A Generic Win32 Sample Application

This section introduces the source code components of a generic Win32-based application, named GENERIC. They closely resemble their counterparts in a generic 16-bit Windows-based application. This section assumes that you have used Win32-based applications and therefore are already familiar with windows, menus, and dialog boxes.

The GENERIC application described here consists of the following parts:

- The entry-point function
- The menu
- The window procedure
- The About dialog box

The complete code is shown in Source Code.

The Entry-Point Function

Every Win32-based application must have an entry-point function. The name commonly given to the entry point is the **WinMain** function.

As in most Win32-based applications, the **WinMain** function for the GENERIC application completes the following steps:

- Registering the window class
- Creating the main window
- Entering the message loop

Registering the Window Class

Every window must have a window class. A window class defines the attributes of a window, such as its style, its icon, its cursor, the name of the menu, and the name of the window procedure.

The first step is to fill in a **WNDCLASS** structure with class information.

Next, you pass the structures to the **RegisterClass** function. The GENERIC application registers the GenericAppClass window class as follows:

```
wc.lpszClassName = "GenericAppClass";
wc.lpfnWndProc = MainWndProc;
wc.style = CS_OWNDC | CS_VREDRAW | CS_HREDRAW;
wc.hInstance = hInstance;
wc.hIcon = LoadIcon( NULL, IDI_APPLICATION );
wc.hCursor = LoadCursor( NULL, IDC_ARROW );
wc.hbrBackground = (HBRUSH)( COLOR_WINDOW+1 );
wc.lpszMenuName = "GenericAppMenu";
wc.cbClsExtra = 0;
wc.cbWndExtra = 0;

RegisterClass( &wc );
```

For more information on the menu, see The Menu. For more information on the window procedure, see The Window Procedure.

Creating the Main Window

You can create the window by calling the **CreateWindow** function. The GENERIC application creates the window as follows:

```
hWnd = CreateWindow( "GenericAppClass",
   "Generic Application",
   WS_OVERLAPPEDWINDOW|WS_HSCROLL|WS_VSCROLL,
   0,
   0,
   CW_USEDEFAULT,
   CW_USEDEFAULT,
   NULL,
   NULL,
   hInstance,
   NULL
);
```

The first parameter is the name of the class that we registered. The remaining parameters specify other window attributes. This call creates the window, but Windows does not display a window until the application calls the **ShowWindow** function. The GENERIC application displays the

window as follows:

```
ShowWindow( hWnd, nCmdShow );
```

Entering the Message Loop

Once the main window is created and displayed, the **WinMain** function can being its primary task, which is to read messages from the application queue and dispatch them to the appropriate window.

Windows does not send input directly to an application. Instead, it places all mouse and keyboard input from the user into a message queue, along with messages posted by Windows and other application. The application must read the message queue, retrieve the messages, and dispatch them so that the window procedure can process them.

The GENERIC application uses the following message loop:

```
while( GetMessage( &msg, NULL, 0, 0 ) ) {
   TranslateMessage( &msg );
   DispatchMessage( &msg );
}
```

The **GetMessage** function retrieves a message from the queue. The **DispatchMessage** function sends each message to the appropriate window procedure. The **TranslateMessage** function translates virtual-key message into character messages. In GENERIC, this is necessary to implement menu access keys.

The Menu

Most applications include a menu to provide a means for the user to select commands. The most common way to create a menu is to define it as a resource in the resource-definition file. The GENERIC application has a single menu, named Help, with a single command, About. The resource is defined as follows:

```
GenericAppMenu MENU
{
```

```
   POPUP "&Help"
   {
      MENUITEM "&About",          IDM_ABOUT
   }
}
```

The name of the menu resource is specified when registering the window class.

Selecting the About command causes GENERIC to display the about dialog box.

The Window Procedure

Every window must have a window procedure. The name of the window procedure is user-defined. The GENERIC application uses the following window procedure for the main window:

```
LRESULT WINAPI MainWndProc( HWND, UINT, WPARAM, LPARAM );
```

The WINAPI modifier is used because the window procedure must be declared with the standard call calling convention.

The window procedure receives messages from Windows. These may be input messages or window-management messages from Windows. You can optionally handle a message in your window procedure or pass the message to Windows for default processing by calling the **DefWindowProc** function. The GENERIC application processes the WM_PAINT, WM_COMMAND, and WM_DESTROY messages, using a switch statement that is structured as follows:

```
switch( msg ) {
      case WM_PAINT:
      ...
      case WM_COMMAND:
      ...
      case WM_DESTORY:
      ...
      default:
         return( DefWindowProc( hWnd, msg, wParam, lParam
```

The WM_PAINT message indicates that you should redraw what's in all or

part of your application's window. Use the **BeginPaint** function to get a handle to a device context, then use the device context for drawing within the application's window, with functions like **TextOut**. Use **EndPaint** to release the device context. The GENERIC application displays a text string, "Hello, World!", in the window using the following code:

```
case WM_PAINT:
      hDC = BeginPaint( hWnd, &ps );

      TextOut( hDC, 10, 10, "Hello, World!", 13 );

      EndPaint( hWnd, &ps );
      break;
```

The WM_COMMAND message indicates that a user has selected a command item from a menu. The GENERIC application uses the following code to check if its About menu item has been selected:

```
case WM_COMMAND:
         switch( wParam ) {
            case IDM_ABOUT:
            ...
            break;
         }
```

Most window procedures process the WM_DESTROY message. Windows sends this message to the window procedure immediately after destroying the window. The message gives you the opportunity to finish processing and post a WM_QUIT message in the application queue. The GENERIC application handles the WM_DESTROY message as follows:

```
case WM_DESTROY:
   PostQuitMessage( 0 );
   break;
```

The About Dialog Box

A dialog box is a temporary window that displays information or prompts the user for input. The GENERIC application includes an About dialog box. Every application should include an About dialog box. The dialog box displays such information as the application's name and copyright information.

You create and display a dialog box by using the **DialogBox** function. This function takes a dialog box template and creates a dialog box.

The GENERIC application includes the following dialog box template in the resource-definition file:

```
AboutDlg DIALOG FIXED 6, 21, 198, 99
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION |
CAPTION "About Generic"
FONT 8, "MS Shell Dlg"
BEGIN
    DEFPUSHBUTTON   "&OK", IDOK, 72, 74, 40, 14
    LTEXT           "Generic Application", 104, 45, 14,
                    128, 8
    LTEXT           "Written as a sample", 105, 45, 35, 59
    LTEXT           "Microsoft Corporation", 106, 45, 45,
    LTEXT           "Copyright (c) 1996", 107, 45,
                    54, 138, 8
END
```

The name of the source is specified as AboutDlg. For more information, see Dialog Resource.

When the user clicks About from the Help menu, the following code in the window procedure displays the About dialog box:

```
case WM_COMMAND:
   switch( wParam ) {
      case IDM_ABOUT:
         DialogBox( ghInstance, "AboutDlg", hWnd, (DLGPROC
                    AboutDlgProc );
      break;
   }
break;
```

The last parameter is a pointer to a dialog box procedure. It has the following prototype.

```
LRESULT WINAPI AboutDlgProc( HWND, UINT, WPARAM, LPARAM );
```

A dialog box procedure is similar to a window procedure, but usually processes only dialog initialization and user-input messages. The GENERIC application contains the following message processing code:

```
switch( uMsg ) {
```

```
   case WM_INITDIALOG:
      return TRUE;
   case WM_COMMAND:
      switch( wParam ) {
         case IDOK:
            EndDialog( hDlg, TRUE );
            return TRUE;
      }
   break;
}

return FALSE;
```

Source Code

The GENERIC application consists of the following files:

- GENERIC.C
- GENERIC.H
- GENERIC.RC
- GENERIC.DLG

GENERIC.C

GENERIC.C contains code for the GENERIC application. It includes GENERIC.H.

```
/*****************************************************
*  generic.c: Source code for generic
*
*  Comments: Generic Win32-based Application
*
*  Functions:
*     WinMain      - Application entry point
*     MainWndProc  - main window procedure
*     AboutDlgProc - dialog procedure for About dialog
*
*
\*****************************************************
```

```
/*********************  Header Files  ********************

#include <windows.h>
#include "generic.h"

/*********************  Prototypes  **********************

LRESULT WINAPI MainWndProc( HWND, UINT, WPARAM, LPARAM );
LRESULT WINAPI AboutDlgProc( HWND, UINT, WPARAM, LPARAM );

/*******************  Global Variables *******************

HANDLE ghInstance;

/*********************************************************
*  Function: int PASCAL WinMain(HINSTANCE, HINSTANCE, LPST
*
*   Purpose: Initializes Application
*
*  Comments: Register window class, create and display the
*            window, and enter message loop.
*
*
\*********************************************************

int PASCAL WinMain( HINSTANCE hInstance,
    HINSTANCE hPrevInstance,
    LPSTR lpszCmdLine,
    int nCmdShow )
{
   WNDCLASS wc;
   MSG msg;
   HWND hWnd;

   if( !hPrevInstance )
   {
      wc.lpszClassName = "GenericAppClass";
      wc.lpfnWndProc = MainWndProc;
      wc.style = CS_OWNDC | CS_VREDRAW | CS_HREDRAW;
      wc.hInstance = hInstance;
      wc.hIcon = LoadIcon( NULL, IDI_APPLICATION );
      wc.hCursor = LoadCursor( NULL, IDC_ARROW );
      wc.hbrBackground = (HBRUSH)( COLOR_WINDOW+1 );
      wc.lpszMenuName = "GenericAppMenu";
      wc.cbClsExtra = 0;
      wc.cbWndExtra = 0;
```

```
      RegisterClass( &wc );
   }

   ghInstance = hInstance;

   hWnd = CreateWindow( "GenericAppClass",
      "Generic Application",
      WS_OVERLAPPEDWINDOW|WS_HSCROLL|WS_VSCROLL,
      0,
      0,
      CW_USEDEFAULT,
      CW_USEDEFAULT,
      NULL,
      NULL,
      hInstance,
      NULL
   );

   ShowWindow( hWnd, nCmdShow );

   while( GetMessage( &msg, NULL, 0, 0 ) ) {
      TranslateMessage( &msg );
      DispatchMessage( &msg );
   }

   return msg.wParam;
}

/****************************************************
* Function: LRESULT CALLBACK MainWndProc(HWND, UINT, WPARA
*
*  Purpose: Processes Application Messages
*
* Comments: The following messages are processed
*
*           WM_PAINT
*           WM_COMMAND
*           WM_DESTROY
*
*
\****************************************************

LRESULT CALLBACK MainWndProc( HWND hWnd, UINT msg, WPARAM
   LPARAM lParam )
{
```

```
   PAINTSTRUCT ps;
   HDC hDC;

   switch( msg ) {

/*******************************************************
*     WM_PAINT:
\*******************************************************

      case WM_PAINT:
         hDC = BeginPaint( hWnd, &ps );

         TextOut( hDC, 10, 10, "Hello, World!", 13 );

         EndPaint( hWnd, &ps );
         break;

/*******************************************************
*     WM_COMMAND:
\*******************************************************

      case WM_COMMAND:
         switch( wParam ) {
            case IDM_ABOUT:
               DialogBox( ghInstance, "AboutDlg", hWnd, (D
                          AboutDlgProc );
            break;
         }
      break;

/*******************************************************
*     WM_DESTROY: PostQuitMessage() is called
\*******************************************************

      case WM_DESTROY:
         PostQuitMessage( 0 );
         break;

/*******************************************************
*     Let the default window proc handle all other message
\*******************************************************

      default:
         return( DefWindowProc( hWnd, msg, wParam, lParam
   }
```

```
   return 0;
}

/*******************************************************
* Function: LRESULT CALLBACK AboutDlgProc(HWND, UINT, WPAR
*
*  Purpose: Processes "About" Dialog Box Messages
*
* Comments: The About dialog box is displayed when the use
*           About from the Help menu.
*
\*******************************************************

LRESULT CALLBACK AboutDlgProc( HWND hDlg, UINT uMsg, WPARA
{
   switch( uMsg ) {
      case WM_INITDIALOG:
         return TRUE;
      case WM_COMMAND:
         switch( wParam ) {
            case IDOK:
               EndDialog( hDlg, TRUE );
               return TRUE;
         }
      break;
   }

   return FALSE;
}
```

GENERIC.H

GENERIC.C contains header information for the GENERIC application. It is included in GENERIC.C and GENERIC.RC.

```
/*******************************************************
* generic.h: Header file for Generic
*
*
\*******************************************************

/******* Menu Defines *******/
```

```
#define IDM_ABOUT      1000
```

GENERIC.RC

GENERIC.RC contains resource information for the GENERIC application. The dialog resources are included in GENERIC.DLG.

```
/*********************************************************
* generic.rc: Resource script for Generic
*
*
\*********************************************************

#include <windows.h>
#include "generic.h"
#include "generic.dlg"

GenericAppMenu MENU
{
   POPUP "&Help"
   {
      MENUITEM "&About",          IDM_ABOUT
   }
}
```

GENERIC.DLG

GENERIC.DLG defines the About dialog box for the GENERIC application. This file is included in GENERIC.RC.

```
/*********************************************************
* generic.dlg: Dialogs for Generic
*
*
\*********************************************************

1 DLGINCLUDE "generic.h"

AboutDlg DIALOG FIXED 6, 21, 198, 99
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION |
CAPTION "About Generic"
```

```
FONT 8, "MS Shell Dlg"
BEGIN
    DEFPUSHBUTTON   "&OK", IDOK, 72, 74, 40, 14
    LTEXT           "Generic Application", 104, 45, 14,
                    128, 8
    LTEXT           "Written as a sample", 105, 45, 35, 59
    LTEXT           "Microsoft Corporation", 106, 45, 45,
    LTEXT           "Copyright (c) 1996", 107, 45,
                    54, 138, 8
END
```