

The Wayback Machine - <https://web.archive.org/web/20120327064501/http://www.imagemagick.org:80/Usage/>

Examples of ImageMagick Usage

(Version 6)

This web pages presents a set of examples using [ImageMagick](#) from the command line. However they are also examples of what can be done using the ImageMagick Application Programming Interface (API). As such these pages should be the first stop for IM users after reading the terse user manuals.

Often the same questions of "How do I..." gets asked, over and over again on the network. The examples in these web pages I hope will answer most of the common 'how-to' questions that arise.



[Download Page](#), [Fedora RPM](#),
[Linux SRPM](#), [Beta Release](#)
[Discourse Server](#)

Mirrors of IM Examples...

[Germany - Marcus Knappe](#)

Other related sites...

[Fred's ImageMagick Scripts](#)
[RubbleWebs, PHP using IM CLI](#)
[MagickWand Examples in C](#)

▲ [Main ImageMagick Website](#)

▲ [Anthony's Graphics Lab](#)

===== Practical Examples =====

- ▲ [Basic Usage](#)
Basic command and image handling
- ▲ [Image File Handling](#)
Reading and writing images
- ▲ [Common Image Formats](#)
Handling GIF, JPEG, and PNG images
- ▲ [Text to Image Handling](#)
Converting text into images
- ▲ [Compound Font Effects](#)
Font drawing styles and techniques
- ▲ [Annotating Images](#)
Labeling and overlaying images
- ▲ [Thumbnails and Framing](#)
Small reference images of large photos
- ▲ [Photo Handling](#)
Modifying photographs
- ▲ [Lens Correction](#)
Correcting Photo Distortions
- ▲ [Montage, Arrays of Images](#)
Image Indexes and Arrays
- ▲ [Layers of Multiple Images](#)
Combining multiple images together
- ▲ [Animation Basics](#)
Creation and study of animations
- ▲ [Animation Optimizations](#)
Making GIF animations smaller
- ▲ [Animation Modifications](#)
Changing and merging animations
- ▲ [Video Handling](#)
Handling real life video images
- ▲ [Image Comparing](#)
Comparing two or more images
- ▲ [Advanced Techniques](#)
Complex manipulations of images
- ▲ [Background Examples](#)

===== Basic Techniques =====

- ▲ [Canvas Creation](#)
Creating canvas and background images
- ▲ [Color Basics and Channels](#)
Low Level Color Handling
- ▲ [Color Modifications](#)
General Color Changes
- ▲ [Masking and Background Removal](#)
Alpha Channel, and Transparency handling
- ▲ [Color Quantization and Dithering](#)
Reducing the number of colors
- ▲ [Cutting and Bordering](#)
Slicing, Dicing, Framing, Trimming
- ▲ [Resizing or Scaling](#)
Enlarging and Shrinking Images
- ▲ [Compositing Images](#)
Overlaying and Merging Two Images
- ▲ [Drawing on Images](#)
Vector graphics, MVG and SVG images
- ▲ [Simple Image Warping](#)
Flipping, Rotating, Twisting
- ▲ [Distorting Images](#)
Carnival House of Mirrors
- ▲ [Image Transformations](#)
Drastic changes to the look of an image
- ▲ [Image Mapping Effects](#)
Lens, Glass and Ripple Effects
- ▲ [Blurring and Sharpening Images](#)
Blurring, Sharpening and Shadows
- ▲ [Morphology of Shapes](#)
Using Pixel Neighbourhoods
- ▲ [Convolution of Images](#)
Weighted Averaged Neighbourhoods
- ▲ [Fourier Transforms](#)
Modifying Images in the Frequency Domain
- ▲ [Anti-Aliasing](#)

Examples of creating random backgrounds

-  [Repositories Links](#)
Links to other IM scripts and info.
-  [Reference Index](#)
Quick index for specific options.
-  [Support Scripts](#)
Shell Scripts used by examples.

Anti-aliasing effects and problems

-  [Miscellaneous](#)
Miscellaneous Examples
-  [APIs, Scripting, Building](#)
Usage in other environments
-  [Usage under Windows](#)
IM on Windows PC
-  [Development and Bugs](#)
Development Proposals and Bugs, New and Old

Symbols for Added Notes within example pages...



Hint, Tip or Extra Info



For More Advanced Users



Older Version Warnings

Test Image Storage Directories... [Small Images](#) ([image display](#)), [Photographs](#) ([fancy index](#))

ImageMagick Examples - Introductory Notes

What is ImageMagick? A no holds barred summary

[ImageMagick](#) is designed for batch processing of images. That is, allow you to create more complex methods and techniques in a script (shell, dos, perl, PHP, etc) so that you can then apply the image processing operations to many images, or as part of a sub-system of some other tool, such as a web application, video processing tool, panorama generator, and so on. *It is not a GUI image editor.*

ImageMagick is first of all an image-to-image converter. It was what it was originally designed to do. That is it will convert an image in just about any image format (tell us if it can't) to any other image format.

But it is also a library of Image Processing Algorithms. These can be access via the command line, and shell/dos scripts, which is what these example pages are specifically about, or via a large number of programming languages, such as C, C++, Perl, Ruby, PHP, etc, etc, etc. (see [ImageMagick API's](#))

Speed was however never a major feature, more the quality of its image results. That is not to say that it can't transform images in a reasonable time, but it is not blindingly fast. Because of this it can be slow for some processing operations, especially in its attempts to compress images into image formats that have limited capability.

ImageMagick however concerns itself mainly with images in the form of a 'raster' or "rectangular array of pixels". It will handle 'vector' image formats like Postscript and PDF, but at the cost of converting those images into a 'raster', and generating a vector image wrapper around a raster image. As a result vector images are often processed badly in the default case, but specific options can be used to improve this situation. See, [A word about Vector Image formats](#).

About these Examples of ImageMagick Usage

These pages were developed from and are a continuation of my [Collection of ImageMagick Hints and Tips](#) page I first started in 1993 (and placed on the new fangled world-wide-web making its appearance around the same time). Many aspects of IM and notes not included in these pages are still present in that document. However while these pages were designed for you to look at, the hints and tip document was only for my own edification, so may be vague or chaotic in places. You are welcome to look at it, learn and make comments on it.

Other examples were grabbed or developed from answers to users on the [IM Forums](#) or contributed to me as solutions to various problems.

I look forward to suggestions, and email from other IM users, with such email generally resulting in improvements and expansions to these example pages.

Command Line Environments

All examples are written for use on UNIX, and specifically Linux systems, using BASH scripting. As a consequence some examples use a shell 'for-do' loop, and most use a backslash '\' at the end of a line to continue the command onto the next line. The longer commands are broken into separate lines to try to further highlight the steps being applied in that command.

However you can still use these examples from **PC Windows Batch Scripts** with some changes to the handling of certain characters. **PHP Scripts** also need to make some slight modifications to examples, to run the command directly from PHP 'system' calls.

See [Windows Usage](#) and [API's and Scripting](#) for more information on using the the ImageMagick commands in these alternative environments. Contributions and test examples welcome.

PerlMagick, or other APIs

The examples should also be able to be converted to run from all application interfaces, such as Perl, C, C++, Ruby, PHP MagickWand, and so on. I recommend trying things out on the command line first, until you get right, then convert the resulting operations to the specific API you are using.

The command line really only deals with a single image sequence at any one time, though this has improved enormously with IM version 6. But APIs do not have this problem, allowing you to manipulate multiple image sequences separately or together to form more complex scripts.

This ability will also simplify the examples and remove the need to save images as temporary files, as many of the command line examples require. As such with an API only permanent and semi-permanent images need saved to disk.

Basically, let the example pages give you a start, to see just what is possible with ImageMagick, develop just what you want to do, before encoding the results in your own scripts and API code, where it is harder to make extensive changes.

I also recommend you comment your API code, heavily, adding the command line equivalents to what you are trying to do, if possible. This lets you check and compare results using the command line letting you debug problems that you may later come across, especially as improvements are made to image processing in the Core ImageMagick Library.

Downloading Input Images and Results

As much as possible, I try to use an IM built-in image (such as "logo:", or "rose:") as an input image to IM example commands, or generate an input images from IM commands. I also often re-use the output of previous commands, in the later examples. This means you usually do not need to download 'test' images to try out the examples yourself with your own Imagemagick.

However such generated or built-in images are not always convenient, so when I do use an external image, I tend to re-use that input image, or the results of previous examples, over and over, for all the examples of that section.

That said, the original input image is often only displayed in cases where the image changes dramatically, or in very minor ways by the example command.

In any case just about all the IM example commands given are also executed in the same web directory in which they appear, to produce the results also shown. That is the command you see is the command that was actually used to generate the image.

As the command is executed in the same directory you can edit the pages URL link in your browser to download or view the input image(s) used by the command shown. An extra copy of external source images has also been placed in the "[images](#)" and "[img_photos](#)" sub-directories. See also the example of a [Fancy Photo Index](#) of those images.

If text output or image information is produced by an example, it is saved to a text file, and an image generated for direct display on the web page. Selecting the text output image will jump you to a copy of the actual text output by the command.

In all these examples, clicking on the image results should let you download the actual output resulting from example IM commands. Be warned however that not all browsers will understand all image formats.

External Image Sources

By the way, most of the source images used come from [Anthony's Icon Library](#), especially the [background tiles](#), [large clip-art](#), and [dragons](#) sections of the library. (I like dragons!)

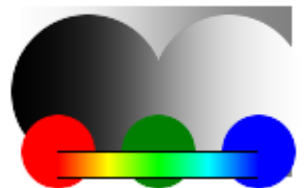
This library actually predates the WWW. I created it in 1991 due to the lack of good clean iconic images for use on the X window system. The advent of the WWW has of course changed this, but my original library still exists and remains available as a source of image, though is not actively growing.

Larger and some specific images are often contributed to IM examples, by the author of the example (who is listed at the bottom of those contributed sections).

If you are looking for a specific image I recommend using [Google Image Search](#) to find something appropriate. You can of course convert or resize such images using IM for your own purposes, though I would be careful about copyright if you plan to use such images commercially.

PNG Images on Web Pages

In many examples, I use a PNG image format, such as that shown to the right of this text. This image format allows you to save image with semi-transparent pixels, a feature few other image formats provide. It is also a very well understood image format and as such usable by most image programs and web browsers today.



That said, some web browsers however do NOT display transparent "PNG" images correctly (most notably Microsoft Internet Explorer v6, though IE v7 does). Because of this I generally prefer to use JPEG and GIF image formats, and only use PNG when generating images with semi-transparent pixels, or requiring a exact colors for later examples.

To allow IE v6 browsers to display PNG images, I use a special 'style sheet' using a complex java script. For information on this, see [PNG with transparency for IE](#), though this is technically not a ImageMagick problem.

Displaying Images on your Screen

Display problems also occur when displaying images on screen, as such I recommend using a command like the following to tile a 'checkerboard' pattern underneath the image to highlight any transparent and semi-transparent pixels in the results.

```
composite -compose Dst_Over -tile pattern:checkerboard image.png x:
```



The source image used in the results above is a special PNG format [test image](#), which was generated using the "[generate_test](#)" shell script. Normally the command would output the results to your display, and not onto the not in a web page like this.

If you look carefully you can see the checkerboard pattern though the semi-transparent colors of the resulting image. However the actual image displayed is fully opaque, so should work on all displays, web browsers, and image viewers.

As of IM v6.0.2, the "display" program performs something like this automatically, though does not seem to handle images using color tables (GIF) in this way. Also see [Show Output Display](#) for another way of displaying results directly on the screen without saving them.

Font Usage

The fonts I use in these examples are from a small collection true type fonts I have found over the years, and saved for my own use. Some of these are copyright, so I cannot publish them online.

You are however welcome to substitute other fonts that you have available, as the examples should work (with some image size changes) with any appropriate font you have available on your system, such as the Microsoft "Arial" font, or even "Times-BoldItalic", one of which should work on most systems.

To see what fonts are currently available on your version of IM, run the following command...

```
convert -list type      # for IM older than v6.3.5-7
convert -list font      # for newer versions
```

WARNING: If the font requested is not found, ImageMagick will silently substitute a default font, which is usually very plain looking (Arial or Times). So test the font before hand, to make sure that it is the one you want, and not the default.

On my Linux system, I use a special Perl script "[imagemick_type_gen](#)" to generate a "type.xml" font listing file, and is saved in the ".magick" sub-directory of your home. This script locates (run "updatedb" first if you just added new fonts), and describes all the fonts I have available on my system. This means I only need to specify the name of the font I want to use and not the full path to a specific font file.

This lets me...

```
# Instead of using the command...
convert -font $HOME/lib/font/truetype/favorite/candice.ttf \
        -pointsize 72 label:Anthony  anthony.gif

# I can use the simpler font label...
convert -font Candice -pointsize 72 label:Anthony  anthony.gif
```



Before IM v6.1.2-3, the "type.xml" file was named "type.mgk". If you are using a earlier version of IM please note this change.

The fonts used in these examples are listed in a [Montage of Example Fonts Example](#). My personal favorite is Candice, so it gets used quite a bit.



If you also like the 'candice' font, or any of the other fonts I use, grab it from [Free Fonts](#) or [1001 Fonts .com](#).

Example Page Updates

These example pages are in an on going cycle of improvement. Generally I find I stop adding to pages for long periods of time when my interests become focused on other things.

Also all examples are re-built using the latest beta release for IM, allowing me to see changes and bugs that may appear in each version of IM before it is generally released. However the example images shown is what the given IM command produces on my system. If you get something different your IM is probably a much older version, with old bugs, or your IM is incorrectly installed.

Note that [emailing me](#), or discussing some aspect of ImageMagick on the [IM Mailing List](#), or [IM User Forum](#) will generally result in whole sections being added or new examples to existing sections created. The more discussion, the better the examples become.

If you are doing anything interesting with IM, please share, and allow me provide examples of your technique to the rest of the IM community. Some of the biggest advances in IM usage have come from users just like you.

Special Thanks

A special thanks goes to Cristy, who has tirelessly spent months, upgrading, bug fixing, and putting up with my off the wall suggestions. Especially with regards to my major suggestions for the command line processing, parenthesis, image sequence operators, and GIF animation processing.

He has done a marvelous job to make version 6 the best and most advanced command line image processing program available, and while most users will not show that appreciation, I certainly do appreciate the effort he has put into IM.

I also like to thank Gabe Schaffer, who has been most helpful in discussions involving, the JPEG format and library, affine matrix operators, and the [Magick Vector Graphics](#) in general.

And to Glenn Randers-Pehrson, who looks after the PNG coder module, and seems to like to be involved with Color Quantization and Dithering. He was the first to add 'halftone' dithering to IM, before I revised the whole section in the IM Core, and later added new dithers to the ordered dither configuration file.

And finally I want to thank the huge number of people with problems, suggestions, and solutions, who generally lurk on the [IM Mailing List](#), or [IM User Forum](#). Many now have their names as contributors of ideas or suggestions throughout IM Examples.

I also like to thank the people who regularly answer questions on the forums, such as 'Bonzo' and his web site [RubbleWebs](#), detailing using IM command line from PHP scripts. Also 'scri8e' and his web site [Moons Stars](#) for glitter and star handling. Of course I also like to thank Pete 'el_supremo' (see his [MagickWand C-Programming](#)), and the many others who regularly answer other peoples questions.

A special thanks goes to [Fred Weinhaus](#) a researcher from the early days of image processing who was a major help in the initial implementation of [General Image Distortion Operator](#). You can see Fred's ImageMagick scripts on [Fred's ImageMagick Site](#), often as a proof of concept for future IM additions.

Also to Professor Nicolas Robidoux, of Laurentian University, for the reworking of the [Elliptical Weighted Average Resampling](#), which vastly improves the output of [General Image Distortion](#).

There are also users of ImageMagick that rather than hiding how they do things lets you see and modify the IM commands directly as part of the large project.

Well enough "Yadda, yadda, yadda", go look at some of the examples.

Created: 7 November 2003

Updated: 15 March 2011

Author: [Anthony Thyssen](mailto:A.Thyssen_AT_griffith.edu.au), <A.Thyssen_AT_griffith.edu.au>

*Examples Generated with: **IM v6.7.0-10 2011-11-03***

Licence: IM Examples follows the same [Licence as ImageMagick](http://www.imagemagick.org/Usage/)

URL: <http://www.imagemagick.org/Usage/>