28

INTERFACING WITH HELP FILES

Providing Help to the User

Because Windows provides standard functionality for many common application tasks application designers can concentrate on providing extra features. As a result, applications have become extremely powerful. Unfortunately, as applications become more powerful, they also become more complicated. Providing written documentation rarely solves this problem, because typical users are not inclined to read a manual until they have reached a fairly high level of frustration.

The obvious solution to this problem is to provide your documentation in an online format. Windows provides an application called Windows Help (WINHLP32.EXE) that allows the user to easily browse through online documentation that is in the standard Help file (.HLP) format. In addition, Windows provides functions and messages that an application can use to interact with Windows Help. This interaction allows an application to provide context-sensitive help, tips, training cards, and other features not available in written documentation.

When creating the Help file, you specify the help text, bitmaps, sounds, and other resources that will be used by Windows Help. The help text consists of the actual documentation, as well as the specification of keywords, titles, browsing order, and other information. These are all combined by the Windows Help Compiler into the Help (.HLP) file that your application will reference.

Providing User Access to the Help File

Your application can use several methods to present your Help file to the user. At a minimum, your application should provide a Help menu item as the right-most item on the menu bar. This pop-up menu should have the submenu items shown in Tables 28-1 and 28-2. Notice that the Help menu item is defined differently for Windows 95 than for Windows NT and Windows 3.x. Your application can determine the host platform at run time and provide either of these menus. Figures 28-1 and 28-2 show examples of these menus.

Table 28-1. Items to Include in the Help Menu Item in Windows NT and Windows 3.x Applications

Menu Item	Description
Contents	Displays the topic identified as the contents topic in your .HLP file.
How to Use Help	Provides information to the user on general navigation through the Windows help system.
Search For Help On_	Allows the user to enter a keyword, then displays a list of topics matching that key.
About _application_	Displays a dialog box identifying the application, version, etc. This menu item should be separated from the others by a menu separator.

Table 28-2. Items to Include in the Help Menu Item in Windows 95 Applications

Menu Item	Description
Help Topics_	Brings up the Windows help index (the Help Finder).
About _application_	Displays a dialog box identifying the application, version, etc. This menu item should be separated from the others by a menu separator.

Your application should define these menu items in its resource file, and process the associated WM_COMMAND messages in its window procedure. To process the About menu item, the application typically displays a dialog box that contains the application's icon, version number, copyright information, and any other identification information. Figure 28-3 shows an example of the dialog box displayed by the Windows NOTEPAD.EXE application in response to the About menu item. The other menu items are processed by using the WinHelp() function, which provides interaction between the application and the Windows help system. Refer to the description of the WinHelp() function later in this chapter for more information.

The following is an example of a Windows 95 application that handles the Help Topics and About menu items.

```
LRESULT CALLBACK WndProc( HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam )
{
   switch( uMsg )
   {
      case WM_COMMAND :
              switch( LOWORD( wParam ) )
              {
                 case IDM_HELP_TOPICS :
                        WinHelp( hWnd, "HELPTEST.HLP", HELP_FINDER, 0 );
                        break;

                 case IDM_ABOUT :
                        DialogBox( hInst, "AboutBox", hWnd, (DLGPROC)About );
                        break;

                 case IDM_EXIT :
                        DestroyWindow( hWnd );
                        break;
              }
              break;

      case WM_DESTROY :
              // Close Help file if it is displayed.
              //.....................................
              WinHelp( hWnd, "HELPTEST.HLP",  HELP_QUIT, 0 );

              PostQuitMessage(0);
              break;

      default :
            return( DefWindowProc( hWnd, uMsg, wParam, lParam ) );
   }

   return( 0L );
}
```

Providing Context-Sensitive Help

In addition to providing a Help menu item, whenever possible your application should provide context-sensitive help. Context-sensitive help uses the WinHelp() function to display a topic that is determined based on the current state of the application. When a user requests context-sensitive help, Windows sends your application a WM_HELP message. The user requests context-sensitive help by pressing <F-1>, or clicking on the Help icon (if available) in the application's title bar, and then clicking on a menu, control, or window in your application for which to receive help. This results in a different form of the WM_HELP message being sent to your application. In either case, the lParam parameter of the WM_HELP message points to a HELPINFO structure that your application can use to determine which help topic to display. The application should processing the WM_HELP message by calling WinHelp() with the HELP_WM_HELP flag and an array of control IDs and context help IDs.

If possible, applications should also provide a pop-up menu with a What's This? menu item which displays context-sensitive help for the item the on which the right mouse button was clicked. This is done by processing the WM_CONTEXTMENU message and calling the WinHelp() function with HELP_CONTEXTMENU and the same array of control IDs and context help IDs as used with the WM_HELP message.

The following example illustrates the handling of the WM_HELP and WM_CONTEXTMENU messages in a dialog procedure to provide context-sensitive help.

Listing 28-1. Context-Sensitive Help in a Dialog Box

```
LRESULT CALLBACK HlpDialog( HWND hDlg,
                            UINT message,
                            WPARAM wParam,
                            LPARAM lParam)
{
static DWORD nIDs[] = { IDC_RADIO1, HELP_BASE+IDC_RADIO1,
                        IDC_RADIO2, HELP_BASE+IDC_RADIO2,
                        IDC_RADIO3, HELP_BASE+IDC_RADIO3,
                        IDOK,       HELP_BASE+IDOK,
                        IDCANCEL,   HELP_BASE+IDCANCEL,
                        IDHELP,     HELP_BASE+IDHELP,
                        0, 0 };

   switch (message)
   {
       case WM_HELP :
               {
                  int i = 0;
                  LPHELPINFO lpInfo = (LPHELPINFO)lParam;

                  // Make sure that the window is within our help IDs.
                  // This elliminates the "No help..." message.
                  //.............................................
                  while( nIDs[i] )
                  {
                     if ( nIDs[i] == (DWORD)GetWindowLong( lpInfo->hItemHandle, GWL_ID ) )
                     {
                        WinHelp( lpInfo->hItemHandle, "HELPTEST.HLP", HELP_WM_HELP, (DWORD)(LPVOID)nIDs );
                        break;
                     }
                     i+=2;
                  }
               }
               break;

       case WM_CONTEXTMENU :
               {
                  int i = 0;

                  // Make sure that the window is within our help IDs.
                  // This elliminates the "No help..." message.
                  //.............................................
                  while( nIDs[i] )
```

```
            {
               if ( nIDs[i] == (DWORD)GetWindowLong( (HWND)wParam, GWL_ID ) )
               {
                  WinHelp( (HWND)wParam, "HELPTEST.HLP", HELP_CONTEXTMENU, (DWORD)(LPVOID)nIDs );
                  break;
               }
               i+=2;
            }
         }
         break;
   .
   .
   .
```

Applications can use the SetMenuContextHelpID() and SetWindowContextHelpID() functions to set the context ID values for its menus, windows, and controls. These values are then passed to your application in the dwContextID member of the HELPINFO structure with the WM_HELP message.. You can query the context ID values of menus, windows, and controls by using the GetMenuContextHelpID() and GetWindowContextHelpID() functions.

Using Training Card Help

Training cards provide a more interactive approach to the Windows help system and helps the user perform complex tasks by leading them through the process. An application initiates training card help by combining the HELP_TCARD flag with another command in a call to WinHelp(). Windows help will display the requested topic, which normally will contain one or more authorable buttons that the user can select. When the user selects one of these buttons, or carries out some other action, Windows help sends a WM_TCARD message to the application, indicating the user's action. See the example under the WM_TCARD message later in this chapter for an example of an application using training card help. Help Files Function Summary
Table 28-3 summarizes the functions used to interface with help files. A detailed description of each function follows the table.

Table 28-3. Interfacing with Help Files Function Summary

Function	Purpose
GetMenuContextHelpId	Retrieves the help context ID of the specified menu.
GetWindowContextHelpId	Retrieves the help context ID of the specified window or control.
SetMenuContextHelpId	Sets the help context ID of the specified menu.
SetWindowContextHelpId	Sets the help context ID of the specified window or control.
WinHelp	Executes the Windows help engine.
Messages	
WM_HELP	Sent to an application when the user presses the <F1> key or clicks on the question mark icon in the application's title bar.
WM_TCARD	Allows interaction between an application and the Windows help engine during a training card session.

GETMENUCONTEXTHELPID — WINDOWS 95

Description	An application can use GetMenuContextHelpId() to retrieve the help context ID of the specified menu. This ID value is set using the SetMenuContextHelpId() function.
Syntax	DWORD GetMenuContextHelpId(HMENU hMenu)
Parameters	
hMenu	HMENU: Specifies the menu handle of the menu whose context ID is to be returned.

Returns	The context ID of the specified menu, if successful. Otherwise, the value zero is returned.
Include File	winbase.h
See Also	SetMenuContextHelpId()

GETWINDOWCONTEXTHELPID — WINDOWS 95

Description	An application can use GetWindowContextHelpID() to retrieve the help context ID of the specified window or control. This ID value is set using the SetWindowContextHelpId() function. A window or control that has not had its help context ID set inherits the ID of its parent.
Syntax	DWORD GetWindowContextHelpId(HWND hWindow)
Parameters	
hWindow	HWND: Specifies the window or control handle of the window whose context ID is to be returned.
Returns	The context ID of the specified window, if successful. Otherwise, the value zero is returned.
Include File	winbase.h
See Also	SetWindowContextHelpId()

SETMENUCONTEXTHELPID — WINDOWS 95

Description	An application can use this function to set the help context ID of the specified menu. This ID value is retrieved using the GetMenuContextHelpId() function. The menu context ID is available through the HELPINFO structure during processing of the WM_HELP message.
Syntax	BOOL SetMenuContextHelpId(HMENU hMenu, DWORD ContextHelpID)
Parameters	
hMenu	HMENU: Specifies the menu handle of the menu whose context ID is to be set.
ContextHelpID	DWORD: Specifies the help context ID. The menu items help context ID is set to this value.
Returns	TRUE if successful; otherwise, FALSE is returned.
Include File	winbase.h
See Also	GetMenuContextHelpId()

SETWINDOWCONTEXTHELPID — WINDOWS 95

Description	An application can use this function to set the help context ID of the specified window or control. Retrieve this ID value using the GetWindowContextHelpId() function. A window or control that has not had its help context ID set inherits the ID of its parent. The window or controls context ID is available through the HELPINFO structure during processing of the WM_HELP message.
Syntax	BOOL SetWindowContextHelpId(HWND hWindow, DWORD ContextHelpID)
Parameters	
hWindow	HWND: Specifies the window or control handle of the window whose context ID is to be returned.
ContextHelpID	DWORD: Specifies the help context ID. The window or control items help context ID is set to this value.
Returns	TRUE if successful; otherwise, FALSE is returned.
Include File	winbase.h
See Also	GetWindowContextHelpId()

Description The WinHelp() function provides interaction between the Windows help system and the application.

Syntax BOOL WinHelp(HWND hWnd, LPCTSTR lpszHelpFile, UINT uCommand, DWORD dwData)

Parameters

hWnd HWND: Specifies the window handle of the window requesting help. This is normally the application's main window, except when uCommand is HELP_CONTEXTMENU or HELP_WM_HELP, in which case it is the ID of the control or window about which help is requested.

lpszHelpFile LPCTSTR: Specifies the name of the application's .HLP file. The file name may have a greater-than sign (>) and window name specified after it, in which case the help is displayed in a secondary window of the type specified by the window name. This secondary window name must have been specified in the WINDOWS section of the .HPJ file.

uCommand UINT: Specifies a command to the Windows help system. This command may be any of the values listed in Table 28-4.

dwData DWORD: Provides additional data as required by the uCommand parameter.

Table 28-4. Available Values for the uCommand Parameter of WinHelp()

uCommand	Description
HELP_COMMAND	Executes the identified help macros. The dwData parameter points to a string of help macros, separated by semicolons. If a macro has both a short name and a long name, use the short name.
HELP_CONTENTS	Causes Windows help to display the topic that is identified as the contents topic in the .HPJ file. This is typically used by Windows NT and Windows 3.x applications when the user clicks on the Help Contents menu item. Windows 95 applications should use the HELP_FINDER command instead of HELP_CONTENTS.
HELP_CONTEXT	Causes Windows help to display the topic that is associated with the 32-bit context ID specified by the dwData parameter.
HELP_CONTEXTMENU	Causes Windows help to display the help menu for the selected window, and then display the associated help topic in a pop-up window. The dwData parameter points to an array of DWORD pairs. The first DWORD in each pair is the identifier of a window or control, and the second DWORD is the context ID of the associated topic.
HELP_CONTEXTPOPUP	Causes Windows help to display the topic that is associated with the 32-bit context ID specified by the dwData parameter. The topic is displayed in a pop-up window.
HELP_FINDER	Displays the help file topic list.
HELP_FORCEFILE	Ensures that Windows help is displaying the proper .HLP file. If the specified .HLP file is not being displayed, Windows help opens it. Otherwise, no action is performed.
HELP_HELPONHELP	Displays the WINHELP.HLP help file, providing help on using the Windows help system. This is typically used by Windows NT and Windows 3.x applications when the user clicks on the Help On Using Help menu item. Windows 95 applications should use the HELP_FINDER command instead of HELP_HELPONHELP.
HELP_INDEX	Displays the help file index. This is typically used by Windows NT and Windows 3.x applications when the user clicks on the Help Index menu item. Windows 95 applications should use the HELP_FINDER command instead of HELP_INDEX.
HELP_KEY	Searches the .HLP file for topics that specify a keyword matching the keyword pointed to by the dwData parameter. If more than one topic matches the keyword, the matching topics are listed in a dialog box.
HELP_MULTIKEY	Searches the .HLP file for topics that specify a keyword matching the given keyword. The dwData parameter points to a MULTIKEYHELP structure that identifies the keyword table to search and the keyword string.
HELP_PARTIALKEY	Searches the .HLP file for topics that specify a keyword matching the keyword pointed to by the dwData parameter. If more than one topic matches the keyword, the index tab is displayed. You can force Windows help to display the index tab by specifying an empty string as the keyword.

HELP_QUIT	Informs the Windows help system that the application has finished using help. The .HLP file will be closed, and the help window will be removed.
HELP_SETCONTENTS	Sets the context identifier for the contents topic. Windows help will display this topic in response to a HELP_CONTENTS command or when the user clicks the Contents button in the help window. The dwData parameter specifies a 32-bit context ID for the topic that is to become the contents topic.
HELP_SETINDEX	Specifies which keyword table is to be displayed in the Help Topics dialog box.
HELP_SETWINPOS	Sets the size and position of the main help window, or a secondary window. The dwData parameter points to a HELPWININFO structure.
HELP_TCARD	Initiates training card help. This command cannot be used alone, but must be combined with one of the other help commands using the bitwise OR operation.
HELP_WM_HELP	An application can use this command in response to a WM_HELP message. In this case, the hWnd parameter should be the handle of the control as passed in the HELPINFO structure. The dwData parameter points to an array of DWORD pairs. The first DWORD of each pair specifies a control identifier, and the second DWORD specifies the context ID of a help topic to be displayed for that control identifier.

Returns	TRUE if it is successful; otherwise, FALSE is returned.
Include File	winuser.h
See Also	GetMenuContextHelpID(), GetWindowContextHelpID(), SetMenuContextHelpID(), SetWindowContextHelpID()
Related Messages	WM_HELP, WM_TCARD
Example	See Listing 28-1 at the beginning of this chapter for an example of this function.

WM_HELP

WINDOWS 95

Description	WM_HELP is a message sent to an application when the user requests context-sensitive help. This can happen when the user presses the <F1> key or clicks on the Help icon (if available) on the application's title bar, and then clicks on a control, window, or menu item. Normally, an application responds to this message by issuing the HELP_WM_HELP command with the WinHelp() function, using information in the HELPINFO structure.
Parameters	
lParam	LPHELPINFO: Points to a HELPINFO structure identifying the help request.
Returns	BOOL: An application should return TRUE if it processes this message.
See Also	WinHelp()
Related Messages	WM_TCARD
Example	See Listing 28-1 at the beginning of this chapter for an example of this message.

WM_TCARD

WINDOWS 95

Description	The Windows help system sends WM_TCARD to the application when the user clicks on an authorable button that is associated with the Tcard() help macro or performs some other action within a training card session of Windows help.
Parameters	
wParam	WPARAM: Identifies the action that the user has taken. This may be any of the values listed in Table 28-5.
lParam	LPARAM: Specifies additional data depending on the value of wAction.

Table 28-5. Values for the wAction Parameter

wAction value	Description
HELP_TCARD_DATA	The user has clicked an authorable button. The dwData parameter contains a 32-bit value specified by the help author.
HELP_TCARD_NEXT	The user has c licked an authorable Next button.
HELP_TCARD_OTHER_CALLER	A different application has requested training card help.
IDABORT	The user has c licked an authorable Abort button.
IDCANCEL	The user has c licked an authorable Cancel button.
IDCLOSE	The user has c losed the training card help session.
IDHELP	The user has c licked an authorable Help button.
IDIGNORE	The user has c licked an authorable Ignore button.
IDOK	The user has c licked an authorable OK button.
IDNO	The user has c licked an authorable No button.
IDRETRY	The user has c licked an authorable Retry button.
IDYES	The user has c licked an authorable Yes button.

Returns LRESULT: An application should return zero.

See Also WinHelp()

Related Messages WM_HELP

Example The following example initiates a training card help session, and responds to the WM_TCARD messages IDOK, IDCANCEL, and IDABORT. The example also processes the HELP_TCARD_DATA message by checking the appropriate radio button based on the users selection in the training session.

```
LRESULT CALLBACK HlpDialog( HWND hDlg,
                            UINT message,
                            WPARAM wParam,
                            LPARAM lParam)
{
   switch (message)
   {
        case WM_TCARD :
               switch( wParam )
               {
                  case IDOK :
                         EndDialog( hDlg, IDOK );
                         break;

                  case IDCANCEL :
                         EndDialog( hDlg, IDCANCEL );
                         break;

                  case IDABORT :
                         EnableWindow( hDlg, TRUE );
                         break;

                  case HELP_TCARD_DATA :
                         CheckRadioButton( hDlg, IDC_RADIO1, IDC_RADIO3, lParam );
                         break;
               }
               break;

       case WM_COMMAND:
               switch( LOWORD( wParam ) )
               {
                  case IDOK :
                         EndDialog( hDlg, IDOK );
                         break;

                  case IDCANCEL :
```

```
                        EndDialog( hDlg, IDCANCEL );
                        break;

                   case IDHELP :
                        WinHelp( hDlg, "HELPTEST.HLP", HELP_TCARD | HELP_CONTEXT, IDH_TRAINING );
                        EnableWindow( hDlg, FALSE );
                        break;
               }
               break;
   }

   return (FALSE);
}
```