

ImgLib Programmer's Reference

Welcome to ImgLib. ImgLib allows you to incorporate support for many of the most popular raster image formats without having to write any custom image reading code. ImgLib also works with **WinG**, if available, to speed up some of the operations associated with handling and displaying of raster images. The following is a list of help topics available:

Overview

Library Requirements

ImgLib API Reference

Sample Programs

Credits and Copyright Information

Beta Version Notice

What's Next?

Shareware Notice

Ordering Information

ImgLib Copyright (c) 1994 SimSoft.

Xli Copyright (c) 1989, 1990, 1991, 1992, 1993 Jim Frost, Graeme Gill and others.

Windows, Windows 95, and Windows NT are registered trademarks of Microsoft Corporation.

ImgLib Requirements

ImgLib is a Win32 DLL that has been tested and can be used in Windows NT, Windows 95, and Win32s. As these systems are all different, ImgLib has different requirements running under each version of Windows.

The preferred way to develop applications with **ImgLib** is to use a 32-bit compiler capable of creating Win32 PE-executable files. ImgLib DLL is in itself a Win32 PE-executable which means that to use it in 16-bit Windows 3.1, Win32s has to be present. Using a 32-bit compiler will assure the best performance on any supported platform.

Calling **ImgLib** from a 16-bit application using Universal Thunk (Win32s) or Generic Thunk (Windows NT or Windows95) is also supported. 16-bit applications should link with the 16-bit import library **ImgLib16.lib** that implements the thunking. Under Win32s 2 other 32-bit files are required: **imglib32.dll** and **ilstub32.exe**. These 2 executables are not required in Windows NT or Windows 95 as the 16-bit **imglib16.dll** is capable of calling the actual 32-bit routines in **ImgLib.dll** directly. In either case, the developer using **ImgLib** never needs to be concerned about writing the thunking code.

WinG, although needed by some utility functions, is optional, but recommended.

ImgLib API

The following is a list of all currently available functions callable in ImgLib.

BrightenDIB
CreateDIBPalette
DIBFree
DDBToDIB
DIBToDDB
GetLastImgLibError
GrayDIB
HalftoneDIB
ReadFileIntoDIB
ReadFileIntoDDB
ReduceDIB
SmoothDIB
WinGHalftoneDIB
WriteDIBToFile

void DIBFree (pDIB)

LPVOID pDIB a pointer to a Device-Independent bitmap

The DIBFree function frees memory that was previously allocated to hold a device-independent bitmap.

Returns:

This function does not return a value

Notes:

The **pDIB** parameter must have been previously returned by one of the ImgLib's functions like **ReadFileIntoDIB** or **WinGHalfToneDIB**. Currently, device-independent bitmaps are allocated using **GlobalAlloc**, but the internal implementation of the functions that allocate memory for device-independent bitmaps may change in the future, so freeing it by doing **GlobalFree(GlobalHandle(pDIB))** may not work.

See Also:

ReadFileIntoDIB, **WinGHalfToneDIB**

LPVOID ReadFileIntoDIB (pszFileName)

LPSTR pszFileName a file name

The **ReadFileIntoDIB** function loads one of the supported file formats into memory as a device-independent bitmap. The bitmap is ready then to be displayed using one of the **GDI** or **WinG** functions.

Returns:

If successful, the return value is a pointer to a device-independent bitmap. If unsuccessful, the function returns NULL. To get a specific reason for the failure, call **GetLastImgLibError**.

Notes:

As a convention, device-independent bitmaps are stored in memory as a complete **BITMAPINFO** structure followed immediately by the bitmap data. This is the format expected by **WinG** or **DIB.DRV**. Before displaying a device-independent bitmap on a palette-based display a GDI palette has to be created and selected into the display context. This can be accomplished easily with a call to **CreateDIBPalette**.

See Also:

ReadFileIntoDDB, **CreateDIBPalette**

HBITMAP ReadFileIntoDDB (pszFName, pPal, bAnimPal)

LPSTR pszFName	file name
HPALETTE *pPal	pointer to store a created palette
BOOL bAnimPal	will the palette be used in palette animation?

The **ReadFileIntoDDB** function loads one of the supported file formats into memory as a device-dependent bitmap. The bitmap is ready then to be displayed by using one of the GDI functions.

Returns:

If successful, the return value is a handle of a device-dependent bitmap. If applicable, the location pointed to by **pPalette** will be filled with a palette handle corresponding to the bitmap. If unsuccessful, the function returns NULL. To get a specific reason for the failure, call [GetLastImgLibError](#).

Notes:

To get the size of the bitmap returned by this function, GDI's **GetObject** function should be used. Once not needed, the bitmap can be freed by calling **DeleteObject**. When dealing with true-color image files on 256-color displays, **ReadFileIntoDDB** tries to get the best performance vs. quality results. This is done by performing a fast color quantization but no dithering. Please note that **ReadFileIntoDDB** will usually still achieve better results than [WinGHalftoneDIB](#).

See Also:

[ReadFileIntoDIB](#), [WinGHalftoneDIB](#)

int GetLastErrorLibError (void)

The GetLastErrorLibError function returns the last error number that was generated by ImgLib.

Returns:

An integer corresponding to one of the predefined **ImgLib errors**.

Notes:

Most of the ImgLib functions can generate errors. The error condition will usually be indicated by a NULL return value from one of the ImgLib API functions. If multiple errors occur between calls to **GetLastErrorLibError**, only the last error number will be returned, and the subsequent calls will return the same error number. Below is a list of possible errors returned by **GetLastErrorLibError**:

<u>ERROR_READ_ACCESS_DENIED</u>	Access to read the requested file has been denied
<u>ERROR_WRITE_ACCESS_DENIED</u>	Access to write the requested file has been denied
<u>ERROR_NO_MEMORY</u>	Out of memory processing the request
<u>ERROR_NO_DLL</u>	A DLL required for the requested operation was not found
<u>ERROR_INVALID_POINTER</u>	One of the pointer parameters or pointers used internally is not valid
<u>ERROR_INVALID_ARGUMENT</u>	One of the function arguments was not valid or out of range
<u>ERROR_UNSUPPORTED_IMAGE</u>	The image format received by ImgLib is not supported

See Also:

LPVOID WinGHalftoneDIB (pDIB, phPal)

LPVOID pDIB a 24-bit DIB pointer
HPALETTE *phPal a pointer to a palette

The **WinGHalftoneDIB** dithers a 24-bit DIB using WinG's halftone functions, if available.

Returns:

If successful, the return value is a pointer to a device-independent bitmap. If unsuccessful, the function returns NULL. To get a specific reason for the failure, call **GetLastImgLibError**.

Notes:

WinG must be present to invoke this call. If it is not installed on the system, the function will do nothing and it will return NULL. The returned device-independent bitmap must be freed by calling **DIBFree**.

See Also:

ReadFileIntoDIB, **WinG**, **DIBFree**

WinG

WinG (pronounced "Win Gee") is an optimized library designed to enable high-performance graphics techniques under Windows 3.x, Win32s, Windows NT 3.5, Windows 95, and future Windows releases. WinG allows the programmer to create a GDI-compatible HBITMAP with a Device Independent Bitmap (DIB) as the drawing surface. Programmers can use GDI or their own code to draw onto this bitmap, then use WinG to transfer it quickly to the screen. WinG also provides halftoning APIs that use the standard Microsoft halftone palette to support simulation of true color on palette devices.

All device independent bitmaps returned by ImgLib's functions are 100% compatible with all of WinG's functions.

hPal CreateDIBPalette (pDIB)

LPVOID pDIB a device-independent bitmap pointer

The **CreateDIBPalette** function creates a palette suitable for displaying a given device-independent bitmap.

Returns:

If successful, the return value is a handle to a GDI palette. If unsuccessful, the function returns NULL. To get a specific reason for the failure, call **GetLastImgLibError**.

Notes:

As a convention, device-independent bitmaps are stored in memory as a complete **BITMAPINFO** structure followed immediately by the bitmap data. **CreateDIBPalette** relies on that fact and interprets the colors within the **BITMAPINFO** structure.

See Also:

ReadFileIntoDIB

HBITMAP DIBToDDB (pDIB, phPal)

LPVOID pDIB a device-independent bitmap
HPALETTE phPal a location to store a palette at

The **DIBToDDB** function converts from a device-independent to device-dependent bitmap format.

Returns:

If successful, the return value is a handle of a device-dependent bitmap. If applicable, a palette suitable for displaying the bitmap is stored at the location pointed to by **phPal**. If unsuccessful, the function returns NULL. To get a specific reason for the failure, call **GetLastImgLibError**.

Notes:

Device-independent bitmaps are not as easily manipulated as the device-dependent bitmaps when it comes to displaying, stretching, or cropping the images. **WinG** tries to address some of the performance problems associated with device-independent bitmaps, but still many developers may be more comfortable with device-dependent bitmaps.

See Also:

ReadFileIntoDIB, **CreateDIBPalette**, **WinG**, **DIBToDDB**

LPVOID DDBToDIB (hBMP, hPal)

HBITMAP hBMP a device-dependent bitmap
HPALETTE hPal a GDI palette

The **DDBToDIB** function converts from a device-dependent to device-independent bitmap format.

Returns:

If successful, the return value is a pointer to a device-independent bitmap. If unsuccessful, the function returns NULL. To get a specific reason for the failure, call [GetLastImgLibError](#).

Notes:

The palette handle **hPal** is only required for palette-based bitmaps. Device-dependent bitmaps are not suitable for saving to files and for use with [WinG](#). For these and other reasons it may be necessary to convert to device-independent format. Also, [WinG](#) allows direct modification of device-independent bitmap data which in certain cases can improve the performance of bitmap rendering by bypassing GDI.

See Also:

[ReadFileIntoDIB](#), [CreateDIBPalette](#), [WinG](#), [DIBToDDB](#)

LPVOID BrightenDIB (pDIB, iPercent)

LPVOID pDIB	a device-independent bitmap
int iPercent	brightness percentage

The **BrightenDIB** function brightens or darkens a device-independent bitmap.

Returns:

If successful, the return value is a pointer to a new device-independent bitmap. If unsuccessful, the function returns NULL. To get a specific reason for the failure, call [GetLastImgLibError](#).

Notes:

The **iPercent** parameter specifies the brightness that the resulting bitmap will be as the percentage of the original bitmap. For example, using 110 as iPercent will increase the brightness of the resulting bitmap by 10 percent, and using 90 will darken the returned bitmap by 10 percent. Please note that palette-based device-independent bitmaps can be processed very quickly because only the color table needs to be modified, e.g. a 256-color bitmap will require 256 modifications regardless of the image size. True color bitmaps, on the other hand require that each pixel value be modified, so an 800x600 pixel image will require 400000 color modifications. Also, brightening the image, and then immediately darkening it by the same percentage will not always result in an image that's identical to the original. Calculation roundoffs as well as the maximum and minimum brightness levels are the cause of that behavior. For example, since the maximum color brightness can be 255, brightening white (255,255,255) by 10% will result in the same color, but darkening it by the same amount will result in (229,229,229).

See Also:

[ReadFileIntoDIB](#)

LPVOID ReduceDIB (pDIB, IColors, bDither)

LPVOID pDIB	a device-independent bitmap
long IColors	maximum of colors in the resulting bitmap
BOOL bDither	dither the image using the Floyd-Steinberg dithering

The **ReduceDIB** function performs color reduction on a device-independent bitmap.

Returns:

If successful, the return value is a pointer to a device-independent bitmap. If unsuccessful, the function returns NULL. To get a specific reason for the failure, call [GetLastImgLibError](#).

Notes:

The **IColors** parameter specifies the maximum number of colors that the returned bitmap will contain. Note that the returned bitmap may contain fewer colors than **IColors**. This occurs most often on gray-scale and palette-based images and is a result of the color reduction algorithm creating duplicate color entries which then are eliminated. Color reduction of true color images has fewer chances in causing this phenomenon because of the vast number of unique colors in typical true color images.

See Also:

[ReadFileIntoDIB](#), [DIBFree](#)

ImgLib General Overview

ImgLib is a 32-bit DLL that implements raster image reading, converting, and manipulating functionalities required by today's multimedia applications. ImgLib is being offered as shareware to offer a competitive alternative to many expensive commercial packages. ImgLib's attractive pricing is also possible because a lot of its code has been adopted from freely available X windows sources. For full credits see the [Credits](#) section elsewhere in this document. Besides the ability to read the most popular raster image formats ImgLib offers added functionality to modify the bitmaps in memory to aid in displaying them on all existing Windows-capable displays with best possible results and without sacrificing the image quality. The following is a list of image formats read by ImgLib:

FBM
Sun Raster
CMU WM Raster
Portable Bit Map (PBM, PGM, PPM)
Faces Project
GIF
JFIF style (standard) jpeg
Utah RLE
Windows, OS/2 BMP
Photo CD
X Window Dump
Targa
McIDAS areafile
G3 FAX
PC Paintbrush (PCX)*
GEM Bit
MacPaint
X Pixmap
X Bitmap
TIFF **

* monochrome only

** work in progress

LPVOID SmoothDIB (pDIB, iterations)

LPVOID pDIB	a device-independent bitmap
short iterations	number of times that the smoothing should be applied

The **SmoothDIB** function performs smoothing on a device-independent bitmap.

Returns:

If successful, the return value is a pointer to another device-independent bitmap. If unsuccessful, the function returns NULL. To get a specific reason for the failure, call [GetLastImgLibError](#).

Notes:

The smoothing algorithm is quite effective at removing lossy image compression's artifacts. It may also effectively clean up images that have been dithered from true color to 256-color. Unfortunately, smoothing results in some loss of image detail, so applying it more than once is not advised unless an "out-of-focus" effect is desired.

See Also:

[ReadFileIntoDIB](#), [DIBFree](#)

LPVOID HalftoneDIB (pDIB)

LPVOID pDIB a device-independent bitmap

The **HalftoneDIB** function performs black-and-white halftoning on a device-independent bitmap.

Returns:

If successful, the return value is a pointer to a device-independent monochrome bitmap. If unsuccessful, the function returns NULL. To get a specific reason for the failure, call **GetLastError**.

Notes:

The **HalftoneDIB** function performs color-to-monochrome halftoning that makes the resulting monochrome bitmap suitable for printing on a black-and-white printer. Although Windows' **StretchDIBits** is still the recommended way of getting bitmaps printed, this function may sometimes achieve more desirable results.

See Also:

ReadFileIntoDIB, **DIBFree**

LPVOID GrayDIB (pDIB)

LPVOID pDIB a device-independent bitmap

The **GrayDIB** function converts a color device-independent bitmap to grayscale.

Returns:

If successful, the return value is a pointer to a new grayscale device-independent bitmap. If unsuccessful, the function returns NULL. To get a specific reason for the failure, call **GetLastImgLibError**.

Notes:

The **GrayDIB** function performs grayscaling based on the color intensity of each pixel. Please note that palette-based device-independent bitmaps can be processed very quickly because only the color table needs to be modified, e.g. a 256-color bitmap will require 256 modifications regardless of the image size. True color bitmaps, on the other hand require that each pixel value be modified, so an 800x600 pixel image will require 400000 color modifications.

See Also:

ReadFileIntoDIB, **DIBFree**

Credits

Portions of the ImgLib code are based on XLI, an X Windows-based image viewing utility. XLI is copyrighted material with a very loose copyright allowing unlimited modification and distribution if the copyright notices are left intact. Various portions are copyrighted by various people, but all use a modification of the MIT copyright notice. The intent is to keep the source free, not to stifle its distribution, so please write to **SimSoft** or **Graeme Gill** if you have any questions. The original sources for XLI are available at: <ftp://ftp.x.org/contrib/applications/xli.1.16.tar.gz>

This helpfile was developed using Help Pre-Compiler v3.0 by Antonio Cordero Balcazar. The newest version of Help Pre-Compiler is available at <ftp://ftp.cica.indiana.edu/pub/pc/win3/utl>. The author can be reached at L0063@albeniz.eui.upm.es. Help Pre-Compiler is freeware.

Here are the names of the main contributors to XLI and their copyright statements:

Copyright (c) 1989, 1990, 1991, 1992, 1993 Jim Frost, Graeme Gill and others

Copyright (c) 1991 Tim Northrup

Copyright (c) 1990 Mark Majhor

Copyright (c) 1989, 1990 Kirk L. Johnson

All ImgLib original code, this helpfile, and the sample programs are distributed under the following Copyright:

Copyright (c) 1994, SimSoft

Here is the XLI permission statement. It also applies to the **sample code only** distributed with Imglib:

Permission to use, copy, modify, distribute, and sell this software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation. The author makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

Only the registered users of ImgLib have the right to redistribute the following files free of any royalties:

- ImgLib.dll
- imglib16.dll
- imglib32.dll
- ilstub32.exe

The following applies to all of ImgLib package:

THE AUTHOR DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

SimSoft

SimSoft specializes in Windows and Win32-based software development. Custom programming and consulting is available. For more information contact us at:

SimSoft
P.O. Box 4091
Redondo Beach, CA 90278
USA

or by e-mail at:

arybicki@netcom.com

Graeme Gill is the maintainer of Xli. He can be reached via email as: graeme@labtam.oz.au

Sample Programs

Currently, there is only one sample program, but it uses most of the ImgLib's functionality. The main file is `tstapp.c` and it can be compiled as either 16 or 32-bit program. The corresponding `tstapp.h` and `tstapp.rc` can also be compiled under either 16 or 32-bit environment. The module definition files, however, are different and the 16-bit program should use `tstapp16.def` and a 32-bit program should use `tstapp32.def`.

16-bit programs should link with `imglib16.lib` which will bind them to `imglib16.dll` which in turn will communicate with the main `imglib.dll`. 32-bit programs can link directly with `imglib.lib` which will bind them to the main `imglib.dll`.

Initially, a C++ program was developed to serve as a test application, but it was dependent on MFC 3.0 which is not available to everyone. For that reason, a plain C program (`tstapp.c`) was developed sacrificing some of the nice user-interface features in favor of portability and readability. This way one file (`tstapp.c`) contains all the sample code in contrast to many files generated by the Visual C++ environment.

Please remember that if you distribute your programs as 32-bit applications you only need to include `imglib.dll` in addition to your program files. When distributing a 16-bit program, however, you will need to include `imglib.dll`, `imglib16.dll`, `imglib32.dll`, and `ilstub32.exe`. The last 2 of those are required for Win32s support and would not be necessary for Windows 95 or NT.

Beta Version Notice

This beta version of **ImgLib** may not be distributed with your programs. You will need a registered version (when released) to be able to do that. The most current beta version will always be available at ftp://ftp.netcom.com/pub/simsoft/ImgLib_beta.exe.

Please report all bugs by using the included **bug report form**. To save the form, just copy it to the clipboard from this helpfile, and then paste it to your editor and save. Try to include as much information as you can. If you find a picture file that should be but is not read properly by ImgLib, we can arrange for you to be able to forward it to me, but I will not make my anonymous directory write-enabled unless it is for a short period of time and I am logged in at the same time.

You may discuss any and all features of ImgLib anywhere--SimSoft is not Microsoft :-). Unless someone's life depends on it, I will not remove any features from **ImgLib**, so you may depend on their presence in the release version. For the list of features that I am planning to add in the release version or in future versions, see the **What's Next?** page.

What's Next

Features planned for the release version:

Various performance improvements
Merge two DIB's into one.
Expand a palette-based DIB into true color.
Rotate a DIB by 90, 180, or 270 degrees.
Flip a DIB horizontally, or vertically.
Zoom a DIB

Features planned for future releases:

Color PCX file support.
TIFF file support
Writing DIBs in more than .BMP file format.

ImgLib Bug Report
=====

Is the problem repeatable?

Always ☐ Sometimes ☐ Happened only once ☐

Which most closely describes the problem?

Access violation ☐ Nothing returned ☐ Corrupt image ☐
System hung ☐ Corrupt file ☐ Documentation error ☐

Suspected area of problem:

ImgLib ☐ Universal Thunk code ☐ Generic Thunk code ☐

Is ImgLib useable after this problem?

Yes ☐ No ☐

Is there a particular image file that exposes this problem?

Yes ☐ No ☐

If yes, describe how I can ftp it for testing.

Describe the problem and steps to take to repeat the problem here:

Which version of Windows do you develop under?

Win31 ☐ Windows 95 ☐ Windows NT ☐

What type of programs do you use ImgLib with?

16-bit ☐ 32-bit ☐

Describe the compiler/development environment details here:

How much RAM do you have?

Enter any suggestions for improving ImgLib here:

BOOL WriteDIBToFile (pDIB, pszFileName)

LPVOID pDIB a device-independent bitmap

LPSTR pszFileName file name

The **WriteDIBToFile** function saves a given device-independent bitmap in a Windows bitmap (.BMP) file.

Returns:

If successful, the return value is TRUE. If unsuccessful, the function returns FALSE. To get a specific reason for the failure, call **GetLastImgLibError**.

Notes:

All device-independent bitmaps are supported ranging from 1-bit to 24-bit.

See Also:

ReadFileIntoDIB, **DIBFree**

