第6章 菜 单

让用户以全新的和令人兴奋的方式与应用程序进行交互时，那些普通的旧菜单往往被忽视；当用应用程序向导创建一个应用程序时，它自动为主窗口创建一个通用的菜单，但只要用一点点的努力，就可以使它们看起来更加美观动人。

例12 使用菜单编辑器 用菜单编辑器添加新增的命令到由 AppWizard产生的菜单。

例13 添加一个菜单命令处理函数 用ClassWizard自动将菜单单击命令与处理命令的某个类的成员函数相联系。

例14 根据当前可视文档动态改变菜单 讨论每逢打开一个特定的视图时，怎样用新命令自动更新一个主菜单。

例15 启用和禁用菜单命令 讨论怎样灰化和正常化菜单项。

例16 复选标记菜单命令 讨论怎样在菜单项的边上放置复选标志，用以指示它们当前状态。

例17 单选标记菜单命令 放置一个点到一组菜单项边上，用以指示当前组中边上有点的菜单项是活动的。

例18 动态修改菜单 讨论程序运行时添加一个菜单项到菜单中，以及从菜单中删除一个菜单项。

例19 动态修改系统菜单 讨论怎样添加命令到系统菜单，系统菜单是单击应用程序图标时显示的菜单。

例20 触发一个菜单命令 讨论怎样使一个菜单项在程序中被“ 击中 ”。

例21 创建弹出式菜单 当用户在应用程序视图内右击鼠标时，将创建一个浮动菜单。

6.1 例12 使用菜单编辑器

目标

添加、删除、改变应用程序菜单中的一个命令，或创建一个新菜单。

注意 给菜单添加的任何菜单项，在为它添加菜单命令处理函数之前，将显示灰色，并且是无效的。参见下面“ 怎样添加一个处理函数 ”。

步骤

用菜单编辑器添加一个菜单项到菜单中

1) 要创建一个新的菜单，单击 Developer Studio的Insert/Resource菜单命令，打开“ Insert Resource(插入资源) ”对话框，从列表中选择“ Menu(菜单) ”并单击New。

2) 要编辑一个已有的菜单，在应用程序资源的文件夹中找到相应菜单的 ID，并双击它。

3) 要添加一个新的菜单项到菜单中，用鼠标拖动空白焦点矩形到新菜单项的位置(见图6-1)。

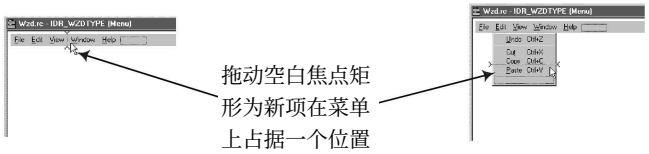

图6-1 拖动空焦点矩形到一个新位置，以添加一个菜单项到该菜单

4) 然后，双击或右击该矩形，选择 Properties (属性)以打开属性对话框(见图6-2)。

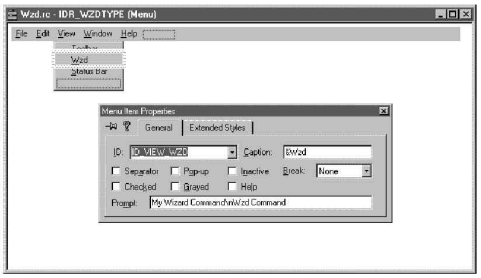

图6-2 用属性对话框改变菜单项的属性

5) 在ID域定义个人的菜单ID，如果用户不输入任何ID，则系统自动创建一个 ID，当单击该菜单项时，该ID成为命令ID发送到应用程序。

6) 在Caption(标题)域输入菜单项的文本，在字符前面添加一个 &号表示用户可以通过键盘访问该菜单项，该字符在菜单中显示一个下划线。

7) 在Prompt(提示)编辑框中填入帮助消息，当鼠标指到该项时，该消息将出现在状态栏上。在该提示的后面添加一个换行符 (\n)，然后添加一个更小的帮助消息。该更小的消息 (也叫做气泡帮助消息)将显示在任何产生该命令的工具栏按钮上面。在这里可以看出，第一消息可以是详细的消息，而第二消息应该简单明了。

```
Opens the file system.\nOpen File
```

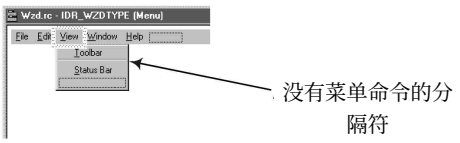

图6-3 选中Separator复选框以添加一个菜单分隔符

8) 可以忽视Checked、Grayed和Inactive复选框，因为在运行时可以动态地改变这些属性 (见例15)。如果只想让该菜单项在菜单中显示一个分隔符，则选中Separator复选框(见图6-3)。

9) 对于菜单条项，如果想使该菜单项和所有它管辖之下的菜单项在菜单条上右对齐，则选取Help复选框(见图6-4)。

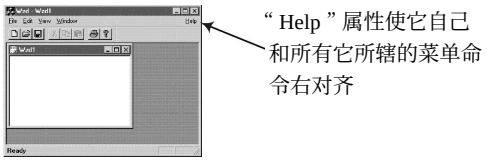

图6-4 选取菜单条中的Help复选框项

10) 如果想把大菜单分割成几列，则在新列开始处选取 Break。Column只是创建一个新列，而Bar创建一列，然后在它和最后一列之间添加一个分隔符，图6-5显示了一个没有条的菜单。

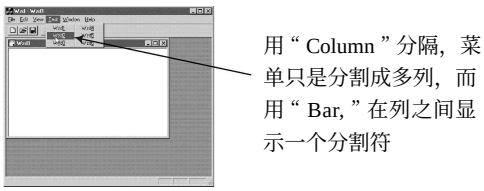

图6-5 选择Break/Column把大菜单分隔成多列

注意 一些菜单属性(包括列分割)不能在正在编辑的菜单图像中反映出来，只有在应用程序建立并运行时，这些属性才能显示出来。

11) 修改一个已有的菜单项，只要用鼠标选取它，并打开它的属性对话框；要删除一个菜单项，用鼠标选中它，然后按键盘上的 Delete键。

说明

也可以动态地创建一个菜单(见例21)

CD说明

在CD上没有该例相应的工程。

6.2 例13 添加一个菜单命令处理函数

目标

用ClassWizard自动添加一个菜单命令处理函数到一个类中，该处理函数实际上是该类的一个成员函数。当一个菜单项被单击时，该函数被 MFC的消息映像系统调用。 ClassWizard添加的函数开始是空的。

步骤

1. 用ClassWizard添加一个命令处理函数

1) 单击Developer Studio的View/ClassWizard菜单命令，打开ClassWizard对话框，从类名组合框中找到并选取用来处理菜单命令的类。

注意 虽然一个类名可能显示在该组合框中，但用它来处理菜单命令仍可能是不合法的，参见本节的“说明”部分。

2) 对象ID列表框包括了所有可得菜单和工具栏命令的 ID，找到合适的ID，并选取它，然后从Messages列表框中选择COMMAND；现在，单击Add Function按钮真正地添加该命令处理函数到该类中，单击Edit Function使环境的文本编辑器打开该类的文件，并定位在该新增函数处，该函数在开始时没有任何代码。这时，运行应用程序，与它关联的菜单项变得有效，但是，单击它显示没有任何反映。

3) 参见图6-6以讨论这些步骤。

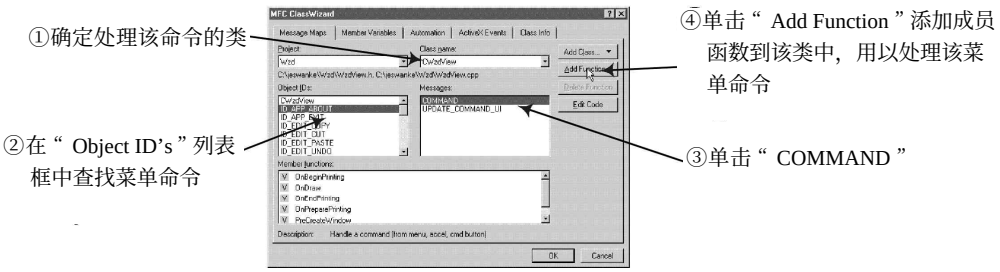

图6-6 遵循这些步骤用ClassWizard添加一个菜单命令

2. 用ClassWizard删除一个命令

重复添加一个命令处理函数的步骤，但是，最后单击 Delete Function按钮，要求ClassWizard从类中删除该菜单处理函数。事实上，所有 ClassWizard删除的是该函数在消息映像中的项，而要从.cpp文件中删除真正的函数还得由你来完成。

说明

- 如果在Message列表框中选择 UPDATE_COMMAND_UI而不是COMMAND，则ClassWizard添加一个用户界面更新处理函数到类中。该处理函数允许用户检查标志、禁用菜单项，甚至改变菜单项文本(见例15与例16)。
- 在添加一个命令处理函数到应用程序中，用以支持一个新的菜单命令之前，那个菜单项命令是灰色的并且是无效的。如果不想为所有的菜单项添加处理函数并使它们看起来有效，可以添加下面的代码到CMainFrame类的构造函数中。

```
CMainFrame::CMainFrame()
{
   m_bAutoMenuEnable =FALSE;
}
```

- 尽管ClassWizard允许用户为项目中的每个类添加一个命令处理函数， MFC的消息处理系统也可能不能顺利的调用那个类来处理命令消息，菜单命令消息跟踪一组路径到达应用程序类，这在第3章已经简述过。在大多数情况下，类可能是在该路径下，尤其是那些AppWizard创建的类。然而，若添加一个新类到应用程序，可能需要手工把路径插入消息流中(见例61)。

CD说明

在CD上没有该例相关的工程。

6.3 例14 根据当前可视文档动态改变菜单

目标

在一个MDI应用程序中，根据当前看到的文档显示一个不同的菜单 (见图6-7)。

策略

当在MDI应用程序中注册多种类型文档模板时，可以轻松获得该功能：一个 MDI应用程序可以编辑多种类型文档，根据当前正在编辑的内容，利用该特征，添加特定的命令到菜单条中。例如，Developer Studio的Image菜单命令，它只在编辑图像时出现(如位图、图标等)。

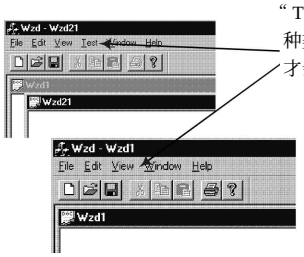

图6-7 注册多个文档模板类型以根据观察到的文档改变菜单

步骤

在应用程序类中定义一个新的文档模板

1) 用ID编辑器为该文档类型创建新的ID。在本例中，我们用IDR_WZDTYPE1。

2) 选择Workspace窗口的ResourceView，并打开Menu文件夹，找到应用程序当前菜单的菜单ID，选中它，然后按Ctrl+C和Ctrl+V复制它。通过右击该复制的ID并单击Property，给它设置新文档类型的ID。

3) 重复上一步为该文档类型创建一个图标，图标位于 Icon文件夹中。

4) 重复上一步在串表中创建一个新文档串类型，串表位于 String Table文件夹中，找到下面这样的串：

```
IDR_WZDTYPE "\nWzd\nWzd\n\n\nWzd.Document\nWzd Document"
```

还应当改变这些值以反映新的文档类型。现在，只要在每次提到应用程序名字的后面填加1，本例中是Wzd。

5) 用ClassWizard有选择地创建一个新视图类和新文档类。参看例 1，确定从哪个MFC类派生类。

6) 现在，可以使用创建的资源和类，在应用程序类中创建一个新的文档模板。直接把下面的代码添加到创建当前文档模板代码的后面。

```
pDocTemplate = new CMultiDocTemplate(
    IDR_WZDTYPE1,                   <<new document id type
    RUNTIME_CLASS(CWzdDoc),         <<new document class
    RUNTIME_CLASS(CChildFrame),     <<MDI child frame
    RUNTIME_CLASS(CWzd2View));      <<new view class
    AddDocTemplate(pDocTemplate);
```

7) 现在，可以用菜单编辑器改变第二个，即拷贝的菜单。当用户打开一个文档时，他们可以选择打开一个原始文档类型还是该新文档类型。如果两种类型同时打开，它们都有各自不同的菜单，单击相应的视图，使应用程序菜单作出相应变化。

说明

- 当使用两种文档类型时，应用程序应该自动提示，并要求用户选择创建新文档时使用哪种文档类型；当打开一个已有文档时，如果应用程序不能确定使用哪种类型的视图，它也会提示用户选择文档类型。在提示中出现的消息是步骤（4）中所示的第二个Wzd值。
- 想更多地了解一般的文档/视图，参见第1章和第2章。

CD说明

在CD上执行项目时，可以选择打开一个 Wzd文档或一个Wzd32文档。选择Wzd，接着用File/New创建一个新文档，然后用 Wzd22创建一个新文档。当交替单击各自的视图时，观察菜单的变化，注意Test菜单项只在Wzd22文档中出现。

6.4 例15 启用和禁用菜单命令

目标

启用和禁用一个菜单项(见图6-8)。

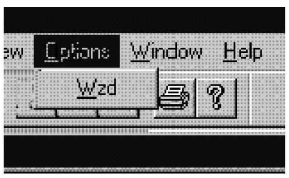

图6-8 添加一个界面消息处理函数，以有选择性地启动和禁用一个菜单命令

策略

如果一个新的菜单项没有命令处理函数，应用程序会自动禁用它。要启用这样的菜单项，只要求用 ClassWizard为该菜单命令添加一个命令处理函数(见例13)。但是，如果想根据应用程序的某些条件，有选择地启用和

禁用一个菜单项，可以用ClassWizard添加一个界面消息处理函数。

步骤

用ClassWizard添加用户界面命令处理函数

1) 遵循例13的步骤，象添加一个菜单命令到类中一样，但不是选择 COMMAND，而是选择UPDATE_COMMAND_UI。然后，ClassWizard会为类创建一个用户界面处理函数。

2) 添加下面代码到该新的处理函数中。在本例中，如果 m_bWzd为TRUE，菜单将变得有效，如果是FALSE，则无效。

```
void CWzdView::OnUpdateWzdButton(CCmdUI* pCmdUI)
{
   pCmdUI->Enable(m_bWzd);
}
```

说明

- 当应用程序首次打开一个菜单时，它将查看在该菜单中每个菜单项是否有一个UPDATE_COMMAND_UI消息处理函数；如果没有，应用程序就检查该菜单项是否有某种COMMAND消息处理函数，如果也没有这样的处理函数，应用程序会自动地禁用该菜单项，并使它显示灰色。通过在 CMainFrame的构造函数中把m_bAutoMenuEnable设为FALSE，可以禁用该自动功能。在软件开发中，该功能有助于提醒用户还有什么没做。
- 应用程序不仅可以启用和禁用菜单项，也可以启用和禁用工具栏按钮、对话条控件和状态栏面板。因为这些条中的控件总是可视的，应用程序一有空闲就更新这些控件。对话框，无论是有模式还是无模式都不受影响。
- 作为一个用户，我总是不喜欢禁用的菜单项和工具栏按钮，因为知道有个命令不能用，但不知道为什么不能用，或不知道怎样使它可用。因此，我建议不要禁用一个菜单项，而是显示一个消息框，用以指示一个菜单项是不能用的，并告知怎样使它可用
- 有关用ClassWizard的例子参见例13，要更多了解MFC怎样更新用户界面，参见第 3章以及下个例子。

CD说明

在CD上执行相应的项目时，Options下的Wzd菜单命令显示无效。

6.5 例16 复选标记菜单命令

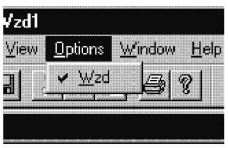

图6-9 添加一个用户界面消息处理函数，以复选框标记一个菜单命令

目标

给菜单项添加一个复选标志(见图6-9)。

策略

用ClassWizard添一个用户界面消息处理函数，它允许我们复选任何菜单项。

步骤

用ClassWizard添加一个用户界面处理函数

遵循前一个例子中的步骤，添加一个用户界面处理函数到一个应用程序类中，但现在使用CCmdUI的另一个成员函数来复选标记该菜单项。

```
void CWzdView::OnUpdateWzdType(CCmdUI* pCmdUI)
{
   pCmdUI->SetCheck(m_bWzd);
}
```

说明

- 每次打开一个菜单时，MFC允许更新菜单中每个菜单项的状态。MFC构造了一个包含CMenu对象的CCmdUI类对象，CMenu对象封装了菜单对象和用来更新菜单项所需的任何消息；然后，MFC使用与发送命令消息到类相同的方法，把CCmdUI对象传递给你，它允许在实际处理命令的相同的地方更新菜单项状态。
- CCmdUI被重载，以便为复选一个菜单项而编写的代码，也可以按下一个工具栏按钮。MFC只为代码传递一个或其他CCmdUI，以便SetCheck( )能复选一个菜单项或按下一个按钮。
- 每次打开菜单项时，菜单项被更新；每次应用程序空闲时，工具栏按钮被更新。
- 有关使用ClassWizard的例子参见例13；有关MFC怎样更新用户界面的消息，参见第3章及前一个和后一个例子。

CD说明

执行CD上的工程时，Options/Wzd命令显示一个复选标志。

6.6 例17 单选标记菜单命令

目标

在一组菜单项的边上显示一个点，用以反映一个变化模式(见图6-10)。

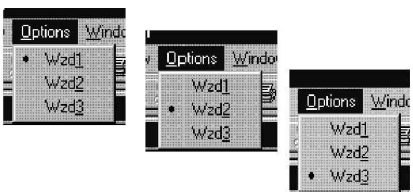

图6-10 为一组中的每一个菜单项添加一个界面消息处理函数，以单选框标记一个菜单命令

策略

使用ClassWizard，为组中的每个菜单项添加一个界面消息处理函数，它允许我们在当前菜单模式边上画一个点。

步骤

1. 用ClassWizard添加命令处理函数到一组菜单项

用ClassWizard为一组菜单项中的每个成员添加一个消息处理函数（见例13），在这些处理函数中，我们将设置以后使用的合适的模式。

```
// set mode based on what menu command was selected
void CWzdView::OnWzd1Type()
{
   m_nWzdMode=CWzdView::mode1;
}

void CWzdView::OnWzd2Type()
{
   m_nWzdMode=CWzdView::mode2;

}
void CWzdView::OnWzd3Type()
{
   m_nWzdMode=CWzdView::mode3;
}
```

2. 用ClassWizard添加用户界面处理函数到该组菜单项

遵循例15的步骤，添加一个用户界面处理函数到这组菜单项中。现在使用 CCmdUI的另一个成员函数，根据命令处理函数所设置的模式，单选标记这些菜单项。

```
// put dot next to correct menu item
void CWzdView::OnUpdateWzd1Type(CCmdUI* pCmdUI)
{
   pCmdUI->SetRadio(m_nWzdMode==CWzdView::mode1);
}

void CWzdView::OnUpdateWzd2Type(CCmdUI* pCmdUI)
{
   pCmdUI->SetRadio(m_nWzdMode==CWzdView::mode2);
}

void CWzdView::OnUpdateWzd3Type(CCmdUI* pCmdUI)
{
   pCmdUI->SetRadio(m_nWzdMode==CWzdView::mode3);
}
```

说明

- 事实上， SetRadio( )和SetCheck( )的唯一区别是，前者在每个菜单项边画个点。没有任何其他幕后处理，而在对话框中的单选按钮则是有的 —— 完全由用户决定一个组中包括哪些菜单项。
- 有关MFC怎样更新用户界面的信息参见第 3章，也可以参看前面的例子。

CD说明

在CD上执行该工程时，选择Options菜单下的一个Wzd命令，则使单选点移到它的边上。

6.7 例18 动态修改菜单

目标

应用程序运行时，添加、删除或改变菜单中的一个命令。

策略

使用CWnd::GetMenu( )访问主菜单，GetMenu( )返回指向CMenu对象的指针，它有一些成员函数，允许我们修改一个菜单。

步骤

1. 封装菜单对象到一个MFC类中

1) 首先，我们将找到应用程序主菜单对象，并把它封装到一个 CMenu对象中。

```
// Get the Main Menu
CMenu* pMainMenu = AfxGetMainWnd()->GetMenu();
```

2) 然后，通过搜索，找到要编辑的子菜单。因为菜单是用位置标识的，而不是其他 ID，因此，必须通过查找那个菜单包含的任何菜单命令 ID，找到期望的菜单。在本例中，我们将寻找一个包含菜单项ID等于IDC_WZD_TYPE的菜单。

```
CMenu* pSubMenu = NULL;
for (int i=0; i<(int)pMainMenu->GetMenuItemCount(); i++)
{
   pSubMenu = pMainMenu->GetSubMenu(i);
   if (pSubMenu && pSubMenu->GetMenuItemID(0) == IDC_WZD_TYPE)
   {
      break;
   }
}
ASSERT(pSubMenu);
```

既然已经有了一个封装所需菜单的 MFC类，可以用它的成员函数进行动态编辑该菜单。

2. 动态编辑菜单

1) 添加一个称为Wzd2，命令ID为IDC_WZD2_TYPE的菜单命令到该菜单中，可以用：

```
pSubMenu->AppendMenu(0,IDC_WZD2_TYPE,"Wzd&2");
```

2) 在Wzd2前插入Wzd3，可以用：

```
pSubMenu->InsertMenu(IDC_WZD2_TYPE,MF_BYCOMMAND,IDC_WZD3_TYPE, "Wzd&3");
```

3) 把Wzd2改变成Wzd4，可以用：

```
pSubMenu->ModifyMenu(IDC_WZD2_TYPE,MF_BYCOMMAND,IDC_WZD4_TYPE, "Wzd&4");
```

4) 删除该菜单中第二项，可以用：

```
pSubMenu->RemoveMenu(1,MF_BYPOSITION);
```

说明

注意子菜单由位置维护，而不是 ID。要查找需要被改变的子菜单，要么需要知道它的确切位置，要么必须寻找包含它的菜单命令 ID。

CD说明

在CD上执行该工程时，在 WzdView.cpp的OnWzdType( )处设置一个断点，然后单击Options/Wzd命令，观察OnWzdType修改Options菜单。

6.8 例19 动态修改系统菜单

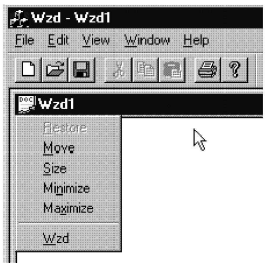

图6-11 可以添加菜单项到系统菜单

目标

添加或删除一个系统菜单命令(见图6-11)。

策略

用CWnd::GetSystemMenu( )获取系统菜单的CMenu类指针，然后用CMenu类成员函数修改系统菜单。

步骤

1. 用CMenu类封装系统菜单

可以用下面的代码封装任何窗口的系统菜单到控制该窗口的窗口类中。

```
CMenu *pSysMenu=GetSystemMenu(FALSE);
```

2. 编辑系统菜单

1) 从系统菜单中删除Close菜单项，可以用：

```
pSysMenu->RemoveMenu(SC_CLOSE,MF_BYCOMMAND);
```

2) 添加一个菜单项，本例中是Wzd，命令ID为IDC_WZD_TYPE，可以用：

```
pSysMenu->AppendMenu(0,IDC_WZD_TYPE,"&Wzd");
```

3. 处理新系统菜单项的消息

要处理最后一步中添加的Wzd命令，用ClassWizard添加一个WM_SYSCOMMAND消息处理函数到该类，可以在该类中找到该命令ID。

```
void CChildFrame::OnSysCommand(UINT nID, LPARAM lParam)
{
   if (nID==IDC_WZD_TYPE)
   {
      AfxMessageBox("Hello.");
   }

   CMDIChildWnd::OnSysCommand(nID, lParam);
}
```

说明

■ 其他标准系统菜单命令ID如下：

```
SC_SIZE
SC_MOVE
SC_MINIMIZE
SC_MAXIMIZE
SC_NEXTWINDOW
SC_PREVWINDOW
SC_RESTORE
```

■ 从系统菜单中删除Close菜单项，也使一个窗口的右上角的Close按钮变成无效。

■ 大多在应用程序最小化并位于任务栏中的时候使用系统菜单。然而，考虑到它的低效性，可能不想在系统菜单中实现太多的功能。

CD说明

在CD上执行该项目时，在 Childfrm.cpp的OnCreate( )处设置一个断点，观察子框架窗口的菜单怎样被修改；也可以在 OnSysCommand( )处设置一断点，然后单击系统菜单中新 Wzd 命令，观察它怎样被处理的。

6.9 例20 触发一个菜单命令

目标

发送一个命令到应用程序，模拟用户单击菜单或工具栏上的命令。

策略

模仿一个菜单或工具栏命令的产生，并发送一个 WM_COMMAND消息到主框架类窗口。

步骤

触发一个菜单命令

把下面的代码放到发送命令的地方，用想触发的菜单命令 ID替换ID_FILE_OPEN。

```
AfxGetMainWnd()->SendMessage(WM_COMMAND,ID_FILE_OPEN);
```

说明

■ 我们本应已经发送该命令消息到每个应用程序的窗口，然而，通过发送到主框架类窗口，该命令将被传递到整个应用程序，这已在第 3章中描述过。

■ 可以用CWnd::SendMessage( )发送任何类型的命令到任意一个窗口。参见附录 C可知怎样获得指向另一个应用程序类的指针，如果想模仿一个键盘敲击或一个鼠标命令，用 CWnd::PostMessage( )。

CD说明

在CD上执行该项目时，单击 Options/Wzd菜单命令，观察它执行与单击 File/Open命令相同的功能。

6.10 例21 创建弹出式菜单

目标

用户右击视窗时创建一个弹出式菜单 (见图6-12)。

策略

用CMenu::CreatePopupMenu( )和CMenu::TrackPopupMenu( )创建弹出式菜单；还用

CMenu的其他一些成员函数动态地用命令填充菜单。

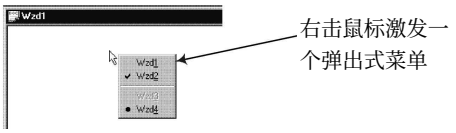

图6-12 用户可以创建一个弹出式菜单，以便用户右击时显示

步骤

1. 创建并填充一个弹出式菜单

1) 用ClassWizard为类添加一个WM_RBUTTONDOWN消息处理函数。

2) 在该新处理函数中，创建一个 CMenu类对象和一个弹出式菜单对象。

```
CMenu menu;
Menu.CreatePopupMenu();
```

3) 添加一个菜单项到该菜单，可以用：

```
menu.AppendMenu(0,IDC_WZD1_TYPE,"Wzd&1");
```

4) 添加一个边上有复选标志的命令，用：

```
menu.AppendMenu(MF_CHECKED,IDC_WZD2_TYPE,"Wzd&2");
```

5) 在该弹出式菜单中放置一个分隔符，用：

```
menu.AppendMenu(MF_SEPARATOR,0,"");
```

6) 添加一个显示灰色和无效的命令，用：

```
menu.AppendMenu(MF_GRAYED, IDC_WZD3_TYPE,"Wzd&3");
```

7) 添加单选按钮到一组菜单项，可以用：

```
menu.CheckMenuRadioItem(IDC_WZD3_TYPE, IDC_WZD4_TYPE,
   IDC_WZD4_TYPE, MF_BYCOMMAND);
```

CheckMenuRadioItem( )的前两个变量是某个范围的命令ID的起始ID和结束ID；第三个变量是用户想在边上画点的菜单项，它将清除该组中任何其他项边上的任何点；如果更喜欢用菜单位置，第四个变量可以用MF_BYPOSITION。

8) 要使一个菜单项成为默认的菜单项，使它以黑体显示，并在用户按回车时被执行，可以用：

```
::SetMenuDefaultItem(menu.m_hMenu, IDC_WZD4_TYPE,
   MF_BYCOMMAND);
```

2. 显示弹出式菜单

1) 在鼠标右击处显示该弹出式菜单，用：

```
CPoint pt;
GetCursorPos(&pt);
menu.TrackPopupMenu(TPM_RIGHTBUTTON, pt.x, pt.y, this);
```

应用程序暂停在TrackPopupMenu( )，直到用户单击该菜单或其他地方。当菜单被单击时，一个WM_COMMAND消息与命令ID一起被发送到TrackPopupMenu( )第四个变量指定的窗口。

2) 使用TrackPopupMenu( )以后，需要用DestroyMenu( )销毁弹出式菜单对象。因为该例中的CMenu对象分配在堆栈中，返回时也将销毁它。

```
menu.DestroyMenu();
```

若不想匆忙地创建一个弹出式菜单，也可以直接从应用程序资源中装入一个预定义的菜单。

3. 用一个应用程序资源创建一个弹出式菜单

1) 用菜单编辑器创建一个新菜单，并给它添加所需的菜单项。

2) 同前面一样，在堆栈中创建Menu类对象，但现在使用CMenu::LoadMenu( )从资源中装入新菜单。

```
CMenu menu;
menu.LoadMenu(IDR_WZD_MENU);            // get menu resource
CMenu* pPopup = menu.GetSubMenu(0);     // get pointer to popup menu
```

3) 禁用该弹出式菜单中的菜单项，用：

```
pPopup->EnableMenuItem(ID_POPUP_WZD1,MF_BYCOMMAND|MF_GRAYED);
```

4) 给菜单项添加一个复选标记，用：

```
pPopup->CheckMenuItem(2,MF_BYPOSITION|MF_CHECKED);
```

5) 可以使用与前面一样的方法显示和销毁该弹出式菜单。

说明

- 注意我们在这里处理两个对象： CMenu类对象和Windows弹出式菜单对象。可以认为CMenu对象是一个Windows资源的C++类封装；弹出式菜单对象实际上是一个用 #32768作为它的窗口类的弹出式窗口。有关 MFC对象和窗口对象参见第 1章。
- 注意正如一个普通菜单一样，从一个弹出式菜单中发送出来的消息是一个具有命令 ID的WM_COMMAND消息。有关消息处理参见第 3章。

CD说明

在CD上执行该工程时，在视图的任何地方右击鼠标以激发一个定制的弹出式菜单。