CHAPTER 1

Using the DrawDib Functions

The DrawDib functions provide much of the functionality of **StretchDIBits** and adds ICM support, improved stretching capabilities, and improved support of low-end display adapters. If a display driver cannot stretch an image, the DrawDib functions can stretch it more efficiently then **StretchDIBits**. The DrawDib functions also efficiently dither true-color images to 256 colors, and dither 8-bit images on 16-color VGA displays. These functions significantly improve the speed and quality of displaying such images on display adapters with limited capabilities.

This chapter discusses the following topics:

- Drawing with the DrawDib functions
- Optimizing **DrawDibDraw**
- Profiling the display characteristics

The DrawDib functions and constants are defined in DRAWDIB.H. The ICMAPP sample application shows how your application can use these functions.

Drawing With the DrawDib Functions

Your application uses the following functions with 8, 16, and 24 bit images to access the basic DrawDib services:

DrawDibOpen

This function opens a DrawDib context for drawing.

DrawDibClose

This function closes a DrawDib context and cleans up.

DrawDibDraw

This function draws a device independent bitmap to the screen.

While your application can use **DrawDibDraw** as an almost one-to-one replacement for **StretchDIBits**, it has the following limitations:

- the DIB must have the DIB_RGB_COLORS format
- your application must use the simplest transfer mode, SRC_COPY
  That is, your application cannot use **DrawDibDraw** to, say, XOR a picture with the screen.

Prior to using **DrawDibDraw**, your application must initialize the DrawDib library and

let it allocate the memory it needs by calling **DrawDibOpen**. This function returns a DrawDib context handle your application uses for other DrawDib functions. If Windows cannot create a DrawDib context, **DrawDibOpen** returns NULL.

Your application can use **DrawDibOpen** to create multiple DrawDib contexts. This lets your application work with several DrawDib contexts that have different characteristics.

The **DrawDibDraw** function draws a device independent bitmap to the screen. This function replaces **StretchDIBits** and it provides transparent support of installable compressors for decompressing the bitmaps. This function has the following syntax:

BOOL DrawDibDraw(*hdd*, *hdc*, *xDst*, *yDst*, *dxDst*, *dyDst*,
lpbi, *lpBits*, *xSrc*, *ySrc*, *dxSrc*, *dySrc*, *wFlags*)

The *hdd* parameter specifies a handle to a DrawDib context. The *hdc* parameter specifies a handle to the display context.

Your application uses the *lpbi* and *lpBits* parameters to specify information about the bitmap drawn. The *lpbi* parameter points to the **BITMAPINFOHEADER** structure for the bitmap and the *lpBits* parameter points to the buffer containing the bitmap.

Your application specifies the source rectangle and destination rectangle with two sets of parameters. The *xDst*, *yDst*, *dxDst*, and *dyDst* parameters specify the X and Y coordinates of the origin of the destination rectangle, and its width and height. The *xSrc*, *ySrc*, *dxSrc*, and *dySrc* parameters specify the X and Y coordinates of the origin of the source rectangle, and its width and height.

The *wFlags* parameter specifies any applicable flags for drawing. The following flags are defined:

DDF_UPDATE

Indicates the last bitmap is to be redrawn. Your application can specify NULL for *lpBits* when it uses this flag.

DDF_SAME_HDC

Uses the handle to the display context specified previously. When used, DrawDib skips preparing the display context and assumes the correct palette has already been realized (possibly by **DrawDibRealize**). Your application should not use this flag until it uses a DrawDib function that specifies the display context. Your application must still specify a handle for *hdc* when it uses this flag.

DDF_SAME_DRAW

Uses the drawing parameters previously specified for this function. Use this flag only if *lpbi*, *dxDst*, *dyDst*, *dxSrc*, and *dySrc* have not changed since last using **DrawDibDraw**.

DDF_DONTDRAW

Indicates the frame is only to be decompressed and not drawn. Your application can use the DDF_UPDATE flag to draw the image later with **DrawDibDraw**.

DDF_ANIMATE

Allows palette animation. If this flag is present, the palette DrawDib creates will have the PC_RESERVED flag set for as many entries as possible, and your application can subsequently animate the palette with **DrawDibChangePalette**. The **DrawDibDraw** function passes this flag to **DrawDibBegin** implicitly.

When your application has finished using the DrawDib context it uses **DrawDibClose** to close the context and clean up. This function uses the handle to the DrawDib context as its only argument. It returns TRUE if the context closed successfully, otherwise it returns FALSE.

Supporting Palettes for the DrawDib Functions

If your application uses DrawDib, it should be palette-aware. That is, it must respond to WM_QUERYNEWPALETTE and WM_PALETTECHANGED messages. If your application does not already use palettes, it will need two short message handlers for these messages. In response to WM_PALETTECHANGED, your application should invalidate the destination window to let DrawDib redraw. In response to WM_QUERYNEWPALETTE, your application uses the following function to respond to this message:

DrawDibRealize

This function realizes palette for drawing.

This function has the following syntax:

UINT DrawDibRealize(*hdd*, *hdc*, *fBackground*)

The *hdd* parameter specifies a handle to a DrawDib context and the *hdc* parameter specifies a handle to the display context. The *fBackground* parameter specifies whether the logical palette is always to be the background palette. If this parameter is nonzero, the selected palette is always a background palette. If this parameter is zero and the device context is attached to a window, the logical palette is a foreground palette when the window has the input focus. It returns number of entries in the logical palette that were mapped to different values in the system palette.

Manipulating Palettes

Applications use the following functions for handling palettes associated with a DrawDib context:

DrawDibSetPalette

This function sets the palette used for drawing device independent bitmaps.

DrawDibChangePalette

This function sets the palette entries used for drawing device independent bitmaps.

DrawDibGetPalette

This function obtains the palette used by a DrawDib context.

If your application is already palette-aware, it might already have realized a palette and it needs to prevent DrawDib from realizing its own palette. Your application can use **DrawDibSetPalette** to notify DrawDib of the palette it would like to use. If your substitute palette does not contain the colors required by images displayed other applications, color shifts will appear in those images. This function has the following syntax:

BOOL DrawDibSetPalette(*hdd*, *hpal***)**

The *hdd* parameter specifies a handle to a DrawDib context. The *hpal* parameter specifies a handle to the palette your application wants to use.

If your application wants to modify the colors in the DrawDib palette, it can use **DrawDibChangePalette**. This function has the following syntax:

BOOL DrawDibChangePalette(*hdd*, *iStart*, *iLen*, *lppe*)

The *hdd* parameter specifies a handle to a DrawDib context.

The *iStart* parameter specifies the starting palette entry number and the *iLen* specifies the number of palette entries to change. The *lppe* parameter specifies a pointer to an array of palette entries.

If the DDF_ANIMATE flag was not set in the previous call to **DrawDibBegin**, this function will not animate the palette. In this case, use **DrawDibRealize** to realize the updated palette.

If your application needs the current palette, it can use **DrawDibGetPalette** to obtain a handle to the palette. If you want to realize the correct palette in response to a window message, just use **DrawDibRealize** instead of using this function to obtain the pallette and resending it. If your application uses this function, it should remember that it does not have exclusive use of the handle. Thus, your application should not free the palette when it is done and it should anticipate that some other application can invalidate the handle.

Your application should rarely need to use **DrawDibGetPalette**.

Optimizing DrawDibDraw

If your application uses **DrawDibDraw** to display a series of bitmaps with the same dimensions and formats, it can use the following function to improve the efficiency of **DrawDibDraw**.

DrawDibBegin

This function prepares **DrawDibDraw** for drawing.

If your application does not use **DrawDibBegin**, **DrawDibDraw** implicitly executes it prior to drawing. When your application uses **DrawDibBegin** prior to **DrawDibDraw**, **DrawDibDraw** does not have to take the time to process the function and wait for it to complete. The **DrawDibBegin** function has the following syntax:

BOOL DrawDibBegin(*hdd*, *hdc*, *dxDest*, *dyDest*, *lpbi*, *dxSrc*, *dySrc*, *wFlags*)

The *hdd* parameter specifies a handle to a DrawDib context and the *hdc* parameter specifies a handle for the display context. Your application should use the values specified for these parameters in **DrawDibDraw**. When your application subsequently uses **DrawDibDraw**, it should set the DDF_SAME_HDC flag to indicate the handle to the display context has not changed.

The *dxDest*, *dyDest*, *dxSrc*, and *dySrc* parameters specify the width and height of the destination rectangle. The *lpbi* parameter points to the BITMAPINFOHEADER structure indicating the format of the image. Your application needs to use this same information when it specifies the parameters for **DrawDibDraw**. If your application changes these values, **DrawDibDraw** will sense the changes and implicitly use **DrawDibBegin** again. While your application must specify the width and height of the source and destination rectangles, it does not need to specify the origins. Thus, your application can redefine the origins in **DrawDibDraw** to use different portions of the image or update different portions of the display.

The *wFlags* parameter specifies applicable flags for the operation. Your application can use the DDF_ANIMATE if it will animate the palette.

Profiling the Display Characteristics

The very first time an application uses DrawDib capabilities, Windows displays a clock face and executes a series of tests to determine the display characteristics. Once the DrawDib libraries characterize the display, they will not run the tests again unless the information is removed (for example, by reinstalling Windows) or the user changes the display driver.

DrawDib Application Reference

This section is an alphabetic reference to the functions provided by Windows for applications using the DrawDib functions. The DRAWDIB.H file defines the functions, constants, and flags used by the DrawDib functions.

DrawDib Function Reference

Applications use the following functions for basic DrawDib operation:

DrawDibOpen

This function opens a DrawDib context for drawing.

DrawDibDraw

This function draws a device independent bitmap to the screen.

DrawDibClose

This function closes a DrawDib context and cleans up.

Applications use the following functions to improve the efficiency of **DrawDibDraw** when they draw a series of images:

DrawDibBegin

This function prepares **DrawDibDraw** for drawing.

DrawDibEnd

This function frees the resources allocated by **DrawDibBegin**.

Applications use the following functions for handling palettes associated with a DrawDib context:

DrawDibChangePalette

This function sets the palette entries used for drawing device independent bitmaps.

DrawDibGetPalette

This function obtains the palette used by a DrawDib context.

DrawDibRealize

This function realizes palette for drawing.

DrawDibSetPalette

This function sets the palette used for drawing device independent bitmaps.

DrawDib Functions

This section contains an alphabetical list of the functions applications can use for accessing the DrawDib services. The functions are identified with the prefix DrawDib.

DrawDibBegin

Syntax

BOOL DrawDibBegin(*hdd*, *hdc*, *dxDest*, *dyDest*, *lpbi*, *dxSrc*, *dySrc*, *wFlags*)

This function prepares **DrawDibDraw** for drawing.

Parameters

HDRAWDIB *hdd*

Specifies a handle to a DrawDib context.

HDC *hdc*

Specifies a handle for the display context.

int *dxDest*

Specifies the width of the destination rectangle.

int *dyDest*

Specifies the height of the destination rectangle.

LPBITMAPINFOHEADER *lpbi*

Specifies a pointer to a **BITMAPINFOHEADER** structure containing the format of the data to be drawn.

int *dxSrc*

Specifies the width of the source rectangle.

int *dySrc*

Specifies the height of the source rectangle.

UNIT *wFlags*

Specifies applicable flags for the operation. The following flags are defined:

DDF_ANIMATE

Allows palette animation.

Return Value

Returns a handle to a DrawDib context if successful, otherwise it returns NULL.

Comments This function prepares to draw a bitmap specified by *lpbi* to the display context *hdc* with stretching to the size *dxDest* and *dyDest*. If *dxDest* or *dyDest* is set to -1, the bitmap is drawn to a 1:1 scale with no stretching.

If *dxSrc* or *dySrc* is set to -1, **DrawDibDraw** uses the whole width or height of the source bitmap.

Use this function only if you want to prepare for drawing an image before you actually have data to draw. If you do not use this function, **DrawDibDraw** implicitly uses it when it draws the image.

See Also DrawDibEnd, DrawDibDraw

DrawDibChangePalette

Syntax **BOOL DrawDibChangePalette**(*hdd*, *iStart*, *iLen*, *lppe*)

This function sets the palette entries used for drawing device independent bitmaps.

Parameters HDRAWDIB *hdd*

Specifies a handle to a DrawDib context.

int *iStart*

Specifies the starting palette entry number.

int *iLen*

Specifies the number of palette entries to change.

LPPALETTEENTRY *lppe*

Specifies a pointer to an array of palette entries.

Return Value Returns TRUE if successful, otherwise it returns FALSE.

Comments If the DDF_ANIMATE flag was not set in the previous call to **DrawDibBegin** or **DrawDibDraw**, this function will not animate the palette. In this case, use **DrawDibRealize** to realize the updated palette.

See Also DrawDibSetPalette, DrawDibGetPalette

DrawDibClose

Syntax **BOOL DrawDibClose**(*hdd*)

This function closes a DrawDib context and cleans up.

Parameters HDRAWDIB *hdd*

Specifies a handle to a DrawDib context.

Return Value Returns TRUE if the context closed successfully, otherwise it returns FALSE.

Comments Use this function to free the **HDRAWDIB** handle and clean up after you have finished drawing.

See Also DrawDibOpen

DrawDibDraw

Syntax **BOOL DrawDibDraw**(*hdd*, *hdc*, *xDst*, *yDst*, *dxDst*, *dyDst*, *lpbi*, *lpBits*, *xSrc*, *ySrc*, *dxSrc*, *dySrc*, *wFlags*)

This function draws a device independent bitmap to the screen.

Parameters

HDRAWDIB *hdd*

Specifies a handle to a DrawDib context.

HDC *hdc*

Specifies a handle to the display context.

int *xDst*

Specifies the x-coordinate of the origin of the destination rectangle.

int *yDst*

Specifies the y-coordinate of the origin of the destination rectangle.

int *dxDst*

Specifies the width of the destination rectangle.

int *dyDst*

Specifies the height of the destination rectangle.

LPBITMAPINFOHEADER *lpbi*

Specifies a pointer to the **BITMAPINFOHEADER** structure for the bitmap.

LPVOID *lpBits*

Specifies a pointer to the buffer containing the bitmap bits. A null value indicates a packed-DIB. The data bits for a packed-DIB follow the color table in the **BITMAPINFOHEADER** structure pointed to by *lpbi*.

int *xSrc*

Specifies the x-coordinate of the source rectangle.

int *ySrc*

Specifies the y-coordinate of the source rectangle.

int *dxSrc*

Specifies the width of the source rectangle.

int *dySrc*

Specifies the height of the source rectangle.

UINT *wFlags*

Specifies any applicable flags for drawing. The following flags are defined:

DDF_UPDATE
Indicates the last bitmap is to be redrawn. If this flag is used, *lpBits* can be NULL.

DDF_SAME_HDC
Uses the handle to the display context specified previously. When used, DrawDib skips preparing the display context and assumes the correct palette has already been realized (possibly by **DrawDibRealize**). Your application should not use this flag until it uses a DrawDib function that specifies the display context. Your application must still specify a handle for *hdc* when it uses this flag.

DDF_SAME_DRAW
Uses the drawing parameters previously specified for this function. Use this flag only if the data specified for the *lpbi* structure, and the *dxDst*, *dyDst*, *dxSrc*, and *dySrc* parameters has not changed since last using **DrawDibDraw**.

DDF_DONTDRAW
Indicates the frame is only to be decompressed and not drawn. The *DDF_UPDATE* flag can be used later to actually draw the image.

DDF_ANIMATE
Allows palette animation. If this flag is present, the palette DrawDib creates will have the PC_RESERVED flag set for as many entries as possible, and the palette can be animated by with **DrawDibChangePalette**. The **DrawDibDraw** function passes this flag to **DrawDibBegin** implicitly.

Return Value

Returns TRUE if successful, FALSE otherwise.

Comments

This function replaces **StretchDIBits** and allows decompression of bitmaps by installable compressors. The function will dither true color bitmaps properly on 8-bit display devices.

DrawDibEnd

Syntax **BOOL DrawDibEnd**(*hdd*)

This function frees the resources allocated by **DrawDibBegin**.

Parameters HDRAWDIB *hdd*

Specifies the handle to the DrawDib context to free.

Return Value Returns TRUE if successful, otherwise it returns FALSE.

Comments Applications do not need to use this function.

DrawDibGetPalette

Syntax **HPALETTE DrawDibGetPalette**(*hdd*)

This function obtains the palette used by a DrawDib context.

Parameters HDRAWDIB *hdd*

Specifies a handle to a DrawDib context.

Return Value Returns a handle for the palette if successful, otherwise it returns NULL.

Comments If you want to realize the correct palette in response to a window message, use **DrawDibRealize** instead of this function. You should rarely need to call this function.

See Also DrawDibSetPalette, DrawDibRealize

DrawDibOpen

Syntax **HDRAWDIB DrawDibOpen**()

This function opens a DrawDib context for drawing.

Return Value Returns a handle to a DrawDib context if successful, otherwise it returns NULL.

Comments Call this function to obtain a handle to a DrawDib context before drawing DIBs.

See Also DrawDibClose

DrawDibRealize

Syntax **UINT DrawDibRealize**(*hdd, hdc, fBackground*)

This function realizes the palette of *hdc* into the DrawDib context specfied by *hdd*.

Parameters

HDRAWDIB *hdd*

Specifies a handle to a DrawDib context.

HDC *hdc*

Specifies a handle to the display context.

BOOL *fBackground*

Specifies whether the logical palette is always to be the background palette. If this parameter is nonzero, the selected palette is always a background palette. If this parameter is zero and the device context is attached to a window, the logical palette is a foreground palette when the window has the input focus.

Return Value

Returns number of entries in the logical palette that were mapped to different values in the system palette. If an error occurs, it returns 0.

DrawDibSetPalette

Syntax

BOOL DrawDibSetPalette(*hdd*, *hpal*)

This function sets the palette used for drawing device independent bitmaps.

Parameters

HDRAWDIB *hdd*

Specifies a handle to a DrawDib context.

HPALETTE *hpal*

Specifies a handle to the palette.

Return Value

Returns TRUE if successful, otherwise it returns FALSE.

Comments

Use this function when the application needs to realize an alternate palette. The function forces the DrawDib context to use the specified palette, possibly at the expense of image quality.

See Also

DrawDibGetPalette