µ §

Scott Skorupa
MS Windows Developer Support

Version 1.0
Rev. Date:

What is an Embedded Window?

An embedded window is a fixed-size window displayed within a winhelp topic. A DLL, which needs to accompany the help file, controls the embedded window. This ability allows the help author to display items that the help engine is not normally capable of displaying, such as 256 color bitmaps.

The help engine creates embedded windows as child windows. These special windows receive most of the normal messages that a window would, including mouse and paint messages. However, there are some limitations to embedded windows, such as not having the ability to take the input focus. The side effect of this limitation is that the embedded window cannot process keystrokes.

Placing an Embedded Window in an RTF file

In order to use an embedded window in a help file, there needs to be a facility to add an embedded window to the source RTF document. The method for adding embedded windows is similar to that of inserting a bitmap by reference. To insert an embedded window to the RTF source file of a help topic, the author needs to add the following line where the embedded window is to appear:

{ew[l,c,r] DLL-name, window-class, author-data}

parameters **ewl**
Denotes a left justified embedded window. The topic text will wrap on the right hand side of the embedded window.

ewc
Denotes a character justified embedded window. The base of the window will line up with the baseline of the characters before and after the reference, creating an "inline" effect.

ewr
Denotes a right justified embedded window. The topic text will wrap on the left hand side of the embedded window.

DLL-name
The name of the DLL that controls the embedded window.

window-class
The class name of the embedded window, registered in the DLL

author-data
Any data that the author wishes to pass to the DLL The string can contain any punctuation mark except for a comma, and is parsed by the

DLL.

For example, the following are valid references to embedded windows:

```
{ewl EMBED, embedclass, "face" }
{ewr BMPDLL, BMP256, c:\windows\256color.bmp}
```

while this next references is not valid:

```
{ewl EMBED, embedclass, face, face2}
```

Initialization

Overview

When the help engine is going to display a topic that contains an embedded window, it first must load the DLL that controls the embedded window. This results in the calling of LibMain for the DLL. In LibMain, the DLL needs to register the window class of the embedded window. Note that this class name needs to match the "window-class" parameter in the ewx reference in the RTF source file. Also, the window class needs to have the CS_GLOBALCLASS bit set, so that the help engine can find the window class when it creates the window. Consult the Windows 3.1 SDK documentation for more information on the RegisterClass API, and the WNDCLASS structure. Once the library loads, the help engine creates the embedded window as a child window of the current topic window, which results in a WM_CREATE message.

WM_CREATE Message

The window procedure for the embedded window receives a WM_CREATE message when the help engine creates the embedded window. The lParam associated with this message contains a pointer to a CREATESTRUCT (see Appendix for details) structure. The lpCreateParams entry of CREATSTRUCT will be pointing to an EWDATA structure that looks like:

```
typedef struct {
        short idMajVersion;
        short idMinVersion;
        LPSTR szFileName;
        LPSTR szAuthorData;
        HANDLE hfs;
        DWORD coFore;
        DWORD coBack;
} EWDATA;
```

The idMajVersion and idMinVersion members hold the major and minor numbers of the help engine. An embedded window should check the value idMajVersion and make sure that it is equal to the value in which the Embedded window is written. If this value is equal, then the embedded window should check to see if the idMinVersion member of the structure is greater than or equal to the idMinVersion that the embedded window was authored. If the validation of "major" and/or "minor" versions fail, the embedded window should return negative one from the WM_CREATE message processing. The reasoning for this is that if changes are made to the help engine that would break older embedded windows, then the "Major" version number would be incremented. If functionality was added to embedded windows, then the "Minor" version number would be incremented. For the windows help engine version 3.1 both of these values will be zero. The szFileName entry is a pointer to a NULL terminated string that contains the fully qualified path to the .HLP file that contains the embedded window. The szAuthorData pointer, points to a NULL terminated string that contains the string passed as the "Author-Data" from the ewx command. The coFore and coBack are the foreground and background colors of the current help topic window.

Use the SetWindowLong API to change the window style while processing the WM_CREATE message. For instance, the following adds a border to the embedded window:

```
SetWindowLong (  hWnd, GWL_STYLE,
   GetWindowLong( hWnd, GWL_STYLE ) | WS_BORDER);
```

Painting

WM_PAINT message

Embedded windows receive WM_PAINT messages just as any other window. In response to the WM_PAINT message, the embedded window paints itself as if it were a normal window. If the embedded window needs to use a palette in order to paint, it must be request the palette from the help engine. This can be done by passing an EWM_ASKPALETTE message to the help engine as follows:

```
hpal = (HPALETTE)SendMessage(GetParent(hwnd), EWM_ASKPALETTE, 0, 0L);
```

The return value of the SendMessage will contain the handle to the palette that the help engine wants the embedded window to select.

Embedded Window Messages

When the help engine needs information from an embedded window, it sends the embedded window's window procedure a message. For the windows 3.1 help engine, there are currently three messages: one to determine the size of the embedded window; one to get the palette of the embedded window; and one for getting the "display" of the embedded window.

EWM_QUERYSIZE

When the help engine needs to display the embedded window, it needs to know the size of the embedded window. To get this information, the help engine sends the window procedure for the embedded window an EWM_QUERYSIZE message. When the embedded window receives this message, the wParam is a handle to the device context for use as a reference. The lParam parameter is a pointer to a POINT structure. An embedded window returns its size by filling out the x and y entries of the point structure. If the embedded window processes this message, then it should return the a value of 1L.

EWM_ASKPALETTE

The help engine sends the EWM_ASKPALETTE message to the window procedure of the embedded window when the help engine needs to determine a palette. When the window procedure receives this message, it should return the handle to the preferred palette. If multiple embedded windows are being displayed, the help engine will typically use the palette of the first window displayed.

If an embedded needs to use a palette when painting itself, it should send this message to the help engine, which will return a handle to the palette desired.

The wParam and lParam values associated with this message are not used.

EWM_RENDER

The help engine sends an EWM_RENDER message to an embedded window when the information displayed needs to be copied to the clipboard or to a printer. The wParam associated with the EWM_RENDER contains the clipboard format in which the help engine desires. The value of wParam can be either CF_TEXT, for copying to the clipboard, or CF_BITMAP, for printing to the printer. If the help engine is requesting the CF_TEXT format, then the embedded window procedure should return a global handle to a NULL terminated ASCII string. If the CF_BITMAP format is being requested, then the window procedure should return a handle to the bitmap. In either case, the help engine frees the appropriate handle.

In order to create a bitmap, the embedded window needs to have a reference DC. This DC is obtained from the RENDERINFO structure passed with the EWM_RENDER message.

RENDERINFO structure

When the EWM_RENDER message is sent to the embedded window procedure, the lParam points to a RENDERINFO structure that looks like the following:

```
typedef struct {
        RECT rc;
        HDC hdc;
} RENDERINFO;
```

The hdc in this structure should be used for creating the bitmap to be returned to the help engine. In order to get the size of the current embedded window, it is common practice for an embedded window to send an EWM_QUERYSIZE message to itself. This removes the need for a global variable to keep track of the size of the window.

Memory Management

When creating embedded windows, it may become necessary to associate blocks of memory with a particular embedded window. For example, within one help topic the same embedded window may be created twice, but each of the windows may be created with different parameters passed for the author-data field. Each embedded window may need to keep track of the author-data that in which the window is associated. Since both of these windows use the same DLL, the parameter can not simply be set to a global variable. To work around this problem, allocate a block of global memory, and store the handle to this memory in the "window extra bytes" for later reference, by using the SetWindowWord API. If this data needs to be retrieved, it can be accessed by using GetWindowWord. Be sure to free this memory at a later time, such as in the processesing of the WM_DESTROY message.

Sample Code

The included sample code demonstrates a basic embedded window, of class "embedclass". When the embedded window itself is included within a help topic, and the value "face" (without the quotation marks) is used for the Author-Data parameter for the embedded window, then a small bitmap of a face will be placed in the window. If the Author Data contains any other value than "face", an error message is displayed, stating that the embedded window is not available.

Some Miscellaneous notes about the sample code:

The sample code allocates a GLOBALDATA structure upon processing the WM_CREATE message, and associates this structure with the window, through the use of SetWindowWord. In this case, the structure is initialized to the values found in the EWDATA structure, that is passed with the WM_CREATE message. The EWDATA values are not used anywhere else in the program, the copying is only performed for demonstrative purposes. In the GLOBALDATA structure there is flag that is set if the "face" parameter is properly passed, otherwise this flag is cleared.

The face bitmap is a known size, 32x32 pixels. This size is used in all cases, no matter what resolution the device on which it is going to be displayed. This will result in different "text" wrapping from display driver to display driver, as well as to the printer. One possible way to work around this problem would be to keep track of the size of the window in logical coordinates, like 1x1 inches, and then stretch the bitmap to that "logical" size. This process has been left as an exercise for the reader.

µ §

CREATESTRUCT

```
typedef struct tagCREATESTRUCT {    /* cs */
    void FAR* lpCreateParams;
    HINSTANCE hInstance;
    HMENU     hMenu;
    HWND      hwndParent;
    int       cy;
    int       cx;
    int       y;
    int       x;
    LONG      style;
    LPCSTR    lpszName;
    LPCSTR    lpszClass;
    DWORD     dwExStyle;
} CREATESTRUCT;
```

The CREATESTRUCT structure defines the initialization parameters passed to the window procedure of an application.

Member **lpCreateParams**
Points to data to be used for creating the window.

hInstance
Identifies the module-instance handle of the module that owns the new window.

hMenu
Identifies the menu to be used by the new window.

hwndParent
Identifies the window that owns the new window. This member is NULL if the new window is a top-level window.

cy
Specifies the height of the new window.

cx
Specifies the width of the new window.

y
Specifies the y-coordinate of the upper-left corner of the new window. Coordinates are relative to the parent window if the new window is a child window. Otherwise, the coordinates are relative to the screen origin.

x

Specifies the x-coordinate of the upper-left corner of the new window. Coordinates are relative to the parent window if the new window is a child window. Otherwise, the coordinates are relative to the screen origin.

style
Specifies the style for the new window.

lpszName
Points to a null-terminated string that specifies the name of the new window.

lpszClass
Points to a null-terminated string that specifies the class name of the new window.

dwExStyle
Specifies extended style for the new window.

Embedded Window Data Structure (EWDATA)

```
typedef struct {
        short idMajVersion;
        short idMinVersion;
        LPSTR szFileName;
        LPSTR szAuthorData;
        HANDLE hfs;
        DWORD coFore;
        DWORD coBack;
} EWDATA;
```

The EWDATA structure is pointed to by the lpCreateParams entry of the CREATESTRUCT structure, which is passed to the embedded window in the WM_CREATE message.

Members: **idMajVersion**
Identifies the "Major" version of the help engine. This will always be 0 for the 3.1 help engine.

idMinVersion
Identifies the "Minor" version of the help engine. This will always be 0 for the 3.1 help engine.

szFileName
Points to the fully-qualified name of the .HLP file containing the embedded window.

szAuthorData
Points to the author-data parameter specified by the ewl, ewc, or ewr command from the source RTF file.

hfs
Specifies a handle to the file system for the .HLP file.

CoForeground
Specifies the foreground color of the main help window.

CoBackground
Specifies the background color of the main help window.

RENDERINFO Structure

```
typedef struct {
        RECT rc;
        HDC hdc;
 } RENDERINFO;
```

The RENDERINFO structure is passed to the Embedded Window as a pointer stored in lParam when the EWM_RENDER message is sent.

Members: **rc**
Rectangle

hdc
The device context in which the bitmap or text uses as a reference.

Messages

EWM_RENDER (0x706A)
The Windows 3.1 help engine will send this message to an embedded window when information is needed for rendering the display of the window to the clipboard or the printer.

Parameters **wParam**
This value is either CF_TEXT, or CF_BITMAP depending on what type of rendering the help engine would like performed

lParam
Points to a RENDERINFO structure

Return Value The LOWORD of the return value is dependent on the wParam parameter.

wParam	**Return Value**
CF_TEXT	Global handle to a null-terminated ASCII string that represents the output of the embedded window.
CF_BITM	Handle to a bitmap that represents the output of the

EWM_QUERYSIZE (0x706B)

The Windows 3.1 help engine will send this message to the embedded window when information is needed about the size of the embedded window.

Parameters **wParam**

A handle to the device context for the Window in which the embedded window is to be drawn.

lParam

A pointer to a POINT structure. Fill in the x field of this structure with the embedded window's width, fill in the y field of this structure with the window's height.

Return Value Return 1 if the message is processed, 0 otherwise.

EWM_ASKPALETTE (0x706C)

The Windows 3.1 help engine will send this message to the embedded window when the palette that the embedded window uses is needed.

Parameters **wParam**

Not Used.

lParam

Not Used.

Return Value **The handle to the palette being used by the embedded window.**