IMAGE HANDLER - LEADTOOLS Demonstration Programs

These programs demonstrate many of the features of the LEADTOOLS 6.1 imaging toolkits. The toolkits include 16- and 32-bit ActiveXs (OCXs), a 16-bit VBX, and 16- and 32-bit DLLs, which are available in various bundled combinations described in LEADTOOLS Products.

These programs are taken directly from the example programs for the LEADTOOLS API. The various toolkits include these and many other examples, with source code. For a summary, refer to Example Programs.

Some of the features in these demonstration programs, such as annotations and region processing, are available only in the Express editions. Also, some significant features, such as data binding, are not demonstrated in these programs. To review all of the available features in various toolkits, refer to Summary of LEADTOOLS Features.

- The following demonstration programs are for WIN32 platforms. They use the LEADTOOLS WIN32 DLLs, and therefore, will only run under Win32s, NT, or Win95.

  WIN32 Demonstration shows most of the features of the LEADTOOLS API in a 32-bit environment. Demonstrated features include file format support, image processing, region processing, and display options.

  WIN32 Annotations demonstrates the high-level annotation features of the LEADTOOLS API in a 32-bit environment. Most of the demonstrated features are automated and require very little programming. Refer to User Interface for Annotations.

- The following demonstration programs are for WIN16 platforms. They use the LEADTOOLS WIN16 DLLs, and therefore, will run under Windows 3.x.

  WIN16 Demonstration shows most of the features of the LEADTOOLS API in a 16-bit environment. Demonstrated features include file format support, image processing, region processing, and display options.

  WIN16 Annotations demonstrates the high-level annotation features of the LEADTOOLS API in a 16-bit environment. Most of the demonstrated features are automated and require very little programming. User Interface for Annotations.

All of the demonstration programs were compiled with the *LEADTOOLS Pro Express* product. To achieve the same performance in your code, you should purchase *LEADTOOLS Pro Express.*

Other Products from LEAD Technologies

LEAD Technologies, Inc.
900 Baxter Street
Charlotte, NC 28204

Sales@Leadtools.com
www.leadtools.com

(704) 332-5532
FAX: (704) 372-8161
BBS: (704) 334-9045
Tech Support: (704) 372-9681

CompuServe: 71333,2237 or GO LEADTECH

LEADVIEW 3 for Windows: An end-user product for handling images. LEADVIEW is a complete imaging file management system. Built-in features include: photo album with extensive search capabilities, image processing, draw, special paint effects, image enhancement, slide show, screen capture, file conversion, archiving, printing, and OLE and DDE support. LEADVIEW was developed using LEADTOOLS Pro Express. It supports all the formats listed in Summary of All Supported Image File Formats. (GIF and TIFF LZW are read-only.) LEADVIEW recently was recognized by Windows Magazine Top 100, 1995 and PC Graphics and Video Highly recommended awards.

LEADTOOLS DATA Compression Toolkit: A toolkit for creating installation disks and archives. This toolkit can save up to 50 percent in disk space over other compression methods used with MSSETUP, InstallShield, or proprietary installation packages. It is an easy-to-integrate16/32 bit compression/decompression system with an in-line encoder/decoder API, file encoder/decoder API, an archive encoder/decoder API, an MSSETUP enhancer, and an InstallShield enhancer. It ships with full on-line documentation, API examples in C, an MSSETUP installation example, and an InstallShield installation example.

Call for details on custom applications, operating system ports and licensing fees.

All products are sold in the U.S. and Canada with a 30-day money back guarantee.

About LEAD Technologies, Inc.

LEAD Technologies is an industry leader in the field of compression, image compression and image processing technology, licensing technology to thousands of developers, software publishers and OEMs, including Corel Corporation for use in CorelDRAW products and Inset Systems for use in Hijaak. Other customers include AT&T, Canon, Delrina, Intel, Intuit, Kodak, Martin Marietta, MicroGraFix, Microsoft, Olympus, Prodigy, Reuters International, Rockwell, Sharp Electronics, 3M, Upjohn and Xerox Corporation.

LEAD's products include LEADVIEW for Windows, an image management tool aimed at end-users, LEADTOOLS, a family of imaging software development toolkits and LEADTOOLS Data, a data compression technology aimed at OEMs. Founded in 1990, LEAD is headquartered in Charlotte, North Carolina.

General Information

Technical Support: As a registered user, free technical support is available. You must register your product to receive technical support. Technical support is available 9:00 AM through 6:00 PM Eastern Standard Time.

MONEY BACK GUARANTEE ON ALL PRODUCTS: In the U.S. and Canada LEAD offers an unconditional 30 day money-back guarantee on all products. Try our toolkit - risk free - and we guarantee that you will select LEAD as your partner in development.

For more Information, please contact:

Sales.@Leadtools.com
Home page: http://www.leadtools.com

LEAD Technologies, Inc.
900 Baxter Street
Charlotte, NC 28204

Main: (704) 332-5532
Sales: (800)-637-4699

Technical support: (704) 372-9681
Fax: (704)372-8161
BBS: (704) 334-9045
CompuServe: 71333,2237 or GO LEADTECH

Example Programs

These are descriptions of the example programs that ship with the LEADTOOLS toolkits.

API Example Programs

ActiveX (OCX) Example Programs

VBX Example Programs

API Example Programs

These are descriptions of the DLL example programs that ship with the LEADTOOLS toolkits.

ANNOTATE	**Annotations** demonstrates the annotation capabilities of the express toolkit.
CLRSPACE	**Color Space** loads and displays the image file specified on the command line, then uses the L_ConvertColorSpace function to convert from RGB to YUV, CMYK, and other formats, redisplaying the image after all conversions are complete.
COLOR	**Color Resolution** loads the 24-bit image file specified on the command line and calls L_ColorResBitmap to optimize and dither the image to 8 bits per pixel.
COMPCB	**Compress CB** loads the image file specified on the command line, then displays the image, compresses the bitmap to file using a callback function, clears the bitmap, loads the compressed image, and displays the newly loaded image.
DELPHI	**Delphi Demo** demonstrates the use of LEADTOOLS with the Borland Delphi development system.
DEMO	**Main Demo** is the main example program. It uses pull-down menus to demonstrate most of the LEADTOOLS features in one application.
DIBDDB	**DIBDDB Demo** demonstrates the functions that translate to and from Windows device independent bitmaps (DIBs) and device dependent bitmaps (DDBs).
DRAW	**Draw Demo** uses pull-down menus to show some of the drawing capabilities available with LEADTOOLS.
EZFUNC	**Simple Load** uses simple LEADTOOLS functions to load an image file specified on the command line and display the bitmap in a window.
EZREGC	**Reg. Class Load** uses the L_BITMAPCLASS registered class to load an image file specified on the command line and display the bitmap in a window.
FEATURE1	**Some Basic Features** loads and displays the 24-bit image file specified on the command line, then makes the following changes, redisplaying the image after each change: optimize to 8 bits per pixel, flip, lighten, resize, and rotate.
FEATURE2	**More Basic Features** loads and displays the 24-bit image file specified on the command line, then makes the following changes, redisplaying the image after each change: dither to 8 bits per pixel, reverse, grayscale, resize, and fill with one color.
FEATURE3	**A Few More Features** loads and displays the image file specified on the command line, copies the image data to a different bitmap, halftones the original image, restores the image by copying the handle of the new bitmap, then redisplays the image.
FEEDLOAD	**Feed Load** demonstrates how to use L_FeedLoad and related functions by simulating the receipt of a transmitted image.
GETROW	**Get Row** loads the 24-bit image file specified on the command line, then rotates the image 90 degrees using L_GetBitmapRow and L_PutBitmapRowCol, and redisplays the image.
INFO	**Info** displays image information from the file specified on the command line.
LOADCB	**Load CB** loads the image file specified on the command line, using a callback function to display the image as it is loaded.
LOADSAVE	**Load and Save** uses pull-down menus to let the user load an image file, display the image in a window, and save the image to a specified file name, format, and bits per pixel. The options are implemented using a Windows common dialog box.
MEMORY	**Load Memory** demonstrates the ability load a file from memory. It allocates memory, loads the image file specified on the command line into the allocated memory, then loads the image from memory to a bitmap using L_LoadBitmapMemory and displays the image.
MEMORYCB	**Load Memory CB** demonstrates the ability use a callback function when loading a file from memory. It allocates memory, loads the image file specified on the command line

	into the allocated memory, then loads the image from memory to a bitmap using L_LoadMemory and a callback function, and displays the image.
MFCDEMO	**MFC Demo** demonstrates the use of LEADTOOLS with the Microsoft Foundation Class (MFC). This example includes a C++ wrapper for the LEADTOOLS API.
OFFSET	**Save w/ Offset** demonstrates the ability to load an image file that is embedded in a larger file. It does so by using the L_LoadFileOffset function, which lets you specify the offset and the size of the embedded file. This example first loads the normal image file specified on the command line. It embeds the image in a second file, which is specified in the second command line argument. It then loads and displays the embedded file.
PAINTDC	**Paint DC** loads the image file specified on the command line, and displays the image. The user can then choose a menu option to modify the source and display rectangles, and redisplay the image to see the changes.
PAINTEFF	**Paint Effect Demo** uses pull-down menus to demonstrate the various transitional painting effects that are available with LEADTOOLS.
REDIRECT	**Redirect I/O** demonstrates how to use the L_RedirectIO function. It redirects the standard I/O functions so that they read and seek into the allocated memory rather than a file.
REGCLASS	**Registered Class Demo** demonstrates most of the capabilities of L_BITMAPCLASS registered class.
RESIZE	**Resize** loads the image file specified on the command line, resizing the image while loading it. It then places the resized image over the original and displays the combined image.
SAVECB	**Save CB** loads the image file specified on the command line, then displays the image and saves it to the current directory using a callback and status window.
SAVEMEM	**Save to Memory** demonstrates the ability to save a file in memory. It lets the user load an image file, display the image in a window, and save the image to a file in memory.
SCRLZOOM	**Scroll & Zoom Demo** uses pull-down menus to let the user load an image file, display the image in a window, and use zooming and scrolling features. In doing so, it also demonstrates the ability of the L_PaintDC function to automatically dither a 16-, 24-, or 32-bit image when displaying on a 4- or 8-bit device.
STAMPGET	**Get Stamp** reads a thumbnail (stamp) image stored in a compressed file that the user selects; then displays the image in a window. The only valid input file is a LEAD or JPEG file that is compressed with a stamp image. (Use the STAMPPUT example first to create an image file that contains a thumbnail.)
STAMPPUT	**Put Stamp** loads the image file specified on the command line, calls L_ColorResBitmap (if necessary) to obtain a 24 bit image, calls L_CompressBitmapWithStamp to compress the image to STAMP.CMP (a LEAD CMP compressed file format), then loads the stamp to a bitmap and displays the stamp.
UNDERLAY	**Underlay** demonstrates the L_UnderlayBitmap function. It loads two images specified on the command line and displays the first one. It then combines the images using the second one as an underlying image, and displays the combined image.

ActiveX (OCX) Example Programs

These are descriptions of the ActiveX (OCX) example programs that ship with the LEADTOOLS toolkits.

Visual Basic

- EXAMPLES\OCX\VB\DEMO contains a sample application that demonstrates most of the ActiveX capabilities in Visual Basic 4.0. To run the sample application, you can run the EXE file in the LEAD62\BIN directory.
- EXAMPLES\OCX\VB\ANNOTATE contains a sample application that demonstrates the annotation capabilities. To run the sample application, you can run the EXE file in the LEAD62\BIN directory.
- EXAMPLES\OCX\VB\BIND contains a sample database application implemented using data binding. For WIN32 systems, this example references the IMAGES\LEADPIC.MDB database. For WIN16 systems, it references the IMAGES\LDPIC16.MDB database. To run the sample application, you can run the EXE file in the LEAD62\BIN directory, specifying the correct database as a command-line argument.
- (32-bit editions only) EXAMPLES\OCX\VB\DB contains sample project code for an ODBC database example in Visual Basic 4.0. To use the DB demo, you must first do the following:
    1. Specify an ODBC data source named LEADACCESS that references the IMAGES\LEADPIC.MDB database.
    2. Open the project in Visual Basic and modify the Dpeople data control's DatabaseName property to reference the IMAGES\LEADPIC.MDB database.
- EXAMPLES\OCX\VB\DRAW uses the LEADTOOLS ActiveX to load an image, to demonstrate graphics functions, and to save the modified image. This program calls several Windows GDI functions using the bitmap as a drawing surface. To run the sample application, you can run the EXE file in the LEAD62\BIN directory.
- EXAMPLES\OCX\VB\MAGNIFY uses two LEAD controls. The first control displays the unmagnified image. The second control, when enabled, can be dragged across the first one to magnify the covered portion of the image. To run the sample application, you can run the EXE file in the LEAD62\BIN directory.

Visual C++

- EXAMPLES\OCX\MFC\DEMO contains sample project code for Visual C++ 1.0 and later (16- or 32-bit). To run the sample application, you can run the EXE file in the LEAD62\BIN directory.
- (16-bit editions only) EXAMPLES\OCX\MFC\ANNOTATE demonstrates the LEADTOOLS annotation capabilities. To run the sample application, you can run the EXE file in the LEAD62\BIN directory.
- (16-bit editions only) EXAMPLES\OCX\MFC\DRAW uses the LEADTOOLS ActiveX to load an image, to demonstrate graphics functions, and to save the modified image. This program calls several Windows GDI functions using the bitmap as a drawing surface. To run the sample application, you can run the EXE file in the LEAD62\BIN directory.
- (16-bit editions only) EXAMPLES\OCX\MFC\MAGNIFY uses two LEAD controls. The first control displays the unmagnified image. The second control, when enabled, can be dragged across the first one to magnify the covered portion of the image. To run the sample application, you can run the EXE file in the LEAD62\BIN directory.

Visual C++ 4.0

- (32-bit editions only) EXAMPLES\OCX\MFC40\DEMO contains sample project code for Visual C++ 4.0. You can then open the project in Visual C++; then compile and run it.
- (32-bit editions only) EXAMPLES\OCX\MFC40\ANNOTATE demonstrates the LEADTOOLS annotation capabilities. To run the sample application, you can run the EXE file in the LEAD62\BIN directory.
- (32-bit editions only) EXAMPLES\OCX\MFC40\DB contains sample project code for a database example in Visual C++ 4.0. To use the DB demo, you must first do the following:
    1. Specify an ODBC data source named LEADACCESS that references the IMAGES\LEADPIC.MDB database.
    2. Open the project in Visual C++ and modify the GetDefaultDBName function to reference the

IMAGES\LEADPIC.MDB database.

- (32-bit editions only) EXAMPLES\OCX\MFC40\DRAW uses the LEADTOOLS ActiveX to load an image, to demonstrate graphics functions, and to save the modified image. This program calls several Windows GDI functions using the bitmap as a drawing surface. To run the sample application, you can run the EXE file in the LEAD62\BIN directory.
- (32-bit editions only) EXAMPLES\OCX\MFC40\MAGNIFY uses two LEAD controls. The first control displays the unmagnified image. The second control, when enabled, can be dragged across the first one to magnify the covered portion of the image. To run the sample application, you can run the EXE file in the LEAD62\BIN directory.

Visual FoxPro

- (32-bit editions only) EXAMPLES\OCX\FOX contains sample project code for Visual FoxPro 3.0. To run the sample application from FoxPro, use the *Program* > *Do* option, and select the \EXAMPLES\OCX\FOX\DEMO.APP application.

  To implement the database capabilities in the DEMO.APP application, specify the current (\FOX) directory as an ODBC data source. Then when you run the application, use the *File* > *Open Database* option to open the ODBC data source.

Access 7.0

- (32-bit editions only) EXAMPLES\OCX\ACCESS7\DEMO contains sample project code for Microsoft Access 7.0. To run the sample application from Access, open the sample database, then open the MAIN form. To avoid ODBC setup issues, this demo does not add images to a database.
- (32-bit editions only) EXAMPLES\OCX\ACCESS7\DB contains sample project code for a database example in Access 7.0. To use the DB demo, you must do the following:

  1. Specify an ODBC data source named LEADACCESS that references the IMAGES\LEADPIC.MDB database.
  2. Open the project in Access 7.0 and modify Query1 to reference the IMAGES\LEADPIC.MDB database.
  3. Run the sample application from Access by opening the MAIN form.

Access 2.0

- (16-bit editions only) EXAMPLES\OCX\ACCESS contains sample project code for Microsoft Access 2.0. To run the sample application from Access, open the MAIN form.

Delphi 2.0

- EXAMPLES\OCX\DELPHI\DEMO contains sample project code for Delphi 2.0. To run the sample application, you can run the EXE file in the LEAD62\BIN directory.
- EXAMPLES\OCX\DELPHI\DRAW uses the LEADTOOLS ActiveX to load an image, to demonstrate graphics functions, and to save the modified image. This program calls several Windows GDI functions using the bitmap as a drawing surface. To run the sample application, you can run the EXE file in the LEAD62\BIN directory.

VBX Example Programs

These are descriptions of the VBX example programs that ship with the LEADTOOLS toolkits.

Main Demo

VBXDEMO.EXE is a demonstration program that uses the LEADTOOLS VBX for displaying and manipulating images. Most of the properties of the LEAD Control are demonstrated at some point in this program. The project that created this .EXE is **DEMO.MAK**, which is in the DEMO subdirectory.

Generally, the menu options reflect the programming capabilities, which you can find by searching in the online help. Options on the *Preferences* menu do the following:
Dithering sets the PaintDither property.

Bitonal Scaling sets the BitonalScaling property.

Paint With Fixed Palette sets the PaintPalette property.

Paint While Load sets the PaintWhileLoad property.

Show Open Options implements a dialogue box for opening a file. The dialogue box lets you select the bits per pixel to load and the page number (for multi-page files). It also lets you access file information using the InfoFile property.

Show Progress sets the EnableStatus property, and uses the status event to show completion percentages, as described in Detecting Events.

Draw Example

VBXDRAW.EXE is a demonstration program that uses the LEADTOOLS VBX to load an image, to demonstrate graphics functions, and to save the modified image. This program calls several Windows GDI functions using the bitmap as a drawing surface. The project that created this .EXE is **DRAW.MAK**, which is in the DRAW subdirectory.

Database Example

VBXDBASE.EXE is a demonstration program that uses the LEAD Visual Basic Control for displaying, adding, and updating images in an ACCESS database. The project that created this .EXE is **DATABASE.MAK**, which is in the DATABASE subdirectory.

A command line argument specifying the full name of the database is required. The command line argument can be set from the project by selecting the following sequence of options from the main VB Window after opening the project: Options, Project, Command Line Arguments.

If you are running the example .EXE, the full name of the database should follow the .EXE name on the command line. One database is provided by LEAD. It is an .MDB file in the IMAGES subdirectory of the main installation directory.

Magnify Example

VBXMAGN.EXE is a demonstration program that uses two LEAD Visual Basic Controls. The first control displays the unmagnified image. The second control, when enabled, can be dragged across the first one to magnify the covered portion of the image. The project that created this .EXE is **MAGNIFY.MAK**, which is in the MAGNIFY subdirectory.

Delphi Example

DPDEMO.EXE demonstrates the use of the LEADTOOLS VBX with Borland Delphi. The project

that created this .EXE is **DPDEMO.DPR**, which is in the DELPHI subdirectory.

License Agreement

This is a copy of the license agreement that appears when you install IMAGE HANDLER.

DEMO LICENSE AGREEMENT FOR LEADTOOLS IMAGE HANDLER

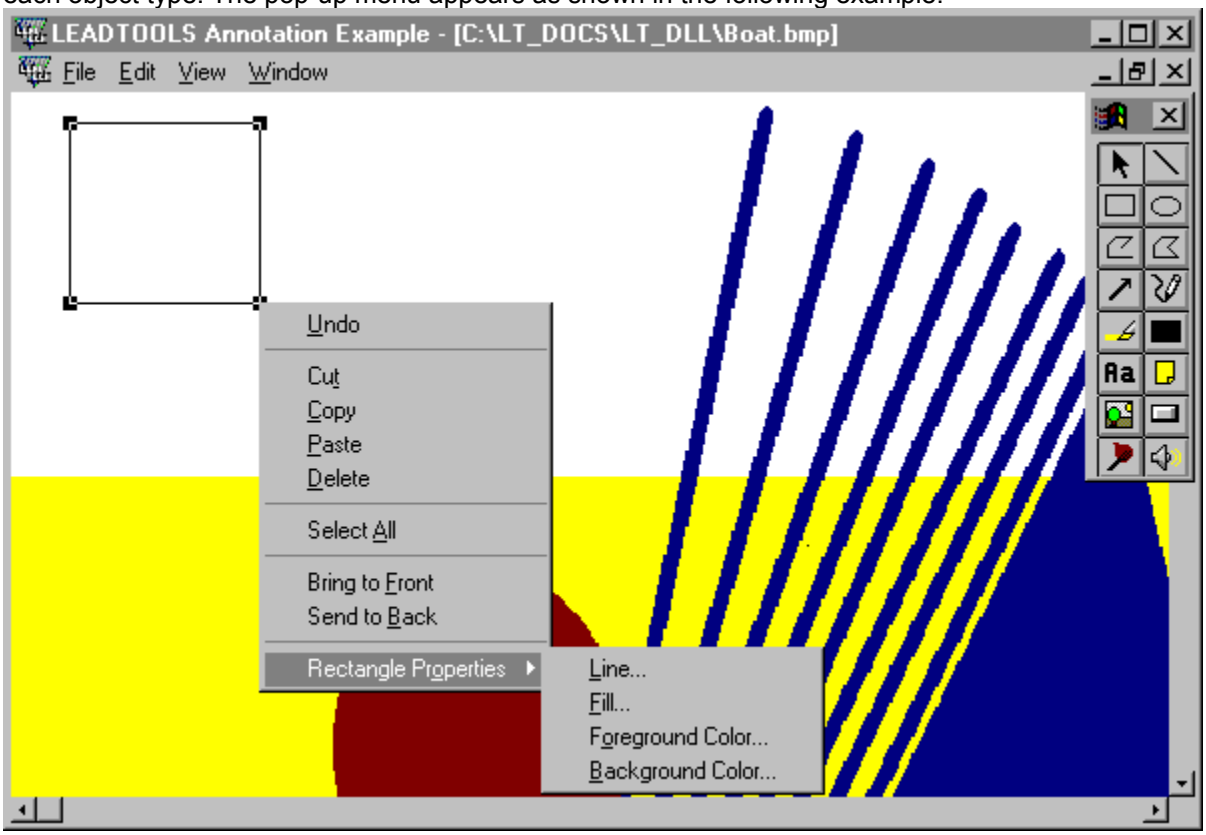
BEFORE YOU CLICK ON THE "ACCEPT" BUTTON AT THE END OF THIS DOCUMENT, CAREFULLY READ THE TERMS AND CONDITIONS OF THIS AGREEMENT. BY CLICKING ON THE "ACCEPT" BUTTON, YOU ARE CONSENTING TO BE BOUND BY AND ARE BECOMING A PARTY TO THIS AGREEMENT. IF YOU DO NOT AGREE TO ALL OF THE TERMS OF THIS AGREEMENT, CLICK THE "DO NOT ACCEPT" BUTTON AND DO NOT USE THE SOFTWARE.

1. LEAD Technologies, Inc. ("LEAD") hereby grants you a nonexclusive, non-sublicensable license to use this LEADTOOLS IMAGE HANDLER software ("Software") in binary executable form for evaluation and trial use purposes only. You may only use the Software, including, without limitation, all underlying .dlls, .exes and other files comprising the Software, for the purpose of evaluating its functionality and performance to determine whether you desire to obtain a development license to a product in the LEADTOOLS' family of developer toolkits, and for no other purpose. To obtain a development license, contact LEAD or one of our authorized resellers.
2. Except for the limited rights granted to you, LEAD and/or its suppliers retain all ownership and other proprietary rights in the Software. You may not modify, reverse engineer, decompile, disassemble or copy (except for backup purposes) the Software nor may you redistribute the Software or any portion thereof.
3. The Software is being provided to you "as is", without warranty of any kind.

User Interface for Annotations

The *Edit* > *User Mode* option lets you select design or run mode for annotations.

In design mode, you can select tools using the *Edit* > *Tool* option and draw annotation objects over the image in the window. You can then use the right mouse button to get a pop-up menu that is tailored to each object type. The pop-up menu appears as shown in the following example:



The popup menu is context sensitive. It lists only the properties that can be changed on the selected object or objects. If you use the popup menu to change a property, the value that you specify is applied to the current selection, and it becomes the default for the next object.

The following topics describe the automated annotation features for particular objects and tools:

- Setting Defaults
- Selection Pointer Tool
- Line Tool
- Rectangle Tool
- Ellipse Tool
- Polyline Tool
- Polygon Tool
- Pointer Tool
- Freehand Line Tool
- Highlight Tool
- Redaction Tool
- Text Tool
- Note Tool
- Stamp Tool
- Hot spot Tool
- Button Tool
- Audio Clip Tool

Setting Defaults

Note: This topic is for Express editions only.

When using annotations in design mode, you can change the default properties by clicking with the right mouse button on an empty spot (a place on the image without any annotations). The popup menu lets you do the following:

- Undo the last user action.
- Select all annotation objects in the container.
- Delete all selected objects.
- Change the default line width and style properties. See Line Dialog Box for Annotations.
- Change the default fill mode (transparent, translucent, or opaque) and the fill pattern. See Fill Dialog Box for Annotations.
- Change the polygonal fill mode, which controls which areas are filled when lines cross. See Fill Dialog Box for Annotations.
- Change the default foreground and background colors. See Color Dialog Box for Annotations.
- Change the font selection and font characteristics. See Font Dialog Box for Annotations.

The default background color does not affect highlight, redaction, and note objects.

Selection Pointer Tool

Note: This topic is for Express editions only.

The selection pointer is the mouse pointer for selecting and manipulating annotation objects in design mode.

With the selection pointer, you can select one or more objects as follows:

- You can select a single object by clicking on the object.
- You can click and drag to form a rectangular selection area that will select all objects that are completely contained within it.
- You can hold down the shift key and click on several objects one-at-a-time to select them all.

When an object is selected, handles appear on the corners of its bounding rectangle. You can click on the body of the object and drag it to another position. You can click on a handle and drag to resize the object. If more than one object is selected, all of them are moved or resized.

In addition, you can use this pointer to rotate an object. If you hold down the shift key, click on a handle, and drag the mouse, the object rotates. If more than one object is selected, all of them are rotated, using the center of the group as the center of rotation. You cannot rotate a note, button, or audio clip. If you rotate a group that includes one of these objects, the object will move to the new position, but will retain its orientation.

The popup menu is also affected if more than one object is selected. In that case, when you click the right mouse button, the popup menu lets you change the properties of all the selected objects, and it lets you delete all the selected objects.

Line Tool

Note: This topic is for Express editions only.

The line annotation object is a simple line. In design mode, the line is started on a mouse-down event and completed on a mouse-up event.

You can then click on the object with the right mouse button to do the following:

- Undo the last user action.
- Cut or copy the object to the clipboard.
- Delete the object.
- Bring the object to the front or send it to the back.
- Change the line width and style properties. See Line Dialog Box for Annotations.
- Change the foreground color. See Color Dialog Box for Annotations.

Rectangle Tool

Note: This topic is for Express editions only.

The rectangle annotation object is a simple rectangle. In design mode, you can click and drag to specify the rectangle in the current window.

You can then click on the object with the right mouse button to do the following:

- Undo the last user action.
- Cut or copy the object to the clipboard.
- Delete the object.
- Bring the object to the front or send it to the back.
- Change the line width and style properties. See Line Dialog Box for Annotations.
- Change the fill mode (transparent, translucent, or opaque) and the fill pattern. See Fill Dialog Box for Annotations.
- Change the foreground and background colors. See Color Dialog Box for Annotations.

Ellipse Tool

Note: This topic is for Express editions only.

The ellipse annotation object is a simple ellipse. In design mode, the you click and drag to specify the ellipse in the current window.

You can then click on the object with the right mouse button to do the following:

- Undo the last user action.
- Cut or copy the object to the clipboard.
- Delete the object.
- Bring the object to the front or send it to the back.
- Change the line width and style properties. See Line Dialog Box for Annotations.
- Change the fill mode (transparent, translucent, or opaque) and the fill pattern. See Fill Dialog Box for Annotations.
- Change the foreground and background colors. See Color Dialog Box for Annotations.

Polyline Tool

Note: This topic is for Express editions only.

The polyline annotation object is an array of points that create a sequence of joined lines. In design mode, each line segment is formed with a mouse click, and the object is completed on a double click.

You can then click on the object with the right mouse button to do the following:

- Undo the last user action.
- Cut or copy the object to the clipboard.
- Delete the object.
- Bring the object to the front or send it to the back.
- Change the line width and style properties. See Line Dialog Box for Annotations.
- Change the foreground color. See Color Dialog Box for Annotations.

Polygon Tool

Note: This topic is for Express editions only.

The polygon annotation object is an array of points that define the vertices of a polygon. In design mode, each line segment is formed with a mouse click, and the object is completed on a double click, which closes the polygon with a line from the last point to the first point.

You can then click on the object with the right mouse button to do the following:

- Undo the last user action.
- Cut or copy the object to the clipboard.
- Delete the object.
- Bring the object to the front or send it to the back.
- Change the line width and style properties. See Line Dialog Box for Annotations.
- Change the fill mode (transparent, translucent, or opaque) and the fill pattern. See Fill Dialog Box for Annotations.
- Change the polygonal fill mode, which controls which areas are filled when lines cross. See Fill Dialog Box for Annotations.
- Change the foreground and background colors. See Color Dialog Box for Annotations.

Pointer Tool

Note: This topic is for Express editions only.

The pointer annotation object is defined by two points, with the an arrow at the first point. In design mode, you can click and drag to specify the points in the current window.

You can then click on the object with the right mouse button to do the following:

- Undo the last user action.
- Cut or copy the object to the clipboard.
- Delete the object.
- Bring the object to the front or send it to the back.
- Change the line width and style properties. See Line Dialog Box for Annotations.
- Change the foreground color. See Color Dialog Box for Annotations.

Freehand Line Tool

Note: This topic is for Express editions only.

The freehand line annotation object is an array of points that create a sequence of joined lines. In design mode, each line segment is formed with a mouse move event when the left button is down, and the object is completed on a mouse up event.

You can then click on the object with the right mouse button to do the following:

- Undo the last user action.
- Cut or copy the object to the clipboard.
- Delete the object.
- Bring the object to the front or send it to the back.
- Change the line width and style properties. See Line Dialog Box for Annotations.
- Change the foreground color. See Color Dialog Box for Annotations.

Highlight Tool

Note: This topic is for Express editions only.

The highlight annotation object is a rectangle with default properties that are appropriate for highlighting an area. In design mode, you can click and drag to specify the rectangle in the current window.

You can then click on the object with the right mouse button to do the following:

- Undo the last user action.
- Cut or copy the object to the clipboard.
- Delete the object.
- Bring the object to the front or send it to the back.
- Change the background color. See Color Dialog Box for Annotations.

Redaction Tool

Note: This topic is for Express editions only.

The redaction annotation object is a rectangle with default properties that are appropriate for blacking out an area. In design mode, you can click and drag to specify the rectangle in the current window.

You can then click on the object with the right mouse button to do the following:

- Undo the last user action.
- Cut or copy the object to the clipboard.
- Delete the object.
- Bring the object to the front or send it to the back.
- Change the background color. See Color Dialog Box for Annotations.

Text Tool

Note: This topic is for Express editions only.

The text annotation object is a rectangle containing a character string with font properties. The text will wrap on word breaks within the object's specified rectangle. In design mode, you can click and drag to specify the rectangle in the current window. A dialog box then lets you enter the text.

You can then click on the object with the right mouse button to do the following:

- Undo the last user action.
- Cut or copy the object to the clipboard.
- Delete the object.
- Bring the object to the front or send it to the back.
- Change the fill mode (transparent, translucent, or opaque). See Fill Dialog Box for Annotations.
- Change the background color. See Color Dialog Box for Annotations.
- Change the font selection and font characteristics. See Font Dialog Box for Annotations.
- Edit the text. See Text Dialog Box for Annotations.

You cannot flip or reverse a text object, but you can rotate it. If you flip or reverse a container that includes a text object, the text will move to the new position, but will retain its orientation.

Note Tool

Note: This topic is for Express editions only.

The note annotation object is a rectangle with a shadow border and color background (yellow by default). It contains a character string with font properties. The text will wrap on word breaks within the object's specified rectangle. In design mode, you can click and drag to specify the rectangle in the current window. A dialog box then lets the user enter the text.

You can then click on the object with the right mouse button to do the following:

- Undo the last user action.
- Cut or copy the object to the clipboard.
- Delete the object.
- Bring the object to the front or send it to the back.
- Change the background color. See Color Dialog Box for Annotations.
- Change the font selection and font characteristics. See Font Dialog Box for Annotations.
- Edit the text .See Text Dialog Box for Annotations.

You cannot flip, reverse, or rotate a note. If you flip, reverse, or rotate a container that includes a note, the note will move to the new position, but will retain its orientation.

Stamp Tool

Note: This topic is for Express editions only.

The stamp annotation object is a bitmap image. In design mode, you can click and drag to specify a rectangle. The bitmap is scaled to fit the rectangle.

You can then click on the object with the right mouse button to do the following:

- Undo the last user action.
- Cut or copy the object to the clipboard.
- Delete the object.
- Bring the object to the front or send it to the back.
- Select an image file to load into the stamp. See Stamp Bitmap Dialog Box for Annotations.

On a device that displays 256 colors or less, you should use a fixed palette when displaying bitmaps. Otherwise, a palette shift will occur on stamp objects.

You cannot flip, reverse, or rotate a stamp object. If you flip, reverse, or rotate a container that includes a stamp, the object will move to the new position, but will retain its orientation.

Button Tool

Note: This topic is for Express editions only.

The button annotation object is an image of a standard Windows button that can be activated. In design mode, you can click and drag to specify the size of the button in the current window. A dialog box then lets you enter the text for the button. In run mode, clicking on the button triggers an AnnClicked event. You can add code to process the event. (The example program merely pops up a message box.)

In design mode, you can then click on the object with the right mouse button to do the following:

- Undo the last user action.
- Cut or copy the object to the clipboard.
- Delete the object.
- Bring the object to the front or send it to the back.
- Change the foreground color. See Color Dialog Box for Annotations.
- Edit the text. See Text Dialog Box for Annotations.

You cannot flip, reverse, or rotate a button. If you flip, reverse, or rotate a container that includes a button, the button will move to the new position, but will retain its orientation.

Hot Spot Tool

Note: This topic is for Express editions only.

The hot spot annotation object is a rectangle that can be activated for an application-defined purpose. In design mode, the rectangle contains a scaled image. You can click and drag to specify the rectangle. In run mode, the object is transparent, but the mouse pointer changes to a hand when it is over the object. In run mode, clicking on the button triggers an AnnClicked event. You can add code to process the event. (The example program merely pops up a message box.)

In design mode, you can click on the object with the right mouse button to do the following:

- Undo the last user action.
- Cut or copy the object to the clipboard.
- Delete the object.
- Bring the object to the front or send it to the back.

You cannot flip, reverse, or rotate a hot spot. If you flip, reverse, or rotate a container that includes a hot spot, the object will move to the new position, but will retain its orientation.

Audio Clip Tool

Note: This topic is for Express editions only.

The audio clip annotation object is a rectangle containing a scaled image that can be activated to play a WAV file. In design mode, you can click and drag to specify the rectangle in the current window. A dialog box then lets you specify the WAV file. In run mode, clicking on the icon plays the WAV file.

In design mode, you can click on the object with the right mouse button to do the following:

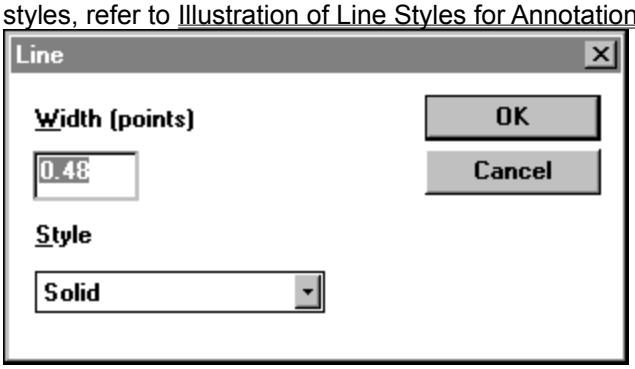
- Undo the last user action.
- Cut or copy the object to the clipboard.
- Delete the object.
- Bring the object to the front or send it to the back.
- Select a WAV file to load into the object. See Audio File Dialog Box for Annotations.

You cannot flip, reverse, or rotate an audio clip. If you flip, reverse, or rotate a container that includes an audio clip, the object will move to the new position, but will retain its orientation.

Line Dialog Box for Annotations

Note: This topic is for Express editions only.

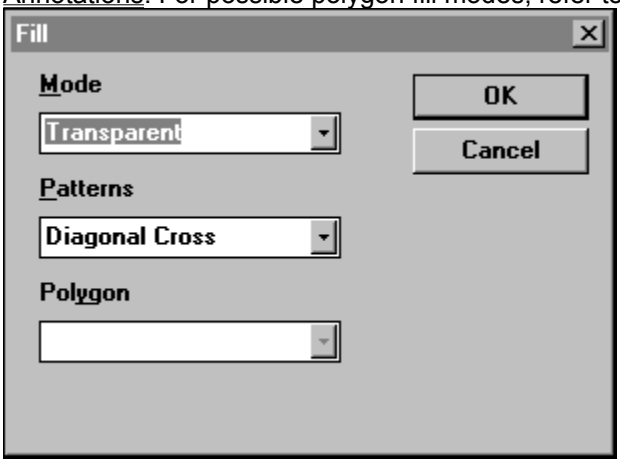
The following dialog box appears in automation mode when you change the line properties of an annotation object. The line width is based on the associated bitmap's physical resolution (DPI), so that it matches the scale of the original scanned image. Setting the width to a small number will not make the line disappear. On any output device, lines will always be at least one pixel wide. To see the possible line styles, refer to Illustration of Line Styles for Annotations.



Fill Dialog Box for Annotations

Note: This topic is for Express editions only.

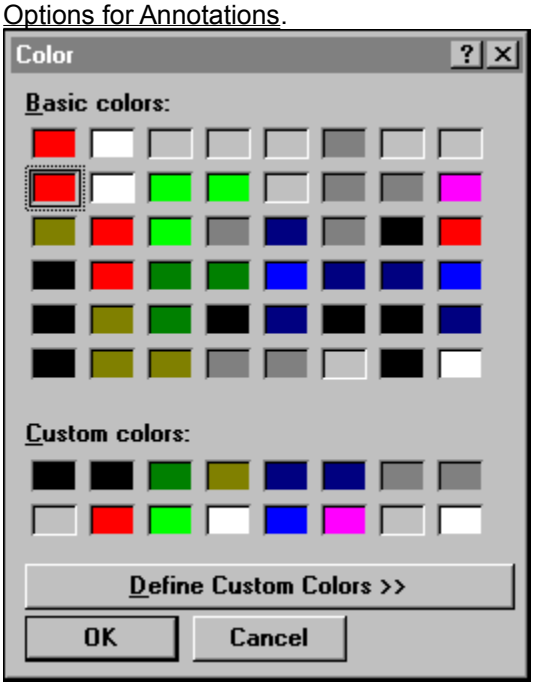
The following dialog box appears in automation mode when you change the fill properties of an annotation object. To see the possible modes and patterns, refer to Illustration of Fill Options for Annotations. For possible polygon fill modes, refer to Illustration of the Polygon Fill Mode for Annotations.



Color Dialog Box for Annotations

Note: This topic is for Express editions only.

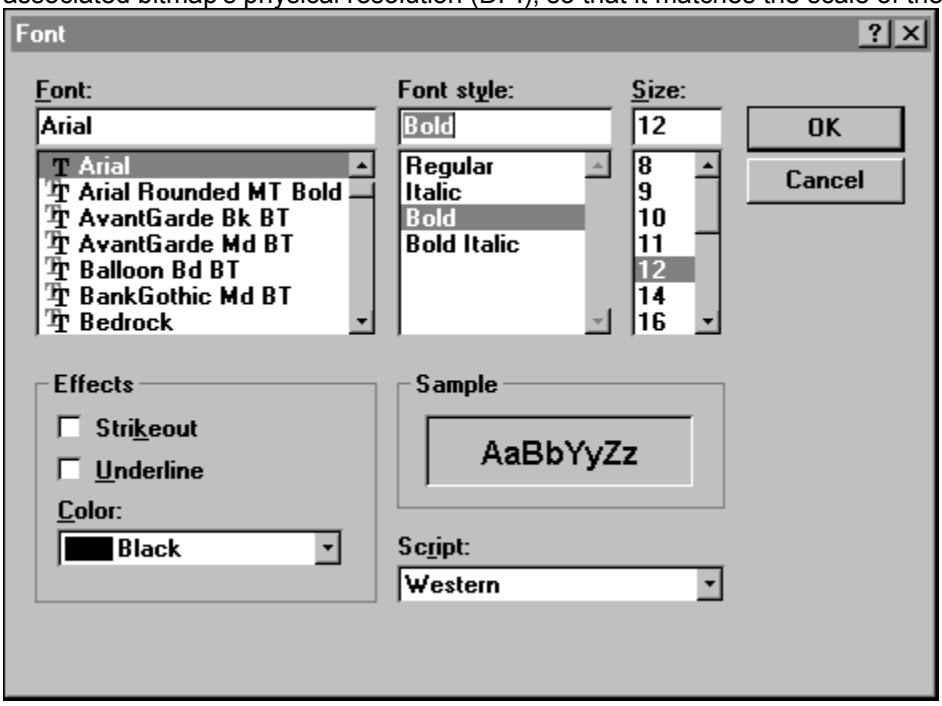
The following dialog box appears in automation mode when you change the foreground or background color property of an annotation object. It is implemented using the Windows common dialog box. To see how the foreground and background colors affect the appearance of an object, refer to Illustration of Fill Options for Annotations.



Font Dialog Box for Annotations

Note: This topic is for Express editions only.

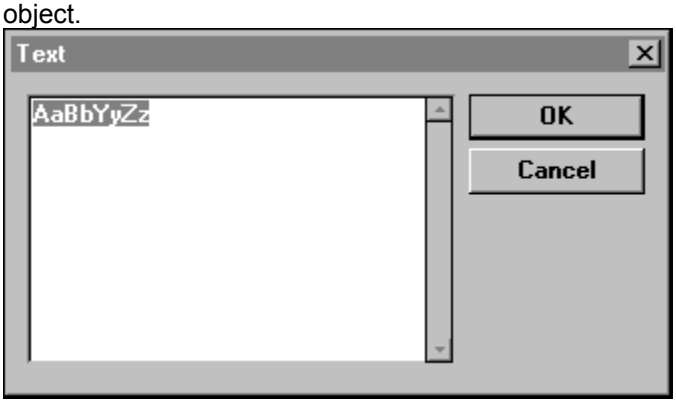
The following dialog box appears in automation mode when you change the font properties of an annotation object. It is implemented using the Windows common dialog box. The font size is based on the associated bitmap's physical resolution (DPI), so that it matches the scale of the original scanned image.



Text Dialog Box for Annotations

Note: This topic is for Express editions only.

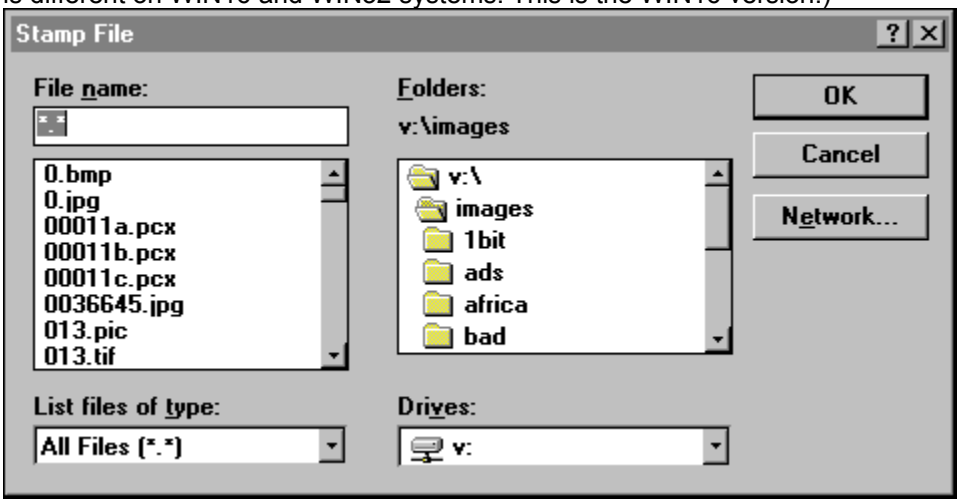
The following dialog box appears in automation mode when you change the text property of an annotation object. Word wrap is implemented in the dialog box and in the rectangle that defines the annotation object.



Stamp Bitmap Dialog Box for Annotations

Note: This topic is for Express editions only.

The following dialog box appears in automation mode when you update the bitmap property of a stamp annotation object. It is implemented using the Windows common dialog box. (The style of this dialog box is different on WIN16 and WIN32 systems. This is the WIN16 version.)



Audio File Dialog Box for Annotations

Note: This topic is for Express editions only.

The following dialog box appears in automation mode when you specify the WAV file for an audio clip annotation object. It is implemented using the Windows common dialog box. (The style of this dialog box is different on WIN16 and WIN32 systems. This is the WIN16 version.)

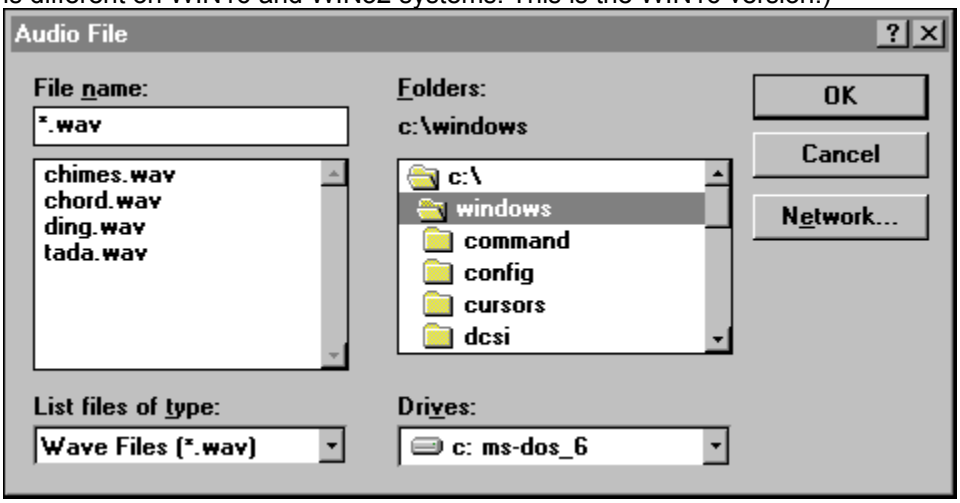


Illustration of Line Styles for Annotations

Note: This topic is for Express editions only.

The following rectangle annotation objects illustrate the available line styles. Note that Windows displays all lines greater than 1 pixel as solid.

Back row:	Solid	Dash	Dot
Front row:	Dash-dot	Dash-dot-dot	Null

Illustration of Fill Options for Annotations

Note: This topic is for Express editions only.

The following rectangle annotation objects illustrate the available fill modes and patterns. The foreground and background colors help to illustrate the effects.

Back row:

Fill mode: Opaque Fill pattern: Solid Fore color: Black Back color: Yellow	Fill mode: Opaque Fill pattern: Cross Fore color: Black Back color: Yellow	Fill mode: Opaque Fill pattern: Backward diagonal Fore color: Black Back color: Yellow	Fill mode: Opaque Fill pattern: Vertical Fore color: Black Back color: Yellow

Front row:

Fill mode: Translucent Fill pattern: Forward diagonal Fore color: Black Back color: Green	Fill mode: Transparent Fill pattern: Diagonal cross Fore color: Black Back color: Green	Fill mode: Opaque Fill pattern: Horizontal Fore color: Red Back color: Green

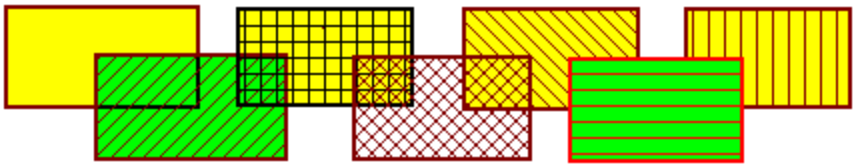


Illustration of the Polygon Fill Mode for Annotations

Note: This topic is for Express editions only.

The following polygon annotation objects illustrate the available polygon fill modes.

Winding	The area inside the resulting exterior lines is filled.
Alternate	The filled area includes the area between odd-numbered and even-numbered polygon sides on each scan line.

Summary of LEADTOOLS Features

This is a summary of LEADTOOLS features for all components:

- DLL functions and Windows messages in the API
- ActiveX (OCX) methods, properties and events
- VBX properties and events

You can use these topics to explore the full range of possibilities for implementing LEADTOOLS capabilities.

General Features

You can use LEADTOOLS to enhance applications such as image databases, printing and drawing programs, video production systems, FAX systems, and systems for transmitting high quality compressed images using phone lines or networks.

Note: Some of the toolkits have more features than others, because of limitations in the development environment, version updates, and edition differences (Express or not). This chapter includes tags (such as [DLL only]) to mark any features that are not available in all toolkits.

The following are general features of the LEADTOOLS toolkits:

- **Scanning.** LEADTOOLS supports 16- and 32-bit TWAIN interfaces for driving the most popular scanning devices. Refer to Working with Other Input and Output.
- **Compression and file format support.** LEADTOOLS supports most popular raster image formats, in different flavors and color depths, and with various compression options. LEAD's own compressed format (LEAD CMP) results in smaller files and better image quality than industry-standard formats. Simple functions let you load from, save to, and convert between the various formats. Refer to Summary of All Supported Image File Formats and Working with Files.
- **Displaying images.** LEADTOOLS renders an image of any color depth (1 to 32 bit) to any display device. You can position the displayed image, and zoom in or out by scaling and cropping the display. On a 256-color device, you can use a fixed palette option to eliminate palette shifting when displaying more than one image at a time. Refer to Working with Displays.
- **Image processing.** LEADTOOLS provides many image processing functions, common ones such as rotating or inverting an image, and sophisticated ones such as edge-detection filters and histogram equalization. For a complete list, refer to Working with the Images in Bitmaps.
- **Color conversion.** LEADTOOLS lets you expand or reduce an images color depth with multiple dithering methods, using customized or optimized palettes. You can separate an image into individual RGB, CMYK, CMY, HSV, and HSL color planes, and you can reconstruct the original image from the separated planes. LEADTOOLS also supports grayscale conversion, and halftoning with rotated screens. Refer to Working with Image Data.
- **[Express editions only] Region processing.** LEADTOOLS lets you apply image processing functions to a portion of a bitmap by specifying a rectangular, polygonal, or elliptical region within the bitmap. Refer to Working with Regions [Express Editions Only].
- **[Express editions only] Annotation.** LEADTOOLS supports annotation of images by letting you overlay images with text, lines, shapes, arrows, buttons, highlights, hot spots, sticky notes, stamps, buttons, and sounds. Refer to Working with Annotations [Express Editions Only].
- **[Express editions only] Special functions for bitonal images.** The LEADTOOLS functions for bitonal images are especially useful for working with images from black-and-white scanners or FAX machines. These functions let you do things such as automatically straighten (deskew) an image, remove spots (despeckle), or clarify a scaled display (using scale-to-gray and favor-black features). Refer to Working with Displays and Working with the Images in Bitmaps.
- **Printing.** LEADTOOLS performs all the image processing necessary to print to any printer with the highest quality, and it can print multiple images on the same page. Refer to Working with Other Input and Output.
- **High-level and low-level functions.** LEADTOOLS lets you choose the level of control you want. You can implement most LEADTOOLS features using high-level functions. With the DLLs, you can go to the lowest level, redirecting the standard input and output, and processing image data line-by-line or pixel-by-pixel.
- **Modular design.** LEADTOOLS provides need-specific DLLs that let you limit the size of your application by including only the applicable DLLs. For example, each supported file format has its own DLL; so you can leave out the formats that you do not wish to support.
- **Multi-thread support.** LEADTOOLS fully supports the development of multi-threaded applications for Windows NT and Windows 95.
- **Code examples.** The LEADTOOLS toolkits have many code examples in the online documentation and in working demonstration programs. You can copy code from the example programs and paste it directly into your application.

Working with Files

LEADTOOLS lets you do the following when working with files on disk and in memory:

- Get information about the image in a file, and set information about how to handle the input.

  DLL functions: L_FileInfo, L_FileInfoMemory, L_ReadFileComment, L_GetPCDResolution, L_SetPCDResolution, L_GetWMFResolution, L_SetWMFResolution, L_SetLoadInfoCallback

  ActiveX elements: GetFileInfo method, InfoBits property, InfoCompress property, InfoFormat property, InfoHeight property, InfoPage property, InfoPCDRes property, InfoSizeDisk property, InfoSizeMem property, InfoWidth property, PCDRes property, PaintWhileLoad property, LoadInfo event, LoadInfoBits property, LoadInfoFlags property, LoadInfoFormat property, LoadInfoHeight property, LoadInfoOffset property, LoadInfoWidth property, LoadInfoXRes property, LoadInfoYRes property

  VBX properties: InfoPage, InfoFile, InfoBits, InfoCompress, InfoFormat, InfoHeight, InfoPCDResolution, InfoScaleHeight, InfoScaleWidth, InfoSizeDisk, InfoSizeMem, InfoWidth, PaintWhileLoad, FilePage, PCDResolution

- Set information about how to save an image to a file.

  DLL functions: L_GetComment, L_SetComment, L_SetExtFileOption

  ActiveX properties: BitmapXRes property, BitmapYRes property

  VBX properties: BitmapXRes, BitmapYRes, SaveBits, SaveFormat, SaveQFactor, SaveMulti

- Load an image from a file into memory as a bitmap, or save a bitmap to a file. The image can be one of several in a multipage file. In a JFIF or LEAD CMP file, it can be either the main image or a thumbnail stored in the same file.

  DLL functions: L_LoadBitmap, L_LoadFile, L_LoadFileOffset, L_GetCompressFileStamp, L_SaveBitmap, L_SaveFile, L_SaveFileOffset, L_CompressBitmapWithStamp

  Windows messages: L_BM_LOAD, L_BM_SAVE

  ActiveX methods: Load method, LoadStamp method, Save method, SaveWithStamp method

  VBX properties: File, LoadStamp, SaveFile, SaveBits, SaveFormat, SaveQFactor, SaveStamp, SaveStampBits, SaveStampHeight, SaveStampWidth

- Convert a file from one format to another.

  DLL functions: The load and save functions, plus L_FileConvert

  ActiveX elements: Implemented with load and save.

  VBX properties: Implemented with load and save.

- [DLL and ActiveX only] Load a file, as is, into memory (to take advantage of file compression), then use the file in memory (to take advantage of memory speed).

  DLL functions: L_FileInfoMemory, L_LoadBitmapMemory, L_LoadMemory, L_SaveBitmapMemory, L_DecompressBitmapMemory, L_DecompressMemory

  ActiveX methods: GetMemoryInfo method, LoadMemory method, SaveMemory method

- [DLL and ActiveX only] Control how data is fed to a load routine. This is useful when receiving transmitted images, such as those on the Internet.

  DLL functions: L_StartFeedLoad, L_FeedLoad, L_StopFeedLoad

  ActiveX methods: StartFeedLoad method, FeedLoad method, StopFeedLoad method

Working with Image Data

LEADTOOLS uses a structure called a bitmap handle to manage image data in memory. The bitmap handle has several fields, one of which is an internal pointer to the bitmap (the actual image data). In the DLL toolkit, your code references the structure directly; in the ActiveX or VBX your code references the LEAD control.

To work with bitmaps and bitmap handles, you can do the following:

- Manipulate the bitmap storage.

  DLL functions: L_ClearBitmap, L_FreeBitmap, L_AccessBitmap, L_ReleaseBitmap

  Windows messages: L_BM_CLEAR, L_BM_FREE

  ActiveX properties: Bitmap property

  VBX properties: Bitmap
- Load a bitmap from a file or other input source. (Refer to Working with Files and Working with Other Input and Output.)
- Obtain or update the information in a bitmap handle. (For palette information, refer to Working with Palettes.)

  DLL functions: L_IsGrayScaleBitmap, and access the bitmap handle structure directly.

  Windows messages: L_BM_INFO, L_BM_GETBITMAP, L_BM_SETBITMAP

  ActiveX properties: BitmapBits property, BitmapHeight property, BitmapWidth property, BitmapXRes property, BitmapYRes property, IsGrayScale property

  VBX properties: BitmapXRes, BitmapYRes, BitmapHeight, BitmapScaleHeight, BitmapScaleWidth, BitmapWidth
- Create a bitmap from scratch.

  DLL functions: L_CreateBitmap, L_InitBitmap, L_AllocateBitmap

  ActiveX elements: BitmapXRes property, BitmapYRes property, CreateBitmap method

  VBX properties: BitmapXRes, BitmapYRes, CreateBitmap, CreateBits, CreateHeight, CreateWidth
- Copy a bitmap to or from the clipboard.

  DLL functions: L_ClipboardReady, L_CopyFromClipboard, L_CopyToClipboard, L_CopyToClipboardExt

  Windows messages: L_BM_COPY, L_BM_PASTE

  ActiveX methods: Copy method, Paste method

  VBX properties: Copy, Paste
- Make a copy of a bitmap handle, or its bitmap, or both.

  DLL functions: L_CopyBitmap, L_CopyBitmapData, L_CopyBitmapHandle, L_CopyBitmapRect

  Windows messages: L_BM_COPYBITMAP

  ActiveX properties: Bitmap property

  VBX properties: Bitmap
- Convert or copy a bitmap to or from a device independent bitmap (DIB) or a device dependent bitmap (DDB).

  DLL functions: L_ConvertFromDDB, L_ConvertFromDIB, L_ConvertToDDB, L_ConvertToDIB, L_ChangeFromDDB, L_ChangeFromDIB, L_ChangeToDDB, L_ChangeToDIB

  ActiveX methods: GetDDB method, GetDIB method, SetDDB method, SetDIB method

VBX properties: DDB, DIB

- [In Visual Basic only] Copy a bitmap to or from a picture object.

  ActiveX methods: GetPicture method, SetPicture method

  VBX properties: PictureCreate, Picture
- Resize a bitmap, with or without resampling.

  DLL functions: L_ResampleBitmap, L_ResizeBitmap, L_SizeBitmap

  Windows messages: L_BM_RESIZE, L_BM_RESAMPLE

  ActiveX methods: Size method

  VBX properties: ResizeType, ResizeHeight, ResizeWidth, Resize
- Trim a bitmap.

  DLL functions: L_CopyBitmapRect

  ActiveX methods: Trim method

  VBX properties: TrimHeight, TrimLeft, TrimTop, TrimWidth, Trim
- Change the bits per pixel or color order of a bitmap (with several possible dithering methods used for color reduction).

  DLL functions: L_ColorResBitmap, L_DefaultDithering

  Windows messages: L_BM_COLORRES

  ActiveX methods: ColorRes method

  VBX properties: BitChgDither, BitChgOrder, BitChgPalette, BitChgUserPal, BitmapBits
- Specify a bitmaps palette. (Refer to Working with Palettes.)
- Change a bitmap to grayscale.

  DLL functions: L_GrayScaleBitmap

  Windows messages: L_BM_GRAYSCALE

  ActiveX methods: Grayscale method

  VBX properties: GrayScale
- Create and merge color separations. A color separation is a set of grayscale images, one for each plane of the color-space model. The supported color-space models are RGB, CMYK, CMY, HSV, and HLS.

  DLL functions: L_ColorSeparateBitmap, L_ColorMergeBitmap

  ActiveX elements: ColorSeparate method, ColorMerge method, ColorPlanes property

  VBX properties: ColorSeparate, ColorPlanes, ColorMerge
- Modify the appearance of a bitmap image, as described in Working with the Images in Bitmaps.
- Display a bitmap image, with options for dithering, positioning, and zooming, as described in Working with Displays.

Working with the Images in Bitmaps

Most of the LEADTOOLS functions that change the appearance of an image act on the bitmap (not just the display). This is an important distinction, because the changes become permanent when the bitmap is saved to a file.

To change the appearance of the image in a bitmap, you can do the following:

- Define the bitmap as a display surface where you can use Windows graphics device interface (GDI) functions for drawing or adding text.

  DLL functions: L_CreateLeadDC, L_DeleteLeadDC

  ActiveX methods: GetBitmapDC method, ReleaseBitmapDC method

  VBX properties: BitmapDC
- [Express editions only] Automatically straighten (deskew) a 1-bit image.

  DLL functions: L_DeskewBitmap

  Windows messages: L_BM_DESKEW

  ActiveX methods: Deskew method

  VBX properties: Deskew
- [Express editions only] Remove specks (despeckle) a 1-bit image.

  DLL functions: L_DespeckleBitmap

  Windows messages: L_BM_DESPECKLE

  ActiveX methods: Despeckle method

  VBX properties: Despeckle
- Rotate the image. (Angles can be precise to 100th of a degree.)

  DLL functions: L_RotateBitmap, L_RotateBitmapFine

  Windows messages: L_BM_ROTATE

  ActiveX methods: Rotate method

  VBX properties: Rotate, RotateFine
- Shear the image in the fashion of a parallelogram. (Angles are precise to 100th of a degree.)

  DLL functions: L_ShearBitmap

  Windows messages: L_BM_SHEAR

  ActiveX methods: Shear method

  VBX properties: ShearX, ShearY
- Change the orientation by flipping the image horizontally or vertically.

  DLL functions: L_FlipBitmap, L_ReverseBitmap

  Windows messages: L_BM_FLIP, L_BM_REVERSE

  ActiveX methods: Flip method, Reverse method

  VBX properties: Flip, Reverse
- Adjust colors and intensities as follows:

  Change brightness using a flat scale.
  Change brightness using gamma correction.
  Change contrast.
  Stretch the range of intensities.
  Remap intensities using a lookup table.

Invert colors.
Change hue and saturation.
Histogram equalize.
Fill with a specified color.
[DLL only] Get and put colors of individual pixels.

DLL functions: L_ChangeBitmapIntensity, L_GammaCorrectBitmap, L_ChangeBitmapContrast, L_HistoContrastBitmap, L_StretchBitmapIntensity, L_RemapBitmapIntensity, L_InvertBitmap, L_ChangeBitmapHue, L_ChangeBitmapSaturation, L_HistoEqualizeBitmap, L_FillBitmap, L_GetPixelColor, L_PutPixelColor

Windows messages: L_BM_INTENSITY, L_BM_GAMMACORRECT, L_BM_CONTRAST, L_BM_INVERT, L_BM_HUE, L_BM_SATURATION, L_BM_HISTOEQ, L_BM_FILL

ActiveX elements: Intensity method, GammaCorrect method, Contrast method, HistoContrast method, StretchIntensity method, RemapIntensity method, RemapTable property, Invert method, Hue method, Saturation method, HistoEqualize method, Fill method

VBX properties: Intensity, GammaCorrect, Contrast, HistoContrast, StretchIntensity, RemapTable, RemapChannel, RemapIntensity, Invert, Hue, Saturation, HistoEq, Fill

- Apply the following kinds of conversions:

  Halftone for display or printing.
  Sharpen or blur.
  Posterize, specifying the number of color planes.
  Mosaic, specifying the tile size.
  Emboss, specifying the lighting direction.
  Soften an image using an average filter.
  Reduce noise using a median filter.
  Add noise in any or all color planes.

  DLL functions: L_HalfToneBitmap, L_HalftoneBitmapExt, L_SharpenBitmap, L_PosterizeBitmap, L_MosaicBitmap, L_EmbossBitmap, L_AverageFilterBitmap, L_MedianFilterBitmap, L_AddBitmapNoise

  Windows messages: L_BM_HALFTONE, L_BM_SHARPEN, L_BM_POSTERIZE, L_BM_MOSAIC, L_BM_AVERAGE, L_BM_MEDIAN, L_BM_ADDNOISE

  ActiveX methods: Halftone method, Sharpen method, Posterize method, Mosaic method, Emboss method, Average method, Median method, AddNoise method

  VBX properties: HalftoneAngle, HalfTone, Sharpness, Posterize, Mosaic, FltEmbossDepth, FltEmboss, Average, Median, AddNoiseChannel, AddNoise

- Apply filters to do the following:

  Detect ranges of intensity.
  Detect edges using gradient or Laplacian edge detection.
  Detect lines using Sobel, Prewitt, shift and difference, or line segment detection.

  DLL functions: L_IntensityDetectBitmap, L_SpatialFilterBitmap

  Windows messages: L_BM_INTENSITYDETECT, L_BM_SPATIALFLT

  ActiveX methods: IntensityDetect method, SpatialFilter method

  VBX properties: IntensityDetectHigh, IntensityDetectLow, IntensityDetect, FltGradient, FltLaplace, FltLineSeg, FltPrewitt, FltShiftDiff, FltSobel

- [Express editions only] Apply morphological (binary) filters to erode or dilate black objects.

  DLL functions: L_BinaryFilterBitmap

  Windows messages: L_BM_BINARYFLT

  ActiveX methods: BinaryFilter method

VBX properties: FltDilation, FltErosion

- [DLL only] Implement your own spatial filters.

  DLL functions: L_SpatialFilterBitmap

- Combine images using boolean and arithmetic operators, and color masking. (This is useful for combining filtered images with originals.)

  DLL functions: L_CombineBitmap

  ActiveX methods: Combine method

  VBX properties: CombineDstLeft, CombineDstTop, CombineHeight, CombineOp, CombineSrcLeft, CombineSrcTop, CombineWidth, Combine

- Combine two images so that one appears to be an underlying texture of the other.

  DLL functions: L_UnderlayBitmap

  ActiveX methods: Underlay method

  VBX properties: UnderlayType, Underlay

- [DLL only] Use low-level functions to change parts of an image as follows:

  Get and put rows of image data.
  Get and put parts of rows.
  Get and put the colors of individual pixels.

  DLL functions: L_GetBitmapRow, L_PutBitmapRow, L_GetBitmapRowCol, L_PutBitmapRowCol, L_GetPixelColor, L_PutPixelColor

- [Express Editions Only] Process a region within a bitmap, as described in Working with Regions [Express Editions Only].

Working with Palettes

Images that are less than 16 bits per pixel require a palette, which is an array of color values. The value of each pixel in one of these images is an index into the palette. (Images that are 16 bits per pixel or higher store the color values directly in the image, as pixel values.)

The number of colors in a palette is limited to the highest possible index value. For example, if an image is 8 bits per pixel, it can have a palette of 256 colors, which is the number of different colors that can be referenced by using the pixel values as 8-bit indexes.

If an image file is less than 16 bits per pixel, the file contains a palette, and when you load the image into a bitmap, the palette is also loaded into the bitmap handle.

Whenever you reduce a bitmap to less than 16 bits per pixel, a palette must be specified or created. You can do any of the following:

- Let LEADTOOLS create a palette that is optimized for the particular image. (As an option, this palette can include the system identity palette.)

  DLL functions: The load functions, plus L_ColorResBitmap

  Windows messages: L_BM_COLORRES

  ActiveX elements: ColorRes method, UserPalette property

  VBX properties: BitChgPalette, BitmapBits
- Specify a fixed palette, which is the same for all images that use it.

  DLL functions: The load functions, plus L_GetFixedPalette, L_ColorResBitmap

  Windows messages: L_BM_COLORRES

  ActiveX methods: ColorRes method

  VBX properties: BitChgPalette, BitmapBits
- Specify a palette of any size that you have created. In creating your own palette, you can let LEADTOOLS do part of the work. For example, you can fill in part of the palette, and let LEADTOOLS fill the rest with optimized colors.

  DLL functions: L_CreatePaintPalette

  ActiveX properties: UserPalette property

  VBX properties: BitChgPalette, BitChgUserPal, BitmapBits
- Modify the colors in the bitmap palette.

  DLL functions: L_GetBitmapColors, L_PutBitmapColors

  Windows messages: L_BM_GETPALETTE

  ActiveX properties: BitmapPalette property

  VBX properties: BitmapPalette
- [DLL and ActiveX only] Copy a palette from a device context.

  DLL functions: L_GetFixedPalette, L_DupPalette

  ActiveX methods: GetPalette method
- [DLL only] Use a table to increase the efficiency of the palette that you define.

  DLL functions: L_CreateUserMatchTable, L_SetUserMatchTable, L_FreeUserMatchTable

Working with Displays

When painting images on any video device, you can do the following:

- Get and set properties that control the behavior of the LEAD control.

  Windows messages: L_BM_SETSTYLE

  ActiveX properties: Standard control properties, plus AutoRepaint property, AutoScroll property, AutoSetRects property, BackColor property, BackErase property, BorderStyle property, ForePalette property, MouseIcon property, MousePointer property, Enabled property, hWnd property

  VBX properties: Align, AutoRepaint, AutoScroll, BackColor, BackErase, BorderStyle, DragIcon, DragMode, Enabled, MousePointer, hWnd, HelpContextID, Index, Name, Parent, PropertySet, TabIndex, TabStop, Tag, Visible

- Get the client area of the LEAD control.

  DLL functions: Implemented using Windows functions.

  ActiveX properties: ScaleMode property, ScaleHeight property, ScaleLeft property, ScaleTop property, ScaleWidth property

  VBX properties: Height, Left, Top, Width

- Define the client area as a display surface where you can use Windows graphics device interface (GDI) functions for drawing or adding text.

  DLL functions: Implemented using Windows functions.

  ActiveX methods: GetClientDC method, ReleaseClientDC method

  VBX properties: hDC

- [ActiveX only] Display a rubberband to mark an area.

  ActiveX elements: SetRubberBandRect method, RubberBandVisible property, RubberBandHeight property, RubberBandLeft property, RubberBandTop property, RubberBandWidth property

- Force repainting of the image in a LEAD control.

  DLL functions: Implemented using **Windows functions.**

  Windows messages: L_BM_BITMAPCHANGED

  ActiveX methods: ForceRepaint method

  VBX properties: Repaint

- Scale and position the image. This lets you zoom in or out, fit the image to a window, and position the image in a window.

  DLL functions: L_PaintDC, L_PaintDCBuffer, L_PaintDCEffect

  Windows messages: L_BM_SETIMAGERECT, L_BM_SETZOOM, L_BM_FOCUSEDZOOM, L_BM_SETBITMAPCLIP

  ActiveX elements: SetDstRect method, SetSrcRect method, DstHeight property, DstLeft property, DstTop property, DstWidth property, SrcHeight property, SrcLeft property, SrcTop property, SrcWidth property

  VBX properties: ImageHeight, ImageLeft, ImageTop, ImageWidth

- Limit the area to be painted by specifying the source and destination cropping areas.

  DLL functions: L_PaintDC, L_PaintDCBuffer, L_PaintDCEffect

  ActiveX elements: SetDstClipRect method, SetSrcClipRect method, DstClipHeight property, DstClipLeft property, DstClipTop property, DstClipWidth property, SrcClipHeight property, SrcClipLeft property, SrcClipTop property, SrcClipWidth property

VBX properties: CropHeight, CropLeft, CropTop, CropWidth, Crop

- Apply transitional effects, commonly used in slide presentations.

  DLL functions: L_PaintDCEffect

  ActiveX properties: PaintEffect property

  VBX properties: Effect

- Apply ROP codes, which determine how to interact with the existing image on the screen.

  DLL functions: L_PaintDC, L_PaintDCBuffer, L_PaintDCEffect

  ActiveX properties: PaintROP3 property

  VBX properties: PaintRop3

- [DLL only] Use a buffer as the source to paint from.

  DLL functions: L_PaintDCBuffer

- [DLL only] Specify an option for fast painting (without device error checking).

  DLL functions: L_GetDisplayMode, L_SetDisplayMode

If the display mode is 256 colors or less, you can do the following:

- Use the palette in the bitmap handle or use the system fixed palette.

  DLL functions: L_GetDisplayMode, L_SetDisplayMode

  ActiveX properties: PaintPalette property

  VBX properties: PaintPalette

- Specify the dithering method, for images that have more bits per pixel than the current video mode. The following are possible options:

  Normal dithering (using error diffusion).
  Ordered dithering, which is faster but less accurate than normal.
  [DLL only] No dithering, which relies on the display device for color reduction.

  DLL functions: L_GetDisplayMode, L_SetDisplayMode

  ActiveX properties: PaintDither property

  VBX properties: PaintDither

For displaying 1-bit (black-and-white) images, you can do the following to enhance the quality of display:

- [Express editions only] Specify a scale-to-gray option, which increases the clarity of the 1-bit images when they are scaled (zoomed in or zoomed out).

  DLL functions: L_GetDisplayMode, L_SetDisplayMode

  ActiveX properties: BitonalScaling property

  VBX properties: PaintBwAsGray, BitonalScaling

- [Express editions only] Specify a favor-black option, which prevents loss of details, such as fine lines, when an image is scaled down (zoomed out).

  DLL functions: L_GetDisplayMode, L_SetDisplayMode

  ActiveX properties: BitonalScaling property

  VBX properties: BitonalScaling

Working with Regions [Express Editions Only]

You can create and maintain shapes, position a shape on a bitmap to define a region to be processed, and use image-processing functions to modify the region. Specifically, you can do the following.

- [ActiveX only] Automate the creation and display of a region.

  ActiveX elements: RgnMarkingMode property, RgnChange event, RgnFrameType property

- [DLL and ActiveX only] Copy and paste a region using a floating bitmap.

  DLL functions: Implemented using lower-level functions.

  ActiveX elements: Floater property, FloaterVisible property, IsPtInFloater method, SetFloaterDstRect method, SetFloaterDstClipRect method, FloaterDstHeight property, FloaterDstLeft property, FloaterDstTop property, FloaterDstWidth property, FloaterDstClipHeight property, FloaterDstClipLeft property, FloaterDstClipTop property, FloaterDstClipWidth property, FloaterHeight property, FloaterWidth property, GetFloaterHandle method

- [DLL and ActiveX only] Create a shape using a rectangle, ellipse, polygon, or specified color.

  DLL functions: L_SetBitmapRgnColor, L_SetBitmapRgnEllipse, L_SetBitmapRgnPolygon, L_SetBitmapRgnRect, L_SetBitmapRgnRoundRect

  ActiveX elements: SetRgnEllipse method, SetRgnRect method, SetRgnRoundRect method, SetRgnColor method, PolygonSize property, PolygonX property, PolygonY property, SetRgnPolygon method

- [DLL and ActiveX only] Size and position the region on the bitmap.

  DLL functions: L_GetBitmapRgnBounds, L_OffsetBitmapRgn

  ActiveX elements: RgnHeight property, RgnLeft property, RgnTop property, RgnWidth property, OffsetRgn method

- [DLL and ActiveX only] Manage the bitmap's region.

  DLL functions: L_FrameBitmapRgn, L_BitmapHasRgn, L_FreeBitmapRgn, L_GetBitmapRgnArea, L_IsPtInBitmapRgn

  ActiveX elements: HasRgn property, IsPtInRgn method, FreeRgn method, GetRgnArea method, RepaintRect method, RgnFrameType property

- [DLL and ActiveX only] Copy to and from standard Windows regions.

  DLL functions: L_GetBitmapRgnHandle, L_SetBitmapRgnHandle

  ActiveX elements: GetRgnHandle method, GetFloaterHandle method, SetRgnHandle method, DeleteRgnHandle method

- [DLL only] Paint only the bitmap's region.

  DLL functions: L_PaintRgnDC, L_PaintRgnDCBuffer, L_PaintRgnDCEffect

- [DLL and ActiveX only] Process the region using any of the appropriate functions described in Working with the Images in Bitmaps.

Working with Annotations [Express Editions Only]

You can create a wide range annotation objects, group them, store them in files, apply them to bitmaps, size and position them, and rotate them. Specifically, you can do the following.

- [DLL and ActiveX only] Create any of the following objects: line, arrow, rectangle, ellipse, polygon, freehand line, text, highlighter, hot spot, note, stamp, redaction, button, or audio clip.

  DLL functions: L_AnnDraw, L_AnnCreate, L_AnnCreateContainer, L_AnnCreateItem, L_AnnCopy, L_AnnDestroy, L_AnnResize, L_AnnSetVisible, L_AnnGetVisible, L_AnnSetSelected, L_AnnGetSelected, L_AnnSetWnd, L_AnnGetWnd, L_AnnGetType, L_AnnGetActiveState, L_AnnSetActiveState, L_AnnSetText, L_AnnGetText, L_AnnGetTextLen, L_AnnSetFontName, L_AnnGetFontName, L_AnnGetFontNameLen, L_AnnSetFontSize, L_AnnGetFontSize, L_AnnSetFontUnderline, L_AnnGetFontUnderline, L_AnnSetFontStrikeThrough, L_AnnGetFontStrikeThrough, L_AnnSetFontItalic, L_AnnGetFontItalic, L_AnnSetFontBold, L_AnnGetFontBold, L_AnnSetForeColor, L_AnnGetForeColor, L_AnnSetBackColor, L_AnnGetBackColor, L_AnnSetLineWidth, L_AnnGetLineWidth, L_AnnSetLineStyle, L_AnnGetLineStyle, L_AnnSetFillPattern, L_AnnGetFillPattern, L_AnnSetFillMode, L_AnnGetFillMode, L_AnnGetPolyFillMode, L_AnnSetPolyFillMode, L_AnnSetBitmap, L_AnnGetBitmap, L_AnnSetTag, L_AnnGetTag

  ActiveX elements: Automated support, plus AnnSetTag method, AnnGetTag method, AnnGetType method, AnnContainer property
- [DLL and ActiveX only] Implement automated annotation support.

  DLL functions: L_AnnCreateContainer, L_AnnCreate, L_AnnSetAutoContainer, L_AnnGetAutoContainer, L_AnnSetDpiX, L_AnnSetDpiY, L_AnnGetDpiX, L_AnnGetDpiY, L_AnnSetTool, L_AnnGetTool, L_AnnGetUserMode, L_AnnSetUserMode, L_AnnSetUndoDepth, L_AnnUndo, L_AnnCreateToolBar, L_AnnGetToolBarChecked, L_AnnSetToolBarChecked, L_AnnSelectRect, L_AnnGetSelectCount, L_AnnSelectPoint, L_AnnHitTest, L_AnnDefine, L_AnnGetBoundingRect, L_AnnGetSelectRect

  ActiveX elements: AnnUserMode property, AnnTool property, AnnClicked event, AnnCreate event, AnnDestroy event, AnnDrawn event
- [DLL and ActiveX only] Scale and position a group to fit the displayed bitmap.

  DLL functions: L_AnnSetScalarX, L_AnnSetScalarY, L_AnnGetScalarX, L_AnnGetScalarY, L_AnnSetOffsetX, L_AnnSetOffsetY, L_AnnGetOffsetX

  ActiveX elements: Implemented with automated annotation.
- [DLL and ActiveX only] Group and ungroup objects.

  DLL functions: L_AnnInsert, L_AnnRemove, L_AnnGetContainer, L_AnnGetTopContainer, L_AnnEnumerate

  ActiveX elements: Implemented with automated annotation.
- [DLL and ActiveX only] Apply an object or group to a bitmap.

  DLL functions: L_AnnRealize

  ActiveX elements: AnnRealize method
- [DLL and ActiveX only] Size and position an object or group.

  DLL functions: L_AnnBringToFront, L_AnnSendToBack, L_AnnSetRect, L_AnnGetRect, L_AnnSetPoints, L_AnnGetPointCount, L_AnnGetPoints

  ActiveX elements: Implemented with automated annotation.
- [DLL and ActiveX only] Rotate an object or group.

  DLL functions: L_AnnRotate, L_AnnMove

  ActiveX elements: Automated annotation, plus AnnRotate method
- [DLL and ActiveX only] Flip, reverse, or shear an object or group.

DLL functions: L_AnnFlip, L_AnnReverse, L_AnnSetPoints, L_AnnGetPointCount, L_AnnGetPoints

ActiveX elements: Automated annotation, plus AnnFlip method, AnnReverse method

- [DLL and ActiveX only] Save an object or group on the Windows clipboard.

  DLL functions: L_AnnCopyFromClipboard, L_AnnCopyToClipboard, L_AnnCutToClipboard, L_AnnClipboardReady

  ActiveX elements: AnnCopy method, AnnPasteReady property, AnnPaste method
- [DLL and ActiveX only] Print an object or group.

  DLL functions: L_AnnPrint

  ActiveX elements: Implemented with automated annotation.
- [DLL and ActiveX only] Save an object or group in a file.

  DLL functions: L_AnnSave, L_AnnSaveMemory, L_AnnSaveOffset, L_AnnLoad, L_AnnLoadMemory, L_AnnLoadOffset

  ActiveX elements: AnnLoad method, AnnSave method
- [DLL only] Save an object or group as a file in memory. (This is useful for database support.)

  DLL functions: L_AnnSaveMemory, L_AnnLoadMemory
- [ActiveX only] Save an object or group in a database using ODBC or Visual Basic data binding.

  ActiveX elements: Implemented with automated annotation.

Working with Databases

You can maintain images in a database, using any valid file format to store an image in a long binary field. You can do so as follows:

- [ActiveX32 only] Use Open Database Connectivity (ODBC).

  ActiveX elements: dbOpen method, dbClose method, dbMove method, dbMoveFirst method, dbMoveLast method, dbMoveNext method, dbMovePrev method, dbLoadBits property, dbAddNew method, dbEdit method, dbLockingMode property, dbUpdate method, dbDelete method, dbRequery method, dbCanAppend property, dbCanRestart property, dbCanScroll property, dbCanUpdate property, dbCurrentRecord property, dbEditMode property, dbIsBOF property, dbIsDeleted property, dbIsEOF property, dbIsOpen property, dbRecordCount property

- [ActiveX and VBX with Visual Basic only] Use data binding.

  ActiveX properties: DataSource property, DataField property, DataSaveBits property, DataSaveFormat property, DataSaveQuality property, DataLoadBits property, DataChanged property

  VBX properties: DataChanged, DataField, DataSource

- [DLL and ActiveX only] Use functions to read files from memory and write files to memory.

  DLL functions: L_FileInfoMemory, L_LoadBitmapMemory, L_LoadMemory, L_SaveBitmapMemory, L_DecompressBitmapMemory, L_DecompressMemory

  ActiveX methods: GetMemoryInfo method, LoadMemory method, SaveMemory method

Working with Other Input and Output

For additional input and output, you can do the following:

- Load a bitmap from a TWAIN device. TWAIN is an industry standard interface used with scanners, digital cameras, and frame grabbers. LEADTOOLS supports device selection and multipage scanning.

  DLL functions: L_TwainAcquire, L_TwainAcquireExt, L_TwainEnumSources, L_TwainSelect, L_TwainSetProps

  Windows messages: L_BM_TWAINSELECT, L_BM_TWAINACQUIRE

  ActiveX elements: TwainAcquire method, TwainSelect method, EnableTwainEvent property, TwainPage event, TwainFlags property, TwainEnumSources method, TwainSourceCount property, TwainSourceList property, TwainAppAuthor property, TwainAppFamily property, TwainAppName property, TwainBits property, TwainFrameHeight property, TwainFrameLeft property, TwainFrameTop property, TwainFrameWidth property, TwainMaxPages property, TwainPixelType property, TwainRes property, TwainSourceName property, TwainRealize method

  VBX properties: EnableTwainMulti, TwainAcquire, TwainSelect, TwainFlags, TwainImage event
- Print one or more bitmaps on a page, positioning, scaling, and cropping the images as necessary.

  DLL functions: L_PrintBitmap, L_PrintBitmapExt, L_PrintBitmapFast

  Windows messages: L_BM_PRINT

  ActiveX methods: Render method, PrintStart method, PrintEnd method

  VBX properties: PrintHeight, PrintLeft, PrintTop, PrintWidth, PrintCrop, PrintCropHeight, PrintCropLeft, PrintCropTop, PrintCropWidth, PrintBitmap
- Capture an image from a screen.

  DLL functions: L_ScreenCaptureBitmap

  ActiveX methods: Capture method

  VBX properties: CaptureHeight, CaptureLeft, CaptureTop, CaptureWidth, Capture
- [VBX only] Share data with other applications using DDE links.

  VBX properties: LinkItem, LinkMode, LinkTimeout, LinkTopic
- [DLL only] Redirect the standard input and output functions so that you can replace the low-level file I/O with your own input and output streams.

  DLL functions: L_RedirectIO

Working with Events and Callbacks

To detect user interrupts or add other processing to the LEADTOOLS internal loops, you can write code for events in the ActiveX and VBX, and for callback functions in the DLL. Specifically, you can do the following:

- Provide a generic function for detecting user interrupts and updating a status bar.

  DLL functions: L_SetStatusCallBack

  ActiveX elements: EnableProgressEvent property, ProgressStatus event

  VBX properties: EnableStatus, Status event
- Handle multipage scanning.

  DLL functions: L_TwainAcquireExt

  ActiveX elements: EnableTwainEvent property, TwainPage event

  VBX properties and events: EnableTwainMulti, TwainImage event
- [DLL only] Provide other function-specific callbacks.

  DLL functions: L_ColorResBitmap, L_DecompressMemory, L_LoadFile, L_LoadFileOffset, L_LoadMemory, L_SaveFile, L_SaveFile, L_SaveFileOffset, L_SetLoadInfoCallback, L_StartCompressBuffer, L_TwainEnumSources
- [ActiveX only] Detect changes in regions.

  ActiveX event: RgnChange event
- [ActiveX only] Turn on or turn off the generation of error events.

  ActiveX properties: EnableMethodErrors property
- Detect standard events.

  DLL functions: Implemented using Windows messages.

  Windows messages: L_BN_CHANGE, L_BN_CLICKED, L_BN_DBLCLK, L_BN_HSCROLL, L_BN_KILLFOCUS, L_BN_PAINTED, L_BN_SETFOCUS, L_BN_UPDATE, L_BN_VSCROLL

  ActiveX events: Change event, Click event, DblClick event, KeyDown event, KeyPress event, KeyUp event, MouseDown event, MouseMove event, MouseUp event, Paint event, Resize event, Scroll event

  VBX events: Change, Click, DblClick, DragDrop, DragOver, GotFocus, KeyDown, KeyPress, KeyUp, LinkChange, LinkClose, LinkError, LinkNotify, LinkOpen, LostFocus, MouseDown, MouseMove, MouseUp, Paint, Resize, Scroll

Working with Raw Image Data [DLL Only]

LEADTOOLS provides low-level functions that you can use to process image data in a buffer. You can use these functions to create your own image processing functions. For example, you can write a callback function that paints an image while it is loading by processing and painting one line at a time.

To work with raw image data, you can do the following:

- Paint a buffer to a device context.

  DLL functions: L_PaintDCBuffer
- Change the bits per pixel.

  DLL functions: L_ConvertBuffer, L_StartDithering, L_DitherLine, L_StopDithering
- Change the color order.

  DLL functions: L_ConvertBuffer
- Compress or decompress the data when working with LEAD CMP or JPEG files.

  DLL functions: L_StartCompressBuffer, L_CompressBuffer, L_EndCompressBuffer, L_DecompressMemory
- Process the data for buffered resizing of the bitmap.

  DLL functions: L_StartResize, L_Resize, L_StopResize
- Change the color-space model to or from any of the following: RGB, YUV, CMYK, CMY, YIQ, HSV, or HLS.

  DLL functions: L_ConvertColorSpace
- Modify the allocated height of a bitmap for mid-stream adjustments.

  DLL functions: L_ChangeBitmapHeight

Working with Image Descriptions and Other Information

LEADTOOLS lets you get information as follows:

- You can get the information you need about an image in a file, such as its format, size, bits per pixel, compression method, size on disk, and size in memory. Refer to Working with Files.
- You can get information about a bitmap in memory, such as its width, height, bits per pixel, memory allocation, and palette usage. Refer to Working with Image Data.
- You can create a histogram that maps the frequency of occurrence of each color in a bitmap.

  DLL functions: L_GetBitmapHistogram

  ActiveX elements: HistogramTable property, GetHistogram method

  VBX properties: Histogram, HistogramAcquire, HistogramChannel
- You can get information about the current version of your toolkit.

  DLL functions: L_VersionInfo

  ActiveX elements: AboutBox method, VersionDate property, VersionLevel property, VersionMajor property, VersionMinor property, VersionProduct property, VersionTime property
- You can get and set information about optional features (such as GIF or TIFF LZW file support or Express capabilities) that you are licensed to use.

  DLL functions: L_IsSupportLocked, L_UnlockSupport

  ActiveX methods: IsSupportLocked method, UnlockSupport method

  VBX properties: SupportType, IsSupportLocked, UnlockSupport

Express Editions

Many functions in the LEADTOOLS toolkits are faster in the express editions than in the professional editions, especially when compressing and decompressing CMP, JPEG, or CCITT images.

Some functions and features are available only in the express editions. They are labeled as [Express editions only] in the documentation.

If you redistribute an application created with an express edition, you must negotiate an individual contract (such as a royalty contract) with LEAD. Contact LEAD for more information.

LZW License Instructions

Warning

Use of this software for providing LZW capability for any purpose is not authorized unless user first enters into a license agreement with Unisys under U.S. Patent No. 4,558,302 and foreign counterparts. For information concerning licensing, please contact:

Unisys Corporation
Welch Licensing Department - C1SW19
Township Line & Union Meeting Roads
P.O. Box 500
Blue Bell, PA 19424
Phone: (215) 986-4411

Introduction To Image Processing with LEADTOOLS

The following topics provide general information about image processing, and where appropriate, they describe the related LEADTOOLS capabilities. These topics build up concepts and terminology; so if you are new to image processing, it is best to read them in order.

Introduction: Bits Per Pixel and Related Ideas

The terminology for image formats can be confusing because there are often several ways of describing the same format. This topic explains what the terms mean.

If an image is 24 bits per pixel, it is also called a 24-bit image, a true color image, or a 16M color image. Sixteen million is roughly the number of different colors that can be represented by 24 bits, where there are 8 bits for each of the red, green, and blue (RGB) values.

A 32-bit image is a specialized true-color format used in image files, where the extra byte carries information that is either converted or ignored when the file is loaded. The extra byte is used for an additional color plane in CMYK files, which are specialized files for color printing. In that case, LEADTOOLS, by default, converts the values to 24-bit RGB values when loading the image. The additional byte may also be used for an Alpha channel, which carries extra information such as a transparency indicator. Since the kind of information that an Alpha channel contains is outside the scope of normal image processing, some LEADTOOLS functions will fill the Alpha channel with zeros. Generally, functions such as reversing or resizing a bitmap will preserve the Alpha channel.

If an image is 16 bits per pixel, it is also called a 16-bit image, a high color image, or a 32K color image. Thirty-two thousand is roughly the number of different colors that can be represented by 16 bits, where there are 5 bits for each of the red, green, and blue values. (Devices that specify 64K color support are also referring to 16-bit images, but they are counting the left-over bit.)

If an image is 8 bits per pixel, it is also called an 8-bit image or a 256-color image. Two hundred fifty-six is the number of different colors that can be achieved by using the image data as 8-bit indexes to an array of colors called a palette.

If an image is 4 bits per pixel, it is also called a 4-bit image or a 16-color image. Sixteen is the number of different colors that can be achieved by using the image data as 4-bit indexes to a palette.

If an image is 1 bit per pixel, it is also called a 1-bit image, a black and white image, a 2-color image, or a bitonal image. Two is the number of different colors that can be achieved by using the image data as 1-bit indexes to a palette. The palette can contain colors other than black and white, although black and white are most common.

If an image is *grayscale*, its red, green, and blue values are all the same, and the values are incremented from the lowest to the highest. For example, an 8-bit grayscale image has 256 shades of gray, with values from 0 to 255. LEADTOOLS supports 4- and 8-bit grayscale.

Most of this same terminology is applied to video cards. For example, an 8-bit card is one that is capable of displaying 256 colors.

Introduction: Color Resolution and Dithering

Color resolution (also called color depth) refers to the number of possible colors in an image, as determined by the bits-per-pixel. If an image in your computer is loaded from a FAX scanner or received as a FAX transmission, it is a black-and-white (1-bit) image. If it is loaded from a color scanner or a JPEG file, it can have 16 million colors. If it is loaded from a GIF file, it is likely to have 256 colors. There are many possibilities, ranging from two colors to 16 million.

LEADTOOLS lets you manipulate an image and display it on any Windows-compatible device, regardless of the image's color resolution. Therefore, in most cases you can load, display, modify, and save an image without ever changing its color resolution. Nevertheless, in some cases you may need to increase or decrease the color resolution. Here are some examples:

- Suppose you need to load a large 24-bit image on a computer that does not have much memory. You can conserve memory by loading it as an 8-bit image. Then, if you were using an 8-bit display device, you would not see any difference in the quality of the image. Of course, there would be some loss of quality on a 24-bit display device, and the loss would be permanent if you then saved the image in the same file.
- Suppose you want to save an image in a different file format. If the new file format does not support the original color resolution, you can specify a different color resolution when you save the image.
- Suppose you want to combine one 8-bit image with another one. To combine two images, they must have the same color resolution, and if they are less than 16 bits per pixel, they must use the same palette. The simplest solution is to convert both to 24 bits per pixel before combining them. Then, if necessary, you can reduce the color resolution of the combined image.

Whenever you reduce an image's color resolution to 8 bits per pixel or less, a *dithering* method comes into play. One alternative is to use a nearest-color match (no dithering), which means that the color of each pixel is changed to the palette color that most closely matches it. If the original image contains subtle color details, the result of a nearest-color match may have large blotches of color that are not very pleasing.

Dithering methods create the appearance of more subtle shades by mixing in pixels of different colors. This is similar to the way newspaper pictures produce the appearance of shades of gray, even though the only actual colors are black and white.

Ordered dithering is the fastest method. It is the default dithering method when painting to a display device that is 256 colors or less. Ordered dithering takes advantage of the fact that the colors in most palettes are ordered so that similar shades are next to each other in the palette. This method avoids blotches of color by adding to or subtracting from the nearest-color value of each pixel to ensure that adjacent pixels do not have exactly the same color. (If the colors in the palette are not ordered, the results are not good.)

All of the other dithering methods in LEADTOOLS use *error-diffusion* algorithms. In error diffusion, the *error* is the difference between the original pixel color and the nearest match, and *diffusion* of this error is what the algorithm accomplishes. *Floyd-Steinberg* is a high-quality, fast error-diffusion method. It is the default method that LEADTOOLS uses if you reduce the color resolution when loading or saving an image, or if you specify error diffusion as the dithering method when painting to a display device that is 256 colors or less.

If you use a LEADTOOLS function to change the color resolution of a bitmap in memory, you can choose from a list of possible dithering methods. All of the alternative error-diffusion methods are slower than Floyd-Steinberg, and the quality may or may not be better, depending on the original image, and depending on whether the resulting image is to be displayed or printed. Choosing an alternative dithering method is a subjective, trial-and-error decision. *Stevenson and Arce* dithering, the slowest of the alternatives, is most likely to produce a higher quality result.

Introduction: Palette Handling

Any image that is 8 bits per pixel or less must have a palette. The palette is an array that contains the colors for displaying the image. The pixel values are indexes into the array. An image can have its own unique palette or it can share a common fixed palette. In either case, every bitmap in memory that is 8 bits per pixel or less has its own copy of a palette (even if it is a common fixed palette).

One way for a bitmap to have a unique palette is to load an image from a file that is 8 bits per pixel or less. In that case, the file contains a palette for the image, and you can assume that the palette is unique. By default, LEADTOOLS loads the palette that is stored in the file and associates it with the bitmap in memory.

Another way for a bitmap to get a unique palette is to create the palette. For example, if you reduce a 24-bit image to 8 bits per pixel, you can let LEADTOOLS create an optimized palette for the image.

By using a unique, optimized palette, you get the best possible image quality, but there is a drawback. When a display device uses a palette (for example, when it is in 256-color mode), it can only use one palette at a time to display colors. Therefore, if you are displaying more than one image at the same time, the current image may look perfect, while all of the others look bad, because they are mapped to the wrong palette. This is called *palette shift*.

Using a fixed palette solves these palette-shift problems. The best approach is to specify the fixed palette as a display option. Then, LEADTOOLS remaps each image to the fixed palette when it is displayed, without disturbing the individual image's palette. (When displaying 16-, 24-, or 32-bit images on a device with 256 colors or less, LEADTOOLS always uses the fixed palette.)

Alternatively, you can use a color resolution function to specify a fixed palette for each individual image. But then, when you save the image to a file, the fixed palette is saved in the file, and any unique palette information is lost.

Introduction: Image Manipulation and Analysis

Most LEADTOOLS functions for image manipulation and analysis act on the bitmap in memory. It is important to keep this in mind, because the changes become permanent when you save the bitmap in a file. Sometimes you want the changes to be permanent, and sometimes you do not. It all depends on the reason for making the changes.

Sometimes, you start with an image that has some undesirable qualities. For example, if the original image is too flat, you may want to use one of the contrast enhancement functions to make permanent improvements.

At other times, you may need to compensate for the display device. For example, an image may appear too dark on a particular monitor. In that case, setting the gamma correction is a good way to improve the brightness. But if you save the modified image, it might then appear too bright on another monitor. In this kind of situation, a good strategy is to modify a copy of the image, and give the user the ability to undo changes. The undo strategy also makes sense with a manipulation, such as a mosaic effect, that intentionally distorts an image.

Some image manipulations, such as edge-detection filters, are commonly used to analyze an image, rather than to improve it. For example, in an industrial quality-assurance application, defects in a machine part may be easier to detect if an image of the part is processed with an edge-detection filter. Similarly, the creation of a *histogram* (an array that charts the frequency of color usage in an image) can be of value in an analytical situation.

Edge detection and histogram creation can also be used as steps in improving an image. For example, creating a histogram is a necessary step, internally, in some of the contrast-improvement functions. Also, an image created with an edge detection filter can be combined with the original image to harden or soften the original's edges.

Introduction: Image Display

When you display an image, the following things come into play:

- Sizing and positioning the display.
- Adapting the image to the color resolution of the display device.
- Making other refinements that affect the display but not the bitmap in memory.

The LEADTOOLS *paint* functions let you specify which part of an image to display, how big to make the display, and where to position the origin (for scrolling and zooming). In the ActiveX (OCX) and VBX, the origin defaults to 0,0 and the size defaults to the bitmap size. This is arbitrary, and even with automatic scroll bars, it is unlikely to be what a user wants to see. Instead, the typical user either wants to see the whole image or wants to select a portion of the image to view.

Fortunately, LEADTOOLS makes it easy to show the user exactly what he wants to see. LEADTOOLS lets you control the size and position of the display without resizing or trimming the bitmap. In fact, it usually is best to think of the bitmap size as a measure of resolution, rather than a measure of size. The larger the bitmap, the more you can zoom in without losing detail.

As for adapting to the color resolution of the display device, if the color resolution of your device is greater than 256 colors, no adaptation is necessary; images are displayed using their own color values. Of course, if a 24-bit image is displayed on a 16-bit device, there is some loss, but there are still no programming issues, since the only remedy is to change the display mode.

If the color resolution of your display device is 256 colors or less, you have to make some programming decisions regarding dithering and palette selection. When displaying 16-, 24-, or 32-bit images, you can determine whether LEADTOOLS uses ordered dithering or error diffusion to handle the color reduction. (LEADTOOLS uses a fixed palette for these images.) When displaying 8-, 4-, or 1-bit images, you must decide whether to use a fixed palette or the individual image's palette. Using the fixed palette avoids palette shifting when more than one images is displayed at the same time. Using the image's original palette produces the best possible display of the image's colors.

Other LEADTOOLS functions let you determine how the new display interacts with whatever is already on the display device. For example, you can use Windows ROP codes to combine the displayed images. Also, when displaying 1-bit images, you can set scale-to-gray or favor-black options that can improve the readability of scaled images (images that are zoomed in or out to fit the available space).

Introduction: Bitmaps in Memory and in Files

The size of a bitmap in memory is determined strictly by its dimensions and its color resolution. For example, to calculate the number of bytes for a 500-by-500 pixel, 24-bit image, you multiply 500 * 500 * 3. (Multiply by 3 because there are 3 bytes in 24 bits.) For an 8-bit image, it would be 500 * 500, plus the size of its palette, which is about 1K.

For any color image, whether the color values are in a palette or in the image data, the color of each pixel in the bitmap consists of red, green, and blue values. They always use the RGB *color-space model,* which means that red, green, and blue values are combined to represent a color.

The size of the same color image when it is stored in a file is often much smaller, and the color-space model is not always RGB. For example, JPEG and LEAD CMP files both use a YUV color-space model, where Y is a *luminance* value, and the U and V values are *chrominance* values. In this color-space model, brightness information is stored in the luminance value, and color information is stored in the chrominance values. JPEG and LEAD CMP achieve data compression through sampling techniques that can affect the accuracy of the YUV values. This kind of compression is sometimes referred to as *lossy* compression, because you can manipulate the compression parameters to choose between greater compression or greater accuracy.

Another color-space model used in image files is CMYK (designed for color printing using cyan, magenta, yellow, and blackness). Other compression techniques include RLE (run-length encoding) and LZW (Lempel-Ziv and Welch). Both of these techniques preserve the accuracy of the image data, and they achieve compression solely by identifying repetitions of data. Some grayscale and 1-bit image file formats also use RLE and LZW compression techniques.

The image file formats are not only used for storage in the computer's file system, but are also used for database storage and for transmission of images. Furthermore, some applications keep frequently-used images in memory in a compressed file format. Thus, they can take advantage of memory speed when loading bitmaps from these files.

Introduction: DIBs, DDBs, and the Clipboard

A *DIB* (device independent bitmap) is a standard Windows representation of a bitmap. It is similar to the LEADTOOLS representation of a bitmap in that it includes palette information. A *DDB* (device dependent bitmap) is a hardware-specific representation that does not include palette information. Sometimes, you may need to manipulate a bitmap using Windows functions that work with DIBs or DDBs. For example, you may need to interact with an existing application that uses those functions. For that purpose, LEADTOOLS functions let you convert or copy to and from DIBs or DDBs. Of course, when converting to and from a DDB, you must preserve the palette information separately.

The c*lipboard* is a standard Windows object for copying data from one application to another. LEADTOOLS lets you copy images to the clipboard and paste images from the clipboard. When copying to the clipboard, most LEADTOOLS functions clear the existing data from the clipboard, and copy image data to it in the form of a DIB, a DDB, and a palette (if one is needed). If that is not what you want, you can use a LEADTOOLS DLL function to specify exactly which actions to take.

Introduction: Printing

LEADTOOLS printing functions are designed to work with Windows printing functions. Normally, you use Windows to select a *device context* (DC) and start the print job. LEADTOOLS functions let you scale and position one or more images in the printer DC. You use Windows functions to get the available page dimensions. You also use Windows functions to eject each page and to end the print job.

In most cases, you can scale and position an image on a printer DC with the same flexibility as when you display an image on a computer screen. You can let the printer handle any dithering or halftoning. However, if you halftone a bitmap, yourself, before sending it to a printer, you should resize the bitmap to the desired height and width, then use the halftone function, then send the image to the printer without scaling it. (You get bad results if you scale a halftoned image.)

In C, C++, Visual Basic, and Delphi, you can use the actual Windows printing functions, or their equivalents in the development system. However, printing with Microsoft Access and FoxPro is more limited, because they do not give you direct access to the printer DC. The LEADTOOLS ActiveX (OCX) handles this problem by providing methods to start and stop the print job. Because of limitations in the current interfaces for ActiveX controls, LEADTOOLS does not support inclusion of images in Access or FoxPro reports.

Introduction: TWAIN Input

TWAIN is a standard for image input devices, such as scanners and digital cameras. Getting an image from a TWAIN device can be very simple. First, there is a standard facility for selecting the input device. It is similar to selecting a default printer, and you can skip this step if you know that the default is right. Then, each TWAIN device has its own user interface, which you invoke to *acquire* an image.

Some complexity is introduced only when acquiring multiple images or when bypassing the user interface. If you acquire multiple images, you must code a callback function or event procedure to handle each image. If you bypass the user interface, you must set some standard TWAIN parameters to provide the information that the TWAIN driver would normally get from the user interface. The tricky part is that some TWAIN drivers implement the standard more fully than others. If you do not know in advance how the TWAIN device will behave, it is a good idea to add logic to your program to test the success of setting the TWAIN parameters. LEADTOOLS provides functions for all of this.

Introduction: Database Interaction

With LEADTOOLS, you can use a variety of strategies to maintain images in a database. This variety is necessary because of the uneven support for storing images using different development and database systems.

The simplest database support is with Visual Basic, where you can bind a LEAD control (ActiveX or VBX) to a data control. You can associate the LEAD control with a long binary field in the database, and let the data control handle all of the database maintenance and navigation.

For other systems with ActiveX support, you may be able to use ODBC (Open Database Connectivity). ODBC is a standard interface for accessing local and remote databases. To use it, you define a data source that is implemented with an ODBC driver; then you access the data source in the same way you would a local database. LEADTOOLS provides ActiveX methods and properties that let you directly access image data through an ODBC data source. There are a couple of limitations in this approach. First, since the ActiveX accesses the image data directly and does not access any other data in the record, you must deal with the problem of identifying the image. To solve this problem, you can use another means of selecting records; then synchronize the ActiveX's view, so that it is always accessing the same record. (The ActiveX tutorials describe how to do this.) The second limitation is with ODBC drivers. Most of the available ODBC drivers do not properly support access to long binary fields. (The ACCESS 7.0 and FoxPro 2.x drivers have been tested successfully and are described in the ActiveX tutorials.)

A more flexible approach on lower-level development systems, such as C, C++, and Delphi, is to use the ability of LEADTOOLS to save a file in memory. Once the image is available in memory in a compressed file format, you can use your choice of database engines to store and retrieve the data.

Finally, you may want to consider using your computer's or network's file system to store the images, use a database to store the file paths as strings, and write your own code to do the synchronization. In some cases, this could be a better solution than storing the images directly in the database.

Introduction: Data Transfer

When receiving transferred data, as in an Internet application or a FAX modem transmission, you often want to display the image as it is received.

To meet that need, LEADTOOLS provides simple functions that let you load image data into a bitmap as it is received. You can also use the LEADTOOLS paint-while-load feature to display the bitmap as it is received.

Summary of All Supported Image File Formats

Format	Read Bits per pixel	Write Bits per pixel
LEAD. This is the LEAD CMP compressed format for grayscale and color images. This format results in smaller files and better image quality than industry-standard formats.	8 for grayscale, 24 for color	8 for grayscale, 24 for color
JFIF. This is the JPEG File Interchange Format. LEADTOOLS supports YUV 4:4:4, 4:2:2, and 4:1:1 color spacing, and YUV 4:0:0 for grayscale.	8 for grayscale, 24 for color	8 for grayscale, 24 for color
Progressive JPEG. This is a JFIF format that is useful for transmitting images, because the first part of the file contains the full dimensions of the image. Therefore, in a paint-while-load routine, you can display the whole image, then progressively clarify it as the rest of the file loads. LEADTOOLS supports YUV 4:4:4, 4:2:2, and 4:1:1 color spacing, and YUV 4:0:0 for grayscale.	8 for grayscale, 24 for color	Read only
JTIF. This is the JPEG Tagged Interchange Format. LEADTOOLS supports YUV 4:4:4, 4:2:2, and 4:1:1 color spacing, and YUV 4:0:0 for grayscale.	8 for grayscale, 24 for color	8 for grayscale, 24 for color
TIFF. This is a tag-based file format designed to promote universal interchanges of digital image data. Because TIFF files do not have a single way to store image data, there are many versions of TIFF. LEADTOOLS supports the most common TIFF formats.	1, 4, 8, 16, 24, 32 also including CMYK (LZW) and RLE (LZW)	1, 4, 8, 16, 24
MPT. This is a multipage TIFF format that enables a file to contain more than one image. It is handled the same as a regular TIFF file, except for the multipage feature.	1, 4, 8, 16, 24, 32 also including CMYK (LZW) and RLE (LZW)	1, 4, 8, 16, 24
TIFF LZW. These files use the Tagged Image File Format with LZW compression. Refer to LZW License Instructions.	1, 4, 8, 16, 24, 32	1, 8, 16, 24
TIFF CCITT. These are compressed TIFF files that are commonly used for FAX transmission and document imaging.	1	1
TIFF CCITT Group 3. These are TIFF CCITT files in a format that is more advanced and more compressed than TIFF CCITT. LEADTOOLS supports both 1-dimension and 2-dimension variations of this format.	1	1
TIFF CCITT Group 4. These are TIFF CCITT files in a format that is more advanced and more compressed than TIFF CCITT Group 3.	1	1
IOCA (ICA). This is the Image Object Content	1	1

Architecture developed by IBM. LEADTOOLS supports these files in an MO:DCA wrapper with embedded 1-bit CCITT Group 3 or Group 4 images. LEADTOOLS also supports IOCA files without an MO:DCA wrapper.		
WinFax Group 3. This is a FAX format created by Delrina for Group 3 support.	1	1
WinFax Group 4. This is a FAX format created by Delrina for Group 4 support.	1	1
FAX Group 3. This is a raw FAX format (without a header) for Group 3 support. LEADTOOLS supports both 1-dimension and 2-dimension variations of this format.	1	1
FAX Group 4. This is a raw FAX format (without a header) for Group 4 support.	1	1
Truevision TGA (TARGA). This is a file format created by Truevision Inc. LEADTOOLS supports all uncompressed and RLE compressed TGA file formats.	8, 16, 24, 32	8, 16, 24, 32
GIF. This is the Graphics Interchange Format created by CompuServe for storing and exchanging color raster images. This format compresses its image data with the LZW compression technique. Refer to LZW License Instructions.	1, 4, 8	8
PNG (Portable Network Graphics). This is a replacement for the GIF format. It is a full-featured (non-LZW) compressed format intended for widespread use without legal restraints.	1, 4, 8, 16, 24, 32	1, 4, 8, 24
Photoshop 3.0 (PSD). This is the format produced by the Adobe Photoshop graphics editor.	1, 8, 24	1, 8, 24
Windows Bitmap (BMP). This is a file format created by Microsoft. Some BMP images are compressed with an RLE type compression.	1, 4, 8, 16, 24, 32 and 1, 4, 8 RLE	1, 4, 8, 16, 24, 32
Windows Metafile (WMF). These files are not bitmap based images. A Windows metafile consists of a collection of device independent functions that represents an image. When a program loads a metafile, these functions are executed to obtain the image.	8, 24	8, 24
PCX. This is a file format created by ZSoft. This format compresses its image data with the RLE type compression.	1, 4, 8, 24	1, 4, 8, 24
DCX. This is a multipage PCX format that enables a file to contain more than one image. It is handled the same as a regular PCX file, except for the multipage feature.	1, 4, 8, 24	1, 4, 8, 24
PostScript Raster (Encapsulated PostScript).	1, 8 (raster)	8 grayscale (raster)

These files are used primarily on PostScript printers. These printers usually offer more variety of fonts and higher resolution than standard laser printers. EPS files will work on any PostScript compatible printer and any end user application that supports placement of EPS files in its work space.	and 1, 4, 8, 16, 24, 32 (embedded TIFF)	
OS/2 Bitmap (OS/2 BMP). These are files created on an OS/2 operating system. LEADTOOLS supports both 1.x and 2.x formats.	1, 4, 8, 24 and 1, 4, 8 RLE	1, 4, 8, 24
CALS Raster. These are 1-bit CCITT Group 4 CALS raster files. CALS is a United States government standard.	1	1
MacPaint (MAC). These Macintosh Paint files are commonly used for monochrome clip art.	1	1
GEM Image (IMG). These files are native to the Graphical Environment Manager developed by Digital Research.	1	1
Microsoft Paint (MSP). These files from early versions of Windows are used for black-and-white drawings and clip art.	1	1
WordPerfect (WPG). These are WordPerfect raster files.	1, 4, 8 (raster)	1, 4, 8 (raster)
SUN Raster (RAS). These files are native to Sun UNIX platforms.	1, 4, 8, 24, 32	1, 4, 8, 24, 32
Macintosh Pict (PCT). These files, produced using Macintosh QuickDraw, are used in desktop publishing and imaging applications.	1, 4, 8, 24	1, 4, 8, 24
LEAD 1BIT. These are 1-bit LEAD compressed files.	1	1
PCD. These are Kodak PhotoCD files.	All	Read only

Copyright Notice

Trademarks

LEAD Technologies, Inc.
900 Baxter Street
Charlotte, NC 28204

(704) 332-5532
FAX: (704) 372-8161
BBS: (704) 334-9045
Tech Support: (704) 372-9681
CompuServe: 71333,2237 or GO LEADTECH
Home page: http://www.leadtools.com

Summary of LEADTOOLS Products

LEAD Technologies, Inc.
900 Baxter Street
Charlotte, NC 28204

Sales@Leadtools.com
www.leadtools.com

(704) 332-5532
FAX: (704) 372-8161
BBS: (704) 334-9045
Tech Support: (704) 372-9681

CompuServe: 71333,2237 or GO LEADTECH

LEADTOOLS VBX Pro: Visual Basic custom control for adding LEADTOOLS capabilities to 16-bit development systems such as Visual Basic 3.0, Visual Basic 4.0 (16-bit), and Borland Delphi (16-bit).

LEADTOOLS ActiveX16 (OCX) Pro: OLE custom control for adding LEADTOOLS capabilities in a Windows 3.x environment. You can use this toolkit with Microsoft Access 2.0, Visual Basic 4.0 (16-bit), and other 16-bit development systems that support OLE custom controls.

LEADTOOLS ActiveX32 (OCX) Pro: OLE custom control for adding LEADTOOLS capabilities in a WIN32 environment such as NT, Windows 95, or Win32s. You can use this toolkit with 32-bit development systems such as Visual Basic 4.0 (32-bit), Microsoft Access 95, Visual FoxPro, and Borland Delphi (32-bit).

LEADTOOLS ActiveX16/32 (OCX) Pro: Two OLE custom controls, combining the features of the ActiveX16 and ActiveX32 products. The LEADTOOLS syntax is the same for both 16- and 32-bit environments. Therefore, if you are using a development system such as Visual Basic 4.0, you can easily write code that compiles in both environments.

LEADTOOLS WIN16 Pro: API for the LEADTOOLS WIN16 DLLs, plus LEADTOOLS VBX Pro and ActiveX16 Pro (which are included to give you the full range of WIN16 development options). The ActiveX16 product is upgraded to include Visual C++ examples and tutorials.

The API for the LEADTOOLS DLLs provides a complete set of compression and manipulation functions. You can use the API with any development system that supports DLLs in the Windows 3.x environment. For example: Microsoft C and Borland C.

LEADTOOLS WIN32 Pro: API for the LEADTOOLS WIN32 DLLs, plus LEADTOOLS ActiveX32 Pro (which is included to give you the full range of WIN32 development options). The ActiveX32 product is upgraded to include Visual C++ examples and tutorials.

The API for the LEADTOOLS DLLs provides a complete set of compression and manipulation functions. You can use the API with any development system that supports DLLs in the WIN32 environment. For example: Microsoft C and Borland C.

LEADTOOLS WIN16/WIN32 Pro: APIs for the LEADTOOLS WIN16 and WIN32 DLLs; plus LEADTOOLS VBX Pro, ActiveX16 Pro, and ActiveX32 Pro. In effect, this product combines LEADTOOLS WIN16 Pro and LEADTOOLS WIN32 Pro.

LEADTOOLS Pro Express: All the features of LEADTOOLS WIN16/WIN32 Pro; plus the additional features and faster performance of the Express editions.

LEADTOOLS OS/2 Pro: API for the LEADTOOLS OS/2 DLLs. The API provides a complete set of compression and manipulation functions. You can use the API with any development system that supports DLLs for OS/2 version 2.1 or later (including OS/2 Warp). For example: IBM C Set++ and Borland C++ for OS/2.

LEADTOOLS OS/2 Pro Express: All the features of LEADTOOLS OS/2 Pro; plus the additional features and faster performance of the Express editions.

For more products, refer to Other Products from LEAD Technologies.

Call for details on custom applications, operating system ports and licensing fees.

All products are sold in the U.S. and Canada with a 30-day money back guarantee.