



ATL COM Component
FOR [ASP](#) and [ColdFusion](#)
Programming/Documentation: **Lewis Sellers**
Artwork: **Cynthia Rosenberry**
<http://www.intrafoundation.com>
products@intrafoundation.com

[Home](#) | [Examples](#)

IMAGEGRAPHER IS A STRAIGHT-FORWARD ATL COM OBJECT WHICH provides basic graph line and icon drawing functions. It currently works with either GIF or TGA format graphic images.

With it you can load a graphic image into memory (your "drawing canvas"), draw any number of lines optionally terminated by vector arrows/circles, as well as compositing icon images of any size onto your canvas before saving the image to a file.

The primary purpose of this software is for creating dynamic "maps" or graphs.

About GIF and TGA Import/Export

This software support both TGA and GIF graphics file formats.

The TGA format is a clean, basic format for storing graphics images that has been used for used in hardware graphics acelerators. It can handle color bit depths of between 1 and 24 (I.E., from monochrome up to 16-million color truecolor format). Where speed is essential, this is the format to use. The main downside to the format is that most web browsers can not display it.

The GIF format, of course, is the oldest of all graphics formats that is known to web browsers. Transparency is supported for drawn ions, but ignored for loading the initial background graphic. Size can be up to 16k by 16k. GIF's can be 1, 2, 16 or 256 colors. The drawing canvas for this software uses up to 16 million colors (24-bit). All GIF are up-converted to 16-million colors when loaded into the software and down-converted to 256 colors when saved to file.

A special note about GIF in this software: As GIF can at maximum handle only 256 colors at a time, the drawing canvas (which draws internally with a 16-million color palette) must be down-converted to a 256-color format to be saved. There were several techniques that were available to do this, but the one we chose to impliment is called "octree quantization". This is generally considered to be the best, most accurate way to do this conversion (or very close there of). The downside is that it is also among the slowest. If you save a drawing canvas in the TGA format and then an idential copy in the GIF you will easilly notice saving as a GIF is slower. This is partly due to the LZW compression algorithm it uses, but mostly due to the 16-million to 256 color conversion using the octree technique.

File and Security Permissions

In order for this software to work properly your file system (typically NTFS) must be set to allow `read` access from your input folder and `write` access to your output folder.

Also, under IIS the Internet Guest Account needs read and write permissions to create these images.

Functions

Reset

i n

The reset function rests all the drawing parameters back to their defaults and erases the drawing canvas.

(Note that when the function [input](#) is called it first calls this function as well.)

ASP Example

```
obj.Reset()
```

Color

i n

<code>red 0-255</code>	<code>green 0-255</code>	<code>blue 0-255</code>	<code>alpha 0-255</code>
------------------------	--------------------------	-------------------------	--------------------------

o u t

Selects the color you will draw with. It takes four parameters: red, green, blue and alpha. All of these parameters can have a numeric value ranging from 0 to 255 inclusive, 0 indicating a complete absence of the color (black) and 255 full intensity.

(Alpha is currently not used, but included for completeness and possible future upgrades).

The default settings is 255,255,255,0, or in other words, full intensity white.

ASP Example

```
obj.Color 255,255,255,0
```

Marker

i n

<code>size 0-200</code>	<code>angle 0-90</code>	<code>type 0=default 1=triangle/arrow 2=circle</code>
-------------------------	-------------------------	---

Sets the current shape of the *marker*, which is a graphic that can optionally be drawn on the end of all vector lines. (See [draw](#) for more details.)

There are currently three marker types available: an arrow shape, a circle and a triangle. You can control the size of the markers (0-200) and, in the case of the arrow, the angle of separation between the rear arrow points. For the angle of separation the bigger the angle the "fatter" the arrow head will seem.

Type:

- 1=arrow

- 2=circle
- 3=triangle

By default the marker is an arrow with a size of 12.

ASP Example

```
obj.Marker 12,90,0
```

Folders

in

inputfoldername	outputfoldername
-----------------	------------------

Defines the absolute input and output folders that all graphics are loaded from and saved to. This can be changed at will.

The **input** folder applies to all graphic files that are referenced from every function except for the **output** function, which naturally refers to the output folder name.

In other words, the following function call (in ASP)

```
obj.Folders( "c:\inetpub\wwwroot\mysite\images\  
"c:\inetpub\wwwroot\mysite\report_images\  
" )
```

states that any graphics we output will be saved to the folder

c:\inetpub\wwwroot\mysite\report_images\. All other graphics we mention from the **input**, **draw** or **composefb** functions will be looked for in the c:\inetpub\wwwroot\mysite\images\ subfolder.

We could have simply allowed the use of absolute file paths for all references to graphics (and we did in early version of the software), but for more complicated generated images this quickly becomes somewhat tedious and confusing.

ASP Example

```
obj.Folders( "c:\inetpub\wwwroot\mysite\images\  
"c:\inetpub\wwwroot\mysite\report_images\  
" )
```

Input

in

filename

out

filebytes

This function creates the background which you do all of your drawing on from the file name you pass it.

As a practical example to explain this, imagine you have a GIF image file called "world.gif" which is a simple map of the world. Issuing the command (in ASP): `obj.Input("world.gif")` will cause ImageGrapher to call the **Reset** function to clear away any previous graphics you have drawn, then create a blank drawing canvas in memory the exact width and height of the image world.gif, and then finally, copy that image into your drawing canvas.

Now that it is set up, you can issue any of the drawing commands that Image Grapher provides to draw upon this background. (And then use the **output** command to write the final out out as a new graphic file.

The location of the file you name is assumed to be relative to the *input* you specify with the **folders** function. For instance, if `world.gif` is located at `c:\inetpub\wwwroot\mysite\images\` you would have used the **folders** function before hand to tell the program what folders to find all your input graphics.

Returns the a count of the bytes that the uncompressed image takes up in memory.

ASP Example

```
obj.Input( "world.tga" )
```

Output

i n

f i l e n a m e

o u t

f i l e b y t e s

Saves your drawing canvas to a file -- the entire point of this software to an extent. The only parameter to this function is the name of the file you wish to create. The exact location of the output folder for the file is determined by the **folder** function and the graphic fileformat is deduced by the extension you give the file.

For example, say you give the command (in ASP):

```
obj.Output("redridinghoodspaththroughthewoods.gif")
```

Now, if you have issued a folder command previously like:

```
obj.Folders( "c:\inetpub\wwwroot\mysite\images\",
" c:\inetpub\wwwroot\mysite\dynamicimages\" )
```

then what happens is a GIF file is created at `c:\inetpub\wwwroot\mysite\dynamicimages\` called `redridinghoodspaththroughthewoods.gif`.

As faras any return values, this function return a number which is equal to the file size in bytes if the function properly exports an image. Otherwise it returns a non-positive error code. (Currently these codes are: "0" if it did not understand the file type, "-1" if TGA encoded failed and "-2" if GIF encoded failed.)

ASP Example

```
obj.Output( "newworld.gif" )
```

MoveTo

i n

x

y

See **draw**. Moves the starting point for any future draw commands to the specified x,y coordinates (*without* drawing any lines).

ASP Example

```
obj.MoveTo 100,100
```

Draw**in**

x	y	linemarker0:1	iconfile
---	---	---------------	----------

Draws a line from the position of the last draw (or moveto) to the specified x,y coordinates. The line draw is in the color last stated by the [color](#) function.

Draw has two optional parameters.

The optional parameter **line marker** will, when used, draw a special predefined basic marker symbol at the end of the line just drawn, such as arrows, triangles or circles. See [marker](#) for a list of options relating to the marker symbols. Specifying a "1" will use a marker, while a "0" while not use a marker.

Additionally, the parameter referred to as **icon file** can optionally place a small graphic at the end point of the line just draw. The graphic can be of any file type that **Image Grapher** understands (i.e., GIF or TGA) and of any practical size. Only GIF images however will be able to use simple transparency when the icon is drawn.

Additionally, the icons are automatically centered to the give x,y coordinates. Now, it is important to understand this last fact, as any icon you wish to use should be created in such a way that if it is to be used to point to the end of the line, then this "point" should be in the exact center of the icon image. For an example, refer to either *ping.gif* or *ping.tga*. You will notice the tip of the pin in the image is at approximately x,y location 40,40 of these 80 x 80 images.

ASP Example

```
obj.Draw 100,100,0,""
obj.Draw 100,100,1,"icon.gif"
```

Width**out**

widthinpixels

Returns the width of the graphic drawing surface you are using. This is always the same as the last graphic you loaded into memory with the [input](#) function.

ASP Example

```
dim mywidth mywidth=obj.Width
```

Height**out**

heightinpixels

Returns the height of the graphic drawing surface you are using. This is always the same as the last graphic you loaded into memory with the [input](#) function.

ASP Example

```
dim myheight myheight=obj.Height
```

ComposeFB**in**

x	y	imagefile
---	---	-----------

ComposeFB (I.E., Compose Front to Back) draws an image over the background positioning the top/left of the image at the given x,y coordinates. If the image file format support transparency (such as GIF), and the image itself has transparency then it be drawn with such.

Note: Unlike the [draw](#), which centers the graphic it draws, this function takes the x,y position you specify to place the image starting at it's top left most region.

ASP Example

```
obj.composeFB 100,100,"mypic.gif"
```

hBar

i n

x1	x2	y1	y2	y3
----	----	----	----	----

Produces a filled horizontal bar element. (See example image to the right and ASP [example 20.](#))

ASP Example

```
obj.hbar 10,20,60,20,10
```



vBar

i n

y1	y2	x1	x2	x3
----	----	----	----	----

Produces a filled vertical bar element. (See example image to the right and ASP [example 21.](#))

ASP Example

```
obj.vbar 10,20,10,50,40
```



Version History

- **v1.7 November 12th 2004**
 - Added new layout. [Cynthia]
 - Cleaned up docs. [Lewis]
- **v1.6 November 6h 2004**
 - First public version.
 - Added documentation, examples and removed a few minor bugs.
 - Fixed TGA palette decoding bug.
 - Fixed Reset bug that could slightly corrupt output (manifesting as a single pixel shift in some GIFs or Mozilla 1.0PR not being able to decode the GIFs at all.)

(As time permits there are a number of minor upgrades I'd like to make to this software, including two new graphing functions as well as "borrowing" code from our INMaL project for BMP, JPEG and PNG support. It should be noted that this software when it was originally created in April 2003 actually "borrowed" it's GIF and TGA code from an early version of INMaL. Reintegration to the current INMaL code-base, besides adding new graphics formats, will allow this software to use a more optimized (i.e. faster) version of the TGA and GIF

classes as well. (The C++ classes handling LZW based GIF and 16-million color Octree colour conversion were written the same week as this software was originally created, thus, this software is still using the v1.0 of said code. For the most part it works fine, but integration of the current code-base would boost the GIF performance.))

(At some point in the future, the documentation needs a thorough rewriting as well.)

(Currently the code is not thread-safe under IIS.)

Went ahead and added [hbar](#) and [vbar](#) functions at the last minute.

Aside from adding faster GIF quantization code, borrowing the INMaL TrueType font renderer may be in the future for this software.

Removed the 2MB Mars map tests 4 and 17.

- **v1.5 April 21 2003**
- **v1.4/v1.3/v1.2 April 19 2003**
- **v1.1 April 16 2003**
- **v1.0 Mar - April 16 2003**

<http://www.intrafoundation.com>. Copyright © 2003, 2004 by Lewis A. Sellers.







