w3 Image 2.0

User Manual

Contents

Copyright

We have done our best to ensure that the information in this manual is both accurate and useful; however, please be aware that errors may exist, and that Dimac Development / Duplo AB (hereafter referred to as Duplo) does not give any guarantees concerning the accuracy of the information here or in the use to which it may be put.

Duplo may have patents and/or pending patent applications covering subject matter in this document.

Dimac Development
Duplo AB
Prästgatan 12

252 24 Helsingborg
Sweden

Phone: +46 42 35 94 00
Fax: +46 42 15 80 50
Homepage: http://www.dimac.net
E-mail: info@dimac.net
Support: support@dimac.net

If any problems occur using this application, please visit our web site http://www.dimac.net.

Introduction

Welcome to w3 Image

The main purpose of w3 Image is to generate and/or manipulate images.
w3 Image is suitable for:

- *Creating diagrams displaying data from your database*
- *Creating images for headings and buttons*
- *Automatic generation of thumbnails*
- *Creating signed images*
- *Color converting*
- *Image analysis*
- *Transforming images*
- *Converting between different image formats*
- *Performing the most common drawing operations*

w3 Image is an ASP compatible COM object easily integrated in your ASP solutions. w3 Image can also be used in other environments such as services or Windows applications. As a COM based solution, it is possible to use the component in several programming languages such as C++, Visual Basic, and other COM supported languages.

The object-oriented structure in w3 Image makes the model straightforward to use. The design has been developed to be as simple as possible to use and also to insure that the objects can be reused within the system. All objects within the w3 Image system are persistable, that is they can be directly loaded or saved to a stream using the IPersistStream interface methods. This is useful when restoring the state of an object after an undetermined time. For example, imagine a tool in which the user could continue the drawings of an image as it was before, with the selected pen, brush, font, and the data.

The system has been optimised to stream the data to the client machine by best performance. This is done by direct accessing the Response object and the write mechanisms to the client machines. Another advantage by using this method, is that the system is now capable to define the image data type before sending it to the client (content type).

The images created with w3 Image are generated on a server and shown at the client as a picture. Therefore, the images can be displayed on all client platforms and browsers (*provided that the image format is supported by the browser*).

w3 Image offers several ways of defining colors. The management has been adapted to suit the web environment, as the colors can be described in a hexadecimal form or in a web color form. ¨

w3 Image can create images with numerous palette algorithms. Depending on the demands of the performance or the importance of the color information in the output data, different algorithms can be selected.

Who should read this manual?

This manual is intended for the users of Dimac's w3 Image. The users should have knowledge in basic programming, JScript, and/or VB Script.

How this manual is organized

Chapter 1: Objects Chapter 1 introduces you to the objects used in w3 Image: images, pens, brushes, fonts, colors, and their different properties.

Chapter 2: Operations Chapter 2 lists all operations you can perform with the different objects. For example, draw a line with a pen, fill a circle with a color, or enter a text. Chapter 2 also describes the image palettes used in w3 Image.

Chapter 3: Image Formats Chapter 3 informs you about the image formats supported by w3 Image.

Chapter 4: Transparency Chapter 4 tells you about the behaviour of transparency in w3 Image.

What's new?

2.0

- w3Image is now fully capable of loading and streaming any of the supported image formats. This by using the LoadImageFromStream and the StreamImage method.

- Support of the WBMP format in all import and export methods. WBMP files is very useful when creating mobile applications and scripts.

- Support of loading images from an url location.

- Transformation methods are added such as GreyScale, Threshold and invertion of images using the Negative method.

- w3Image is extended with lighting operations such as Brightness, Contrast and manipulation of individual intensity by using the ShiftRGB method.

- Version property is added to receive the version string of the dll file.

- The RotateCenter is an extended version of the rotate method. The method rotates the image at the center of the picture and resizes the image if neccessary. The method is very fast and using different types of interpolation methods.

- w3Image has rebuilt the import functions to inherit the transparency of the loaded image. **Note: This may affect your earlier code.** If you've previously loaded jpegs and applied text or other items to them transparently, you may need to set image.bkmode=1 manually after loading them!

- The export of 32 bit PNG images is changed to use the alpha channel in a more proper way.

1.3
- When using the method *StretchBltExt* you can now quickly perform the combination of alpha and transparent blit, since we have added the operation: *alphatransparent*
The steps of the operations are cut down to one single step!

- Get low level access to the memory (useful e.g. when building your own filters) with:
Lock Memory
UnlockMemory

- List all available fonts on the host machine with:
GetFontFamiliesName, getFontFamiliesCount
1.1
- Support for NT4 and IIS4.

- Four new image formats are supported by w3 Image: GIF, TIFF, TGA, and ICO. Please see Chapter 3 for further information.

- Support for transparency when generating GIF and PNG images. Please see Chapter 4 for further information.

- Extended algorithms for transformation operations (Stretch, Scale, and Rotate). Please see the Reference Manual for further information.

Installing w3 Image

To use Dimac's w3 Image you must have the *w3image.dll* registered on your web server. This is done by either running the installation program or copying the *w3image.dll* file to your web server and manually registering it with the command: *regsvr32 w3image.dll*. Any previous installations of w3 Image must first be removed, as described below.

Before registering the *w3image.dll* file, the following files must also be copied or installed in the system32 directory:

cpuinf32.dll
ipl.dll
ipla6.dll
iplm5.dll

iplm6.dll
iplp6.dll
iplpx.dll
iplw7.dll
ijl15.dll
msvcrt.dll

To run the installation program you must have administration permission on the web server. Double click the *w3image.exe* file to run the installation program. The installation will by default install at c:\Program Files\Dimac Development\w3 Image\ together with manuals, references, and some ASP example files. The installation will register the *w3image.dll* file as a COM object.

Execute the ASP Example Files

To better comprehend w3 Image, a good idea is to run the ASP example files that come with the installation. To be able to see the result of the example files, they must be executed on a web server. For further information about their behaviour, please see the comments in the examples.

Uninstalling w3 Image

Uninstall is available in the **Add/Remove Programs** in your setting folders. You can also do this manually by unregistering the *w3image.dll* component (type: *regsvr32 /u w3image.dll*) and then deleting the file.

Chapter 1: Objects

The Object Model in w3 Image

w3 Image consists of a set of COM objects. The most important of these objects is the **image** object. This object holds the image data, that is needed to perform drawing operations. The image data is like a drawing surface on which the user performs the drawing operations. These operations are accomplished by using the other objects in w3 Image: **pens**, **brushes**, **fonts**, and **colors**.

Pens are used when managing simple image routines such as drawing lines, rectangles, and circles. **Brushes** are used to fill objects with color, while **fonts** are used to draw texts in the image. w3 Image uses **color** objects in order to specify colors and simplify conversions between different color formats.

The illustration below shows an example of three image objects using different pens, brushes and fonts.

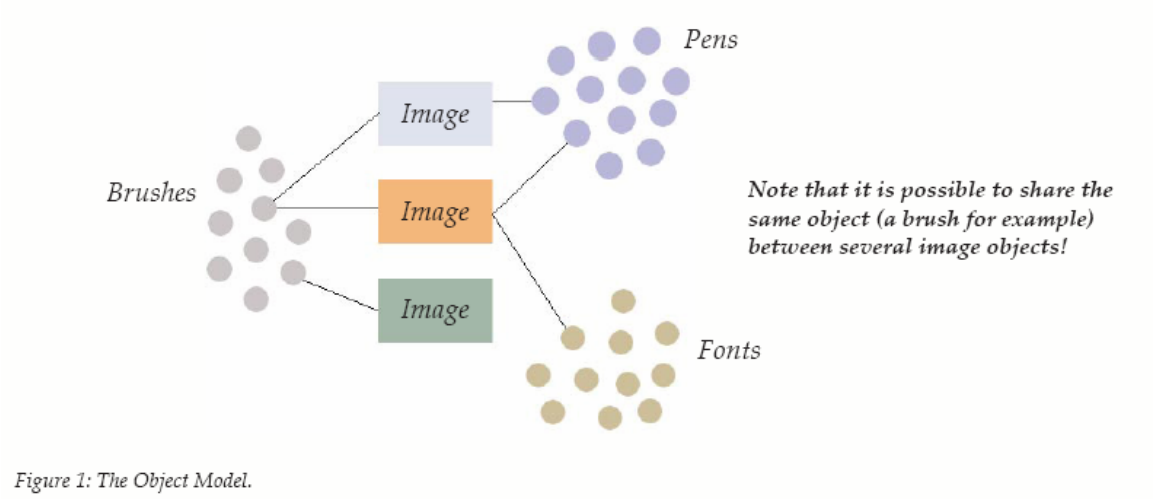


Figure 1: The Object Model.

Working with the Image Object

The image object is the main object in w3 Image. It contains the actual image data, creates all other objects (pens, brushes, fonts, and colors), and handles all drawing operations. The image object can use one object per type at the same time. That is, a pen, a brush, a font, and a color can be selected at the same time. But a certain pen (brush, font, or color) is used until a new pen (brush, font, or color) is selected.

To create an empty surface

Before you can work with an image you have to create an instance of the image object. To perform any later operation it is necessary that the image object has allocated image memory. This is done by either creating an empty surface or by loading an image into the image object.

```
'Create an instance of the image object
Set imageobj = Server.CreateObject( "w3Image.Image" )

'Allocate memory for a 200 * 400 pixels image
imageobj.CreateEmptySurface 200, 400
```

To load an image

The example shows how the image object loads the picture into the memory and allocates the necessary memory. The width and the height of the image object will be exactly the properties of the loaded image.

```
'Load an image
imageobj.LoadImage "c:\test.bmp"
```

To set the background properties

To affect the background color and mode when creating an empty surface, it is important that the mentioned properties are set before the operation of creating an empty surface. This is done simply by...

```
'Set mode to opaque
```

```
imageobj.bkMode = 0

'Set mode to transparent
imageobj.bkMode = 1

'Set background color to blue
imageobj.bkColor = &H000000FF&
```

Note that &H00112233& specifies a hexadecimal value in VB Script and that 0x445566 specifies a hexadecimal value in Java Script!

There are different ways to indicate the value in VB Script, but we recommend the above mentioned example.

Working with Pens

Pens are used to perform simple drawing operations, such as **DrawLine** and **DrawRect** (see section *Simple Drawing* in *Chapter 2: Operations*).

To create and select a pen object

When creating a pen object it is possible to define three types of attributes: style, width, and color.

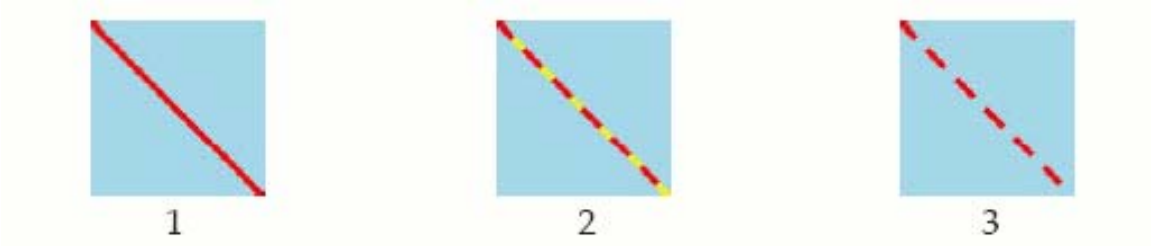
```
'Create a red solid pen with the width of two pixels
Set penobj = imageobj.CreatePen ( "solid", 2, &H00FF0000&)

'Select the created pen
imageobj.SetPen penobj

'Perform the drawing operation with the selected pen
imageobj.DrawLine 10, 10, 100, 100
```

Styles of pens

There are solid and none solid pens. A none solid pen is a dashed and/or dotted line. The width of a non-solid pen is 1 pixel. A pen with a width size larger than 1 pixel can not be non-solid. The width of a solid pen is 1 pixel at the minimum.

When a non-solid drawing operation is used, it is important to define the right background properties in order define the gap color and the transparency.

The three illustrations show a red pen on a blue image. The background color is set to yellow.



Number 1 shows the drawing of a solid pen.

Number 2 and 3 show a non-solid pen. In number 2 the background mode is set to opaque. In number 3 the background mode is set to transparent.

Working with Brushes

Brushes are generally used to paint the interior of polygons and ellipses. w3 Image supports solid brushes which paints with a specified solid color.

To create and select a brush object

The example shows how to create and select a solid brush and perform a drawing operation.

```
'Create a brush object
Set brushobj = imageobj.CreateSolidBrush( &H00A3F403& )

'Select the brush
imageobj.SetBrush brushobj

'Perform the drawing operation with the selected brush
imageobj.FloodFill 10, 10, &H00ABCDEF&
```

Working with Fonts

The font object is used to describe and/or to receive information about a specific font. The most important properties of a font are the typeface, the style, and the size. To create and use a font object, the font needs to be installed on the computer/server.

To create and select a font object

With the following example you create a font with the typeface Tahoma, using a normal style and a height of 48.

```
'Create a font object
Set fontobj = imageobj.CreateFont( "Tahoma", 48,0, "normal",0,
&H0000FF00&,false,false,false )

'Select the font
imageobj.SetFont fontobj

'Perform the drawing operation with the selected font
imageobj.DrawText "Dimac", 15, 20
```

Working with Colors

The benefit by using the color object compared to normal color definition (hexadecimal form), are the powerful color options supplied with the color object. When creating a color object there are three different ways to describe a color value:

· *RGB (as number)*
· *Web Color (RGB as string)*
· *Web Color Name (Web-safe colors)*

RGB (as number)

RGB stands for Red, Green, and Blue. Each color of the RGB component is described as an intensity value, ranging from 0 to 255. When the values of all three components are equal, the result is a shade of grey.

To create a color object using RGB

The example describes a blue color with a red value of 2, a green value of 10, and a blue value of 245.

```
Set colorobj = imageobj.CreateColorRGB( 2,10,245 )
```

To receive the intensity color of each RGB value

Receive the value for each intensity.

```
'Receive the value for each intensity color
red = colorobj.red 'Red = 2
green = colorobj.green 'Green = 10
blue = colorobj.blue 'Blue = 245
```

To change an individual RGB value

Change one of the individual RGB values.

```
'Red is changed to 10
colorobj.red = 10
```

Web Color (RGB as string)

The most common way in scripting is to define each of the intensity values in a hexadecimal form: #RRGGBB.

The color definition describes each intensity value as a hexadecimal value from 00 to FF, starting with the red value followed by the green and finally the blue. It is important that the RGB string starts with the '#' sign to separate it from the string using a web color name.

To create a color object using a web color value

Create a color object with the RGB value of 2, 10, and 245 by describing each value as a hexadecimal number.

```
Set colorobj = imageobj.CreateColor( "#020AF5" )
```

To receive a web color

Receive the web color used by the color object.

```
'Receive the web color value
webcolor = colorobj.webcolor

Response.Write webcolor
'An example of the output could be "#34A510"
```

To change a web color

Change the web color.

```
'Red is changed to the RGB value of 204, 255 and 187
colorobj.webcolor = "#CCFFBB"
```

Web Color Name (Web-safe colors)

Another method to describe a color value is by defining it as a color text. Then the user doesn't have to know the actual RGB value of a color definition. As there are over 16 millions different combinations of a color value, it could be quite difficult to remember the exact color value. Therefore, the most common used web colors are collected in color set, where each value is described as a color text. The color set in w3 Image contains 139 different colors.

To create a color object using a web color name

Create a color object using a web color name and receive its blue intensity value.

```
Set colorobj = imageobj.CreateColor( "antiquewhite" )
blue = colorobj.blue
```

Chapter 2: Operations

Simple Drawing

DrawLine Draw a line between two points with a chosen pen.

DrawEllipse	Draw an ellipse with a chosen pen and fill it with a chosen brush.
DrawRect	Draw a rectangle with a chosen pen and fill it with a chosen brush.
FloodFill	Fill a surface with a chosen brush.

The co-ordinate system in w3 Image
The following illustration shows the co-ordinate system in w3 Image with a filled rectangle.

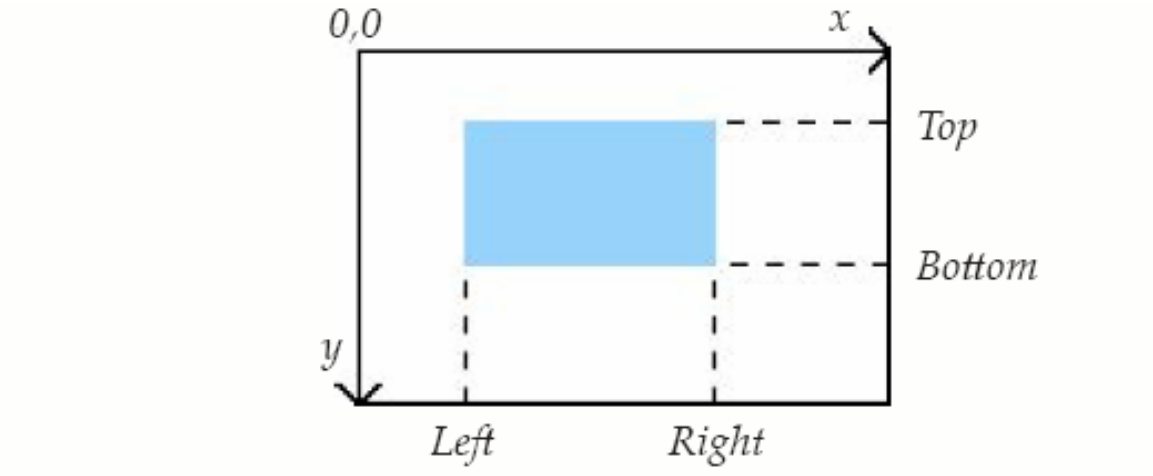


Figure 2: A blue rectangle positioned in the co-ordinate system.

Pixel

SetPixel	Set a chosen color for a separate pixel on a specified position.
GetPixel	Get the value of a pixel on a specified position.

Text

GetTextHeight	Get the height of a text.
GetTextWidth	Get the width of a text.

The height and width of a text
The text height is the height of the selected font. The text width is the width of the surface the text occupies.

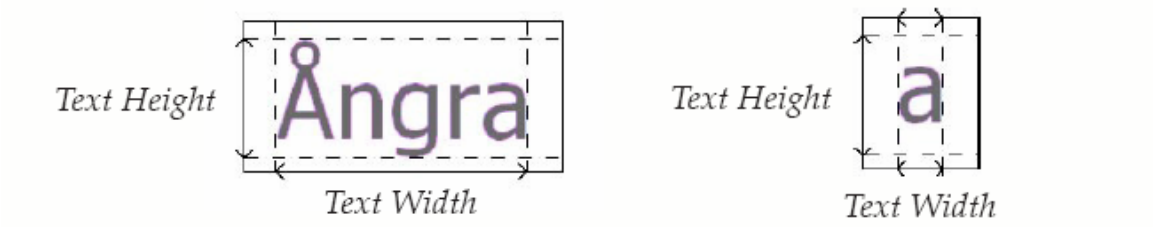


Figure 3 and 4: The illustrations show the text height and width of two words written in the same font.

DrawText Render a text with a chosen font.

Center a text
1. In order to perform any font operation, you have to create an empty surface (1x1 pixel is enough).
2. Then create and select the font you want to use.
3. Define the text to be centered.
4. Receive the width and the height of the text.
5. Load the image in which you want to center the text.
6. Select the font again, since creating a new surface removes previous set objects. (When loading an image a new surface is created.)
7. Calculate the position for the font (top and left).
8. Perform the drawing of the text.
9. Stream the image to the client, that is display the result on the computer screen.

```
1. imageobj.CreateEmptySurface 1,1

2. Set fontobj = imageobj.CreateFont("Tahoma",60,0,"normal",0,
&H00000000&,false,false,true)
imageobj.SetFont fontobj

3. text = "w3 Image is the answer..."

4. width = imageobj.GetTextWidth(text)
   height = imageobj.GetTextHeight(text)

5. imageobj.LoadImage "C:\Projects\www\w3Image\
Demo\images\dimac.jpg"

'Important! Reselect the font (all data is erased when
creating a new surface)

6. imageobj.SetFont fontobj

7. top = ((imageobj.height - height)/2)
 If (top < 0) then
   top = 0
 End If
   left = ((imageobj.width - width)/2)
 If (left <= 0) then
  left = 0
 End If

8. imageobj.DrawText text, left, top

9. imageobj.StreamImage Response, "JPG", 24
```

Kerning

GetFirstKernChar	Get the first kerning charachter from a pair at a specified index.
GetSecondKernChar	Get the second kerning charachter from a pair at a specified index.
GetKernAmount	Get the kerning value from a pair of kerning charachters.

KerningPairsCount Get the number of kerning charachter pairs from a font object.

Blit and Copy

StretchBlt Copy a part of a surface to another part within the same image object.

StretchBltExt Copy a part of a surface to another part within another image object.

An example of the operation alphablending
The illustrations show three images where a part of the second is blended with the first into a third image. The colors blend into each other, so the surface where the blue and the red colors overlap turns to violet. Note that the second image has been stretched and the proportions have been changed.

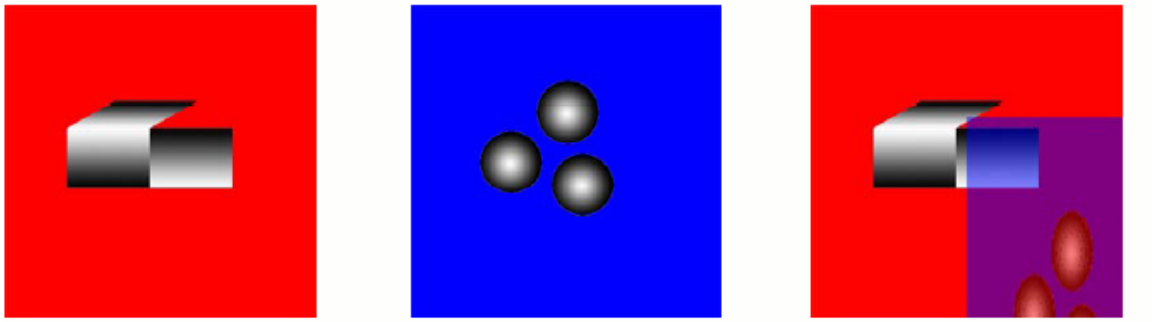


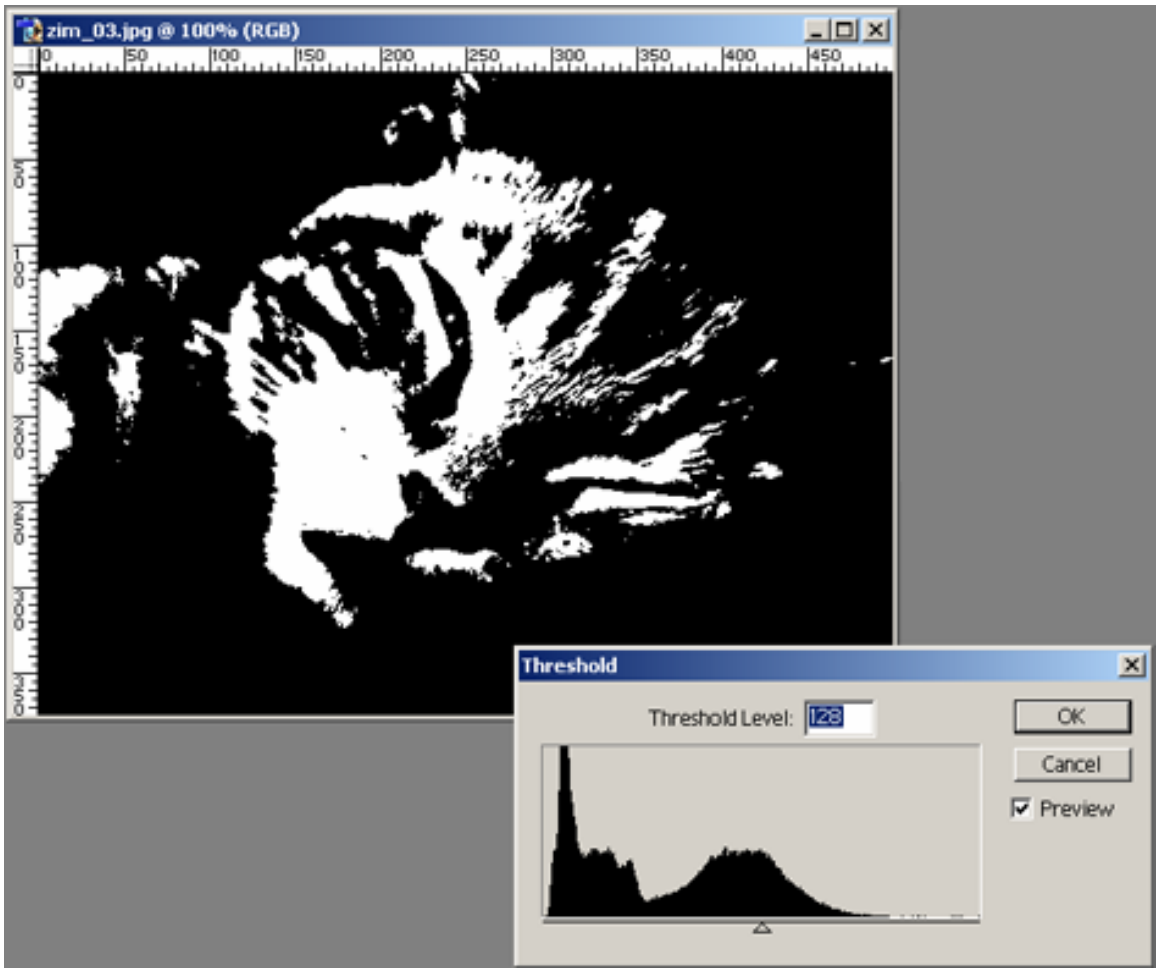
Figure 5-7: You can blend images or part of images into each other.

Clone Clone an image object.

Transformation

To change the dimensions of the images, it must be possible for the image object to manage a set of transformation operations.

Stretch Specify a new size on the surface and adapt the content to the new size.
Scale Scale an image. The image keeps its proportions.
Rotate Rotate an image.
RotateCenter Rotates the image around its center point.
Crop Crop an image.
Flip Turn the image vertically and/or horisontally.
GrayScale Converts the image to grayscale.
Negative Inverts the image.
Threshold Converts the image into black and white, specified by the threshold lightness value. See figure below.
Brightness Adjusts image brightness.
Contrast Adjusts image contrast
ShiftRGB Shift the images R, G or B levels.



Setting the Threshold level 128 makes all pixels whose (R+G+B)/3 are less than 128 turn black, all else turn white.

Image Information

Height The height of the image object's surface in pixels.
Width The width of the image object's surface in pixels.

Import

LoadImage Load an image from a disk and create a surface for the image. Please turn to *Chapter 3: Image Formats* for information on which image formats w3 Image supports.

LoadImageFromStream
Loads an image from a IStream object (for example adodb.stream). Loading from ADODB.Stream requires MDAC2.6 or greater.

LoadImageFromURL
Loads image from an URL.
f.x. `LoadImageFromURL("http://www.dimac.net/logo.jpg")`

Export

SaveImage Save the surface of an image to a file.
StreamImage Stream the surface of an image to an object that supports IStream or IResponse.

Examples of streaming

Streaming image data to Response-object:
When streaming image data to the Response object you can not do any response messages in the script generating the image. The following example tries to return a message via the Response object in the image.asp file. This can not be done, since the text message will interfere with the data generated by w3 Image.

File: index.html

```
<html>
<head>
</head>
<body>
<img name="w3image" id="w3image" src="image.asp">
</body>
</html>
```

File: image.asp

```
<%@ LANGUAGE = VBSCRIPT %>
<%
      Set imageobj = Server.CreateObject("w3image.image")
      img.CretateEmptySurface 100,100
      Response.Write "Test" 'Not allowed!
      imageobj.StreamImage Response, "BMP", 24
%>
```

You can solve this by temporarily not streaming the image, and instead saving the image directly to disk (SaveImage). Naturally, you must then run the image.asp file directly, since there is no streaming. For further information about error handling, please see nr 2-4 of the ASP example files that come with the installation of w3 Image. These files describe how to generate images displaying the error messages.
If you need to contact our support, please do not send these generated images. Instead, post a request at our helpdesk and send the error message with your request.

Streaming to an ADODB.Stream

This code shows an example of streaming the image to an ADODB.Stream.

```
<%@ LANGUAGE = VBSCRIPT %>
<%
      set oImage = Server.CreateObject("W3Image.Image")
      oImage.LoadImage(Server.Mappath("yay_train.jpg"))

            oImage.Flip(2) // flip image, so we see something happens
                           // to it.
            Set objStream = Server.CreateObject("ADODB.Stream")
```

```
            objStream.Type = 1  // read only
            objStream.Open // open stream
            oImage.StreamImage objStream, "GIF", 8, "grayscale"
            //          write to stream
            Set oImage = Nothing // kill image object
            objStream.position = 0 // rewind stream
            // write content to browser
            response.binarywrite objStream.Read()
            // close and dispose of stream object.
            objStream.Close
            Set objStream = Nothing

%>
```

Why should one want to do the above, when you can stream to the Response-object, you ask? ADODB-streams are more flexible, and the above is just an example. Instead of writing the binary stream to the response-object, you could put it into a recordset-field instead (`objRS("MyBlobField").appendChunk objStream.Read() )`

Support of image formats

w3 image supports the functionality of streaming the image data of the following formats:

* *BMP (Bitmap)*
* *JPG (Joint Photographic Experts Group)*
* *PNG (Portable Network Graphic).*
* *GIF (Graphics Interchange Format)*
* *TIFF (Tagged Image File Format)*
* *TGA or TARGA (Truevision Advanced Raster Graphics Adapter)*
* *ICO (Windows Icon)*
* *WBMP (Wireless Bitmap)*

Note that some web browsers can not display all image formats correctly.

Image Palettes

When using images with color depth less or equal to 8 bit/pixel (or palette entries less or equal to 255) a palette is needed. w3 Image uses the palette functions when working with BMP, PNG, GIF, TIFF, TGA and ICO files, but not JPG files.

w3 Image supports four different palette algorithms or functions:

· *First Found*
· *Greyscale*
· *Rainbow*
· *Nearest Color*

Note that the First Found algorithm is used as default when none is specified!

First Found Color Palette (default)

This palette function starts to search from the bottom of an image for the first found color values. If a color value in the image doesn't exist in the palette, it is automatically added to the palette. The function continues to search the image until the palette is filled or until the function has reached the end of the image.

Example:
```
imageobj.SaveImage "test.png", "PNG", 4
```

Positive: Good to use if the image contains fewer colors than palette entries. The colors in the palette matches exactly the value in the image.
Negative: Pretty slow to generate.

GreyScale Palette

This function generates a greyscale palette. Each RGB value is presented in the same matter for the red, green, and blue intensity value. The function splits up the color range in equally divided parts as shown below:

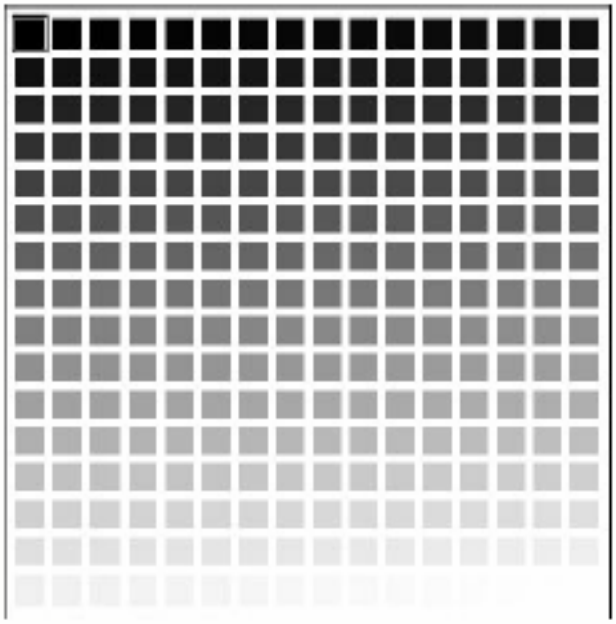


Figure 8: A greyscale palette for a 8 bit image.

Example:
```
imageobj.StreamImage Response, "BMP", 8, "greyscale"
'or (same)
imageobj.StreamImage Response, "BMP", 8, 1
```

Positive: Fast to generate.
Negative: No colors...

Rainbow Palette

Number	Blue	Green	Red
0	0	0	0
1	0	0	128
2	0	128	0
3	0	128	128
4	128	0	0
5	128	0	128
6	128	128	0
7	128	128	128

Color block

Figure 9: The first eight palette entries in a 4 bit image using the rainbow function.

Note that the black color (0x00000) value is always presented one time and at the first color entry. Depending on the number of palette entries, each color block (7 entries) is added continuously after the black color. The rainbow palette below is the result of the algorithm above, used by an 8 bit image.

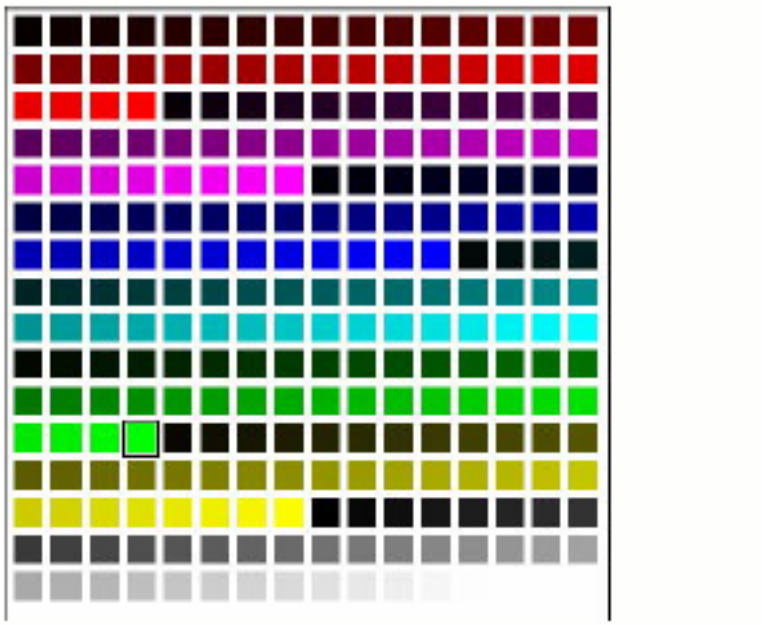


Figure 10: A rainbow palette for an 8 bit image.

Example:

```
imageobj.SaveImage "c:\test.bmp", "BMP", 4, "rainbow"
'or
imageobj.StreamImage Response, "BMP", 4, 2
```

Positive: Fast to generate.
Negative: The steps between the colors are large. It uses colors which are not necessarily needed.

Nearest Color Palette

The nearest color function loops through the pixel values in the image to select the nearest given neighbor. Here the algorithm has split the pixel to a valid RGB value, and rounded off the blue intensity from 20 to its nearest neighbor 16.

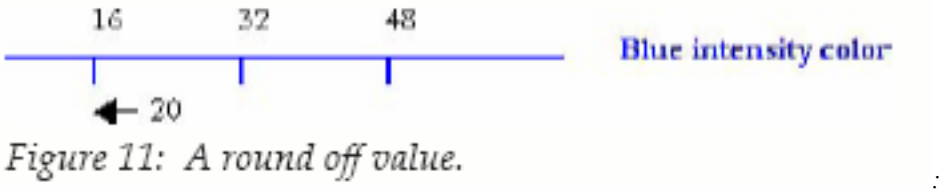


Figure 11: A round off value.

Example:

```
imageobj.StreamImage Response, "BMP", 8, "nearest"
'or
imageobj.StreamImage Response, "BMP", 8, 3
```

Positive: Good when using many colors.
Negative: Slow to generate.

JPG Compression

To accomplish different qualities of the JPG image when exporting the image, the compression rate must be set. w3 Image uses as default a 90% compression rate.
This could be changed as follows:

```
imageobj.StreamImage Response, "JPG", 24, 10
```

Note that the JPG compression rate is now 10% instead of a 90% value. Instead of using the fourth parameter to describe the palette function, JPG uses this parameter as an input when creating and compressing the image.

Chapter 3: Image Formats

w3 Image currently supports these image formats:
* *BMP (Bitmap)*
* *JPG (Joint Photographic Experts Group)*
* *PNG (Portable Network Graphic).*
* *GIF (Graphics Interchange Format)*
* *TIFF (Tagged Image File Format)*
* *TGA or TARGA (Truevision Advanced Raster Graphics Adapter)*
* *ICO (Windows Icon)*
* *WBMP (Wireless Bitmap)*

BMP

The Bitmap file format (BMP) is used for bitmap graphics. Unlike other file formats, which store image data from top to bottom and pixels in red/green/blue order, the BMP format stores image data from bottom to top and pixels in blue/green/red order. Compression of BMP files is not supported, so they are usually very large. When saving a file to the BMP format, add the ".bmp" file extension to the end of its file name.

Example:

```
imageobj.LoadImage "example.bmp"
'Stream a 8 bit BMP image, use a greyscale palette
imageobj.StreamImage Response, "BMP", 8, "greyscale"
```

w3 Image supports: 1, 4, 8, and 24 bit files.
Positive: Easy and fast to generate.
Negative: Large files (uncompressed).

JPG

The Joint Photographic Experts Group format (JPEG) is one of the most popular formats for web graphics. It supports 24 bits of color information, and is most commonly used for photographs and similar continuous-tone bitmap images. The JPEG file format stores all of the color information in an RGB image, then reduces the file size by compressing it, or saving only the color information that is essential to the image. Most imaging applications and plug-ins let you determine the amount of compression used when saving a graphic in the JPEG format. When saving a file in the JPEG format, add the ".jpg" file extension to the end of its file name.

Example:

```
imageobj.LoadImage "example.jpg"
'Note: Colordepth is of no importance
imageobj.SaveImage "mytest.jpg", "JPG", 24
```

w3 Image supports: All JPEG's.
Positive: Excellent when dealing with photos, fast to generate, and the size is small.
Negative: Not optimal for images with few colors in terms of quality and size.

PNG

The Portable Network Graphics format (PNG) will likely be the successor to the GIF file format. PNG is not yet widely supported by web browsers; Netscape versions 4.04 and later and Internet Explorer version 4.0b1 and later, currently support this file format. However, PNG is expected to become a mainstream format for web images and could replace GIF entirely. It is platform independent and should be used for single images only (not animation).
Compared with GIF, PNG offers greater color support and better compression, gamma correction for brightness control across platforms, better support for transparency, and a better method for displaying progressive images. When saving an image to the PNG format, add the file extension ".png" to the end of its file name.

Example:

```
imageobj.LoadImage "example.png"
'Stream a 4 bit PNG image, use default palette
imageobj.StreamImage Response, "PNG", 4
```

w3 Image supports: 1, 4, 8, 24, and 32 bit files. If 24 bit color depth is defined,

the image will be saved/streamed as a 32 bit image.
Positive: Generates small images when using painted images.
Negative: Large files if the amount of colors is high, slow to generate.

GIF

The Graphics Interchange Format (GIF) was originally developed by CompuServe in 1987. It is one of the most popular file formats for web graphics and for exchanging graphics files between computers. It is most commonly used for bitmap images composed of line drawings or blocks of a few distinct colors. The GIF format supports 8 bits of color information or less. It also support transparency and animation, but animation is not supported by w3 Image. When saving an image to the GIF format, add the file extension ".gif" to the end of its file name.

Example:

```
imageobj.LoadImage "example.gif"
'Stream a 4 bit GIF image, use default palette
imageobj.StreamImage Response, "GIF", 4
```

w3 Image supports: 1, 4, and 8 bit files.
Positive: Generates small images when using painted images. Support for transparency.
Negative: Supports no more than 256 colors, hence reduces color information when working with more than 256 colors.

TIFF

Tagged Image File Format (TIFF) is a standard file format for storing images as bit maps. It is used especially for scanned images because it can support any size, resolution, and color depth. The TIFF format is a cross-platform image format. It is also able to store multiple images in one file. This is commonly used for FAX images. As a TIFF file may be a collection of images in various pixel and color formats combined in one file, w3 Image does not handle more than one image per TIFF file. When saving an image to the TIFF format, add the file extension ".tif" to the end of its file name.

Example:

```
imageobj.LoadImage "example.tif"
'Save a 4 bit TIFF image, use default palette
imageobj.SaveImage "C:\images\example.tif", "TIF", 4
```

w3 Image supports: 1, 4, 8, and 24 bit files.
Positive: A standard and a cross-platform format, lossless.
Negative: Large file sizes.

TGA

Truevision Advanced Raster Graphics Adapter (TARGA) Targa graphics are a standard output from Targa devices from Truevision and Pinnacle Systems. These devices normally support video editing on personal computers and can be configured to take still "snapshots" from the video and save them in .TGA format. The format is used to support image capturing and is used with a series of high-resolution graphics adaptors.
When saving an image to the TGA format, add the file extension ".tga" to the end

of its file name.

Example:

```
imageobj.LoadImage "example.tga"
'Save a 4 bit TGA image, use default palette
imageobj.SaveImage "C:\images\example.tga", "TGA", 4
```

w3 Image supports: 8 and 24 bit files.
Positive: Standard format within video.
Negative: Large file sizes.

ICO

The Windows Icon format can be used for icons in any program. You need ico files when you want to change the icons on your desktop or if you want to change the icon of an executable program on your computer. An ICO file is actually a collection of bitmap like images in various pixel and color formats combined in one file. Though, w3 Image does not handle more than one image per ICO file.w3 Image support creation of ICO files up to 256*256 pixels of size.
When saving an image to the ICO format, add the file extension ".ico" to the end of its file name.

Example:

```
imageobj.LoadImage "example.ico"
'Save a 4 bit ICO image, use default palette
imageobj.SaveImage "C:\images\example.ico", "ICO", 4
```

w3 Image supports: 1, 4, and 8 bit files.
Positive: Standard file format for icons.
Negative: Large file sizes.

WBMP

Wireless Bitmap. WBMP's are uncompressed, black and white bitmaps that are intended for use on devices with small screens and limited bandwidth connections.w3 Image support creation of WBMP files up to 127*127 pixels of size.
When saving an image to the WBMP format, add the file extension ".wbmp" to the end of its file name.

Example:

```
imageobj.LoadImage "C:\images\test.gif"
'Save a 1 bit WBMP image.
imageobj.SaveImage "C:\images\example.wbmp", "WBMP", 1
```

w3 Image supports: 1 bit files.
Positive: The only image format that wap pages support.
Negative: Uncompressed, doesn't support larger images than 127*127.

Chapter 4: Transparency

Transparency in w3 Image is supported by the PNG and GIF image formats. The transparent color is always placed at index 0 in a palette.

In version 2.0 w3 Image inherits transparency and background-color from the source image loaded. This only applies to GIF and PNG files, since they are the only one with such information.

Background mode

The background mode can either be solid or transparent. In version 1.3 and below the background mode was by default transparent. **In 2.0 it is only transparent when loading transparent GIF and PNG files.**

Background color

The background color is the color you define to be transparent. The background color is by default set to white. Consequently, creating an image where the surface is white (as deafult), the whole image becomes transparent.
To make the red circle in the illustration below become transparent, you set the background color of the image to red.

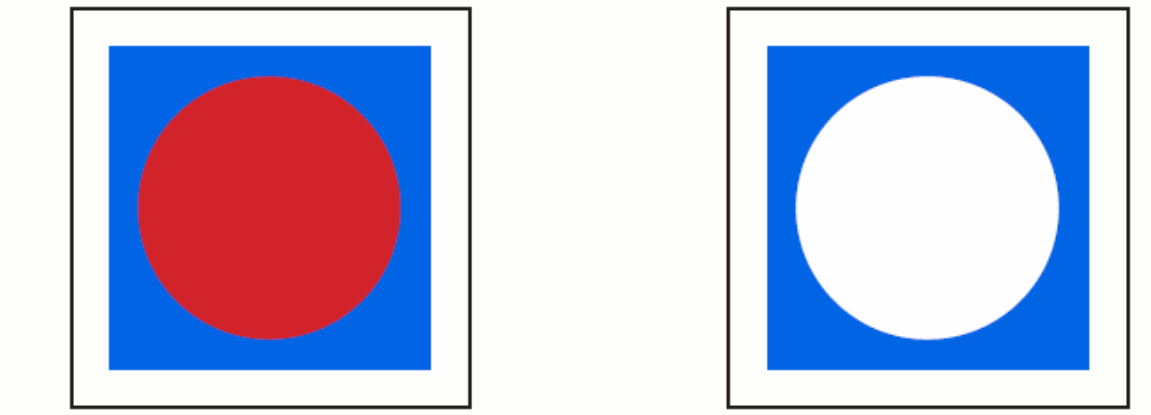


Figure 12: Set the background color of the image to red to make red the transparent color.