우스 버튼을 떼는 일이고, 또 한 사건은 윈도우 지원 키보드에 붙어있는 문맥 메뉴 키를 누르는 일이다.

2. 이 함수를 [리스트 6.3]과 같이 만든다.

리스트 6.3 OnContextMenu() 함수

```

1: void CMenusDlg::OnContextMenu(CWnd* pWnd, CPoint point)
2: {
3:     // TODO: Add your message handler code here
4:
5:     ///////////////////////////////
6:     // 새로 넣을 코드가 여기서부터 시작된다
7:     ///////////////////////////////
8:
9:     // 지역 변수를 선언한다
10:    CMenus *m_lMenu;      // 메뉴에 대한 포인터

```

```

11:     CPoint m_pPoint; // 마우스 위치의 사본
12:
13:     // 마우스 위치를 지역 변수에 복사한다
14:     m_pPoint = point;
15:     // 마우스 위치를 화면 위치로 변환한다
16:     ClientToScreen(&m_pPoint);
17:     // 메뉴에 대한 포인터를 얻어낸다
18:     m_lMenu = GetMenu();
19:     // 서브 메뉴에 대한 포인터를 얻는다
20:     m_lMenu = m_lMenu->GetSubMenu(0);
21:     // 팝업 메뉴를 띄운다
22:     m_lMenu->TrackPopupMenu(TPM_CENTERALIGN + TPM_LEFTBUTTON,
23:                             m_pPoint.x, m_pPoint.y, this, NULL);
24:
25:     ///////////////////////
26:     // 새로 넣을 코드는 여기서 끝난다
27:     /////////////////////
28: }

```

[리스트 6.3]에서 처음 해준 일은 마우스의 현재 위치를 복사한 일이다. 이 마우스 위치는 애플리케이션 윈도우의 왼쪽 상단을 기준으로 한 상대 좌표값으로, 팝업 메뉴를 띄우기 위해서는 이 좌표값을 전체 모니터 화면을 기준으로 한 절대 좌표값으로 바꾸어 주어야 한다. 만일 바꾸지 않을 경우에는 팝업 메뉴가 어디에서 나타날지 책임지지 않겠다.

#164 없으면 그릇 6.3이 되고 있음, 차이 전제에서 바꾸는

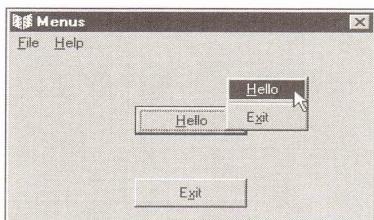
마우스의 위치를 절대 좌표값으로 바꾸어 준 다음에는 윈도우 메뉴에 대한 포인터를 얻어내었다. 이 포인터는 항상 지역 변수로 사용해야 한다. 왜냐하면, 메뉴의 위치는 애플리케이션이 실행되는 동안 바뀔 수 있기 때문이다. 메뉴 포인터를 사용해서 첫 번째 드롭다운 메뉴(서브 메뉴의 순서는 C/C++에서 순서를 세는 방식, 즉 0을 기준으로 센다)의 포인터를 얻어냈고, 이 포인터를 CMenu 클래스의 인스턴스를 가리키 것으로 간주하고 다룰 수 있게 되었다.

이 함수의 핵심이자 마지막은 CMenu의 멤버 함수인 TrackPopupMenu()를 호출한 것이다. 이 함수는 다섯 개의 인수를 받으며 이것을 사용해서 어디에, 또 어떻게 팝업 메뉴를 띄울지를 결정한다. 첫번째 인수는 두 개의 플래그를 조합한 값이다. 첫번째 플래그인 TPM_CENTERALIGN은 마우스 위치를 팝업 메뉴의 중앙으로 놓겠다는 뜻이며, 이 대신 TPM_LEFTALIGN이나 TPM_RIGHTALIGN을 사용해도 된다. 두번째 플래그인 TPM_LEFTBUTTON은 팝업 메뉴의 기능 선택을 왼쪽 마우스 버튼으로 하겠 뜻이다. 물론 오른쪽 버튼으로 하고 싶다면 TPM_RIGHTBUTTON을 사용할 수 있다.

TrackPopupMenu() 함수의 두번째와 세번째 인수는 팝업 메뉴의 화면 위치로서, 모니터 화면을 기준으로 한 절대 좌표값임을 명심하자. 네번째 인수는 메뉴의 명령 (COMMAND) 메시지를 받는 윈도우의 포인터이다. 마지막 인수는 팝업 메뉴를 닫지 않고 사용자가 클릭할 수 있는 사각영역인데, NULL을 넘길 경우 사용자가 팝업 메뉴의 외부를 클릭했을 때 이 메뉴가 닫힌다. [그림 6.11]은 팝업 메뉴가 작동중인 애플리케이션의 실행 모습이다.

그림 6.11 →

작동중인 팝업 메뉴



메뉴에 키보드 가속기를 달자

메뉴 항목을 선택하는 키보드 단축키 중 하나가 바로 키보드 가속기이다. 이 장의 앞부분에서 말했듯이 키보드 가속기는 특별한 키 조합으로서, Ctrl 키와 다른 키를 같이 누름으로써 작동된다. 각각의 키 조합은 하나의 메뉴 이벤트 함수에 대응된다.

키보드 가속기의 작동은 메뉴와 비슷하다. 이것 역시 프로젝트 워크스페이스의 리소스뷰에서 볼 수 있는, 테이블로 정의된 애플리케이션 리소스이며, 각 테이블 엔트리는 ID와 키코드 조합을 가지고 있다. 키보드 가속기를 정의한 후에는 해당 ID에 함수를 연결할 수 있다. 물론 키보드 가속기 엔트리에 동일한 메뉴 항목의 ID를 할당하여 애플리케이션에서 하나의 메시지 맵 엔트리만을 사용하게 할 수도 있다.

일단 모든 키보드 가속기를 정의한 후에는 메뉴 항목에 키 조합을 할당하여 사용자가 알 수 있도록 해야 하는데, 메뉴 항목의 캡션의 끝에다가 \t를 붙이자. \t는 메뉴가 표시될 때 탭으로 바뀌기 때문에 메뉴 캡션과 키 조합의 표시를 분리할 수 있다.

그런데, 이 장에서는 키보드 가속기에 대한 공부는 설명으로만 끝내야 한다. 불행하게도 키보드 가속기는 디자인으로 스타일의 애플리케이션에서는 작동되지 않기 때문이다. 하지만, 몇장을 더 배우고 난 후, 단일 도큐먼트 인터페이스와 다중 도큐먼트 인터페이스 스타일의 애플리케이션을 만들 때 키보드 가속기에 대해 확실히 공부할 수 있게 될 테니 일단은 접어두도록 하자.

요약

이번 장에서는 메뉴에 대해서 공부하였다. 여러분은 비주얼 C++를 사용해서 메뉴를 만드는 방법과 이 메뉴를 다이얼로그 스타일의 애플리케이션에 부착하는 방법을 실습을 통해 알아보았으며, 메뉴를 부착한 다음에는 메뉴 항목에다가 원하는 동작을 수행하는 함수를 연결해 보았다. 이후에는 여러분의 메뉴를 팝업, 즉 문맥 메뉴의 일부로 사용하는 방법을 알아보았고, 키보드 가속기에 대한 개괄적인 소개로 끝을 맺었다.

Q&A

다른 사람들이 사용하는 메뉴의 이름을 꼭 사용해야 하나요? 예를 들면, File이나 Help처럼요. 꼭 이것만을 사용할 필요는 없지 않습니까?

뭐, 물론 메뉴 이름 짓기는 여러분 자유입니다. 하지만, 파일에 관련된 기능은 File 메뉴에, 도움말에 관련된 기능은 Help 메뉴에 넣는 것은 하나의 규칙처럼 받아들여지고 있지요. 메뉴 이름을 Broccoli(브로콜리), Corn(옥수수), Carrot(당근) 따위로 지었다면, 이 메뉴를 Vegetables(야채)로 부를 수 있겠죠. Food(음식)나 Plants(식물)가 더 적합할지도 모르겠지만요. 여러분의 애플리케이션을 다른 사람들이 익히기 쉽게 만들고 싶으시면 메뉴의 풀다운 부분에 들어 있는 항목을 정확하게 대표할 수 있는 이름을 짓도록 하시길 바랍니다. 이것은 상식입니다.

왜 키보드 가속기에 대응하는 키로 하나의 문자만을 사용할 수 없나요?

단일 문자는 WM_KEYDOWN 메시지를 발생시키지, 메뉴 메시지 즉, COMMAND 메시지를 발생시키지 않습니다. 윈도우 운영체제의 설계자가 키보드 가속기의 작동을 결정할 때, 단일 문자 키는 활성화된 애플리케이션에 대한 “입력”으로 간주되도록 하였지요. 만일 단일 문자 키도 키보드 가속기의 구성을 하도록 허용하였다면, 아마 윈도우 운영체제는 입력된 문자가 진짜 입력인지, 키보드 가속기인지 구분할 수가 없을 것입니다. 키조합을 하게 함으로써(기능키 - F1, F2 등의 -는 제외하기로 하지요), 윈도우 운영체제는 간편해질 수 있는 거죠.

실습해 보기

“실습해 보기” 절에서는 배운 것을 확인하는 퀴즈와 이를 활용해서 응용력을 높이기 위한 연습문제를 풀어볼 기회를 가질 수 있게 될 것이다. 퀴즈와 연습문제의 답은 부록 B, “퀴즈 및 연습문제 해답”에 있으나 미리 보라고 만든 것은 절대 아님을 뺏 속 깊이 새겨 두자.

■ 퀴즈

- 메뉴 항목을 선택하면 어떤 이벤트 메시지가 윈도우의 메시지 큐에 들어가는가?
- 다이얼로그 윈도우에 메뉴를 부착하려면 어떻게 할까?
- 메뉴에 대한 이벤트 메시지를 처리하기 위해 설정해줄 기준의 클래스는 어느 것이 좋을까?
- 팝업 메뉴를 띄우게 하는 메시지는 무엇이 있을까?

■ 연습문제

- 메인 윈도우에 버튼을 하나 추가하여 Hello 메뉴 항목을 선택할 때 호출되는 함수와 동일한 함수를 호출하게 해보자.
- 여러분의 애플리케이션에 Help 드롭다운 메뉴를 사용하는 팝업 메뉴를 추가해 보자.