Gifsicle manual

gifsicle

manipulates GIF images and animations

SYNOPSIS

gifsicle [options, frames, and filenames]...

DESCRIPTION

gifsicle is a powerful command-line program for creating, editing, manipulating, and getting information about GIF images and animations.

Gifsicle normally processes input GIF files according to its command line options and writes the result to the standard output. The **-i** option, for example, tells **gifsicle** to interlace its inputs:

```
gifsicle -i < pic.gif > interlaced-pic.gif
```

Gifsicle is good at creating and manipulating GIF animations. By default, it combines two or more input files into a “flipbook” animation:

```
gifsicle pic1.gif pic2.gif pic3.gif > animation.gif
```

Use options like **--delay**, **--loopcount**, and **--optimize** to tune your animations.

To modify GIF files in place, use the **--batch** option. With **--batch**, **gifsicle** will modify the files you specify instead of writing a new file to the standard output. To interlace all the GIFs in the current directory, you could say:

```
gifsicle --batch -i *.gif
```

New users may want to skip to the Examples section at the end.

CONCEPT INDEX

Concepts are on the left, relevant gifsicle options are on the right.

Image transformations

COMMAND LINE

gifsicle’s command line consists of GIF

input files and options. Most options start with a dash (-) or plus (+); frame selections, a kind of option, start with a number sign (#). Anything else is a GIF input file.

gifsicle reads and processes GIF input files in order. If no GIF input file is given, or you give the special filename '-', it reads from the standard input.

gifsicle exits with status 0 if there were no errors and status 1 otherwise.

OPTIONS

Every option has a long form, '**--long-descriptive-name**'. You don't need to type the whole long descriptive name, just enough to make it unambiguous.

Some options also have a short form, '**-X**'. You can combine short options if they don't take arguments: '**-IIb**' is the same as '**-I -I -b**'. But be careful with options that do take arguments: '**-cblah**' means '**-c** blah', not '**-c -b -l -a -h**'.

Many options also have a converse, '**--no-option**', which turns off the option. You can turn off a short option '**-X**' by saying '**+X**' instead.

Mode Options

Mode options tell **gifsicle** what kind of output to generate. There can be at most one, and it must precede any GIF inputs.

--merge, -m
: Combine all GIF inputs into one file with multiple frames and write that file to the standard output. This is the default mode.

--batch, -b
: Modify each GIF input in place by reading and writing to the same filename. (GIFs read from the standard input are written to the standard output.)

--explode, -e
: Create an output GIF for each frame of each input file. The output GIFs are named ‘xxx.000’, ‘xxx.001’, and so on, where ‘xxx’ is the name of the input file (or whatever you specified with ‘**--output**’) and the numeric extension is the frame number.

--explode-by-name, -E
: Same as **--explode**, but write any named frames to files ‘xxx.*name*’ instead of ‘xxx.*frame-number*’. Frames are named using the ‘**--name**’ option.

General Options

General options control the information **gifsicle** prints and where it writes its output. The info options and **--verbose** can be turned off with ‘**--no-X**’.

--info, **-I**
Print a human-readable description of each input GIF to the standard output, or whatever file you specify with **-o**. This option suppresses normal output, and cannot be combined with mode options like **--batch**. If you give two **--info** or **-I** options, however, information is printed to standard error, and normal output takes place as usual.

--color-info, **--cinfo**
Like **--info**, but also print information about input files' colormaps.

--extension-info, **--xinfo**
Like **--info**, but also print any unrecognized GIF extensions in a **hexdump**(1)-like format.

--size-info, **--sinfo**
Like **--info**, but also print information about compressed image sizes.

--help, **-h**
Print usage information and exit.

-o *file*
--output *file*
Send output to *file*. The special filename '-' means the standard

output.

--verbose, -V
Print progress information (files read and written) to standard error.

--no-warnings, -w
Suppress all warning messages.

--no-ignore-errors
Exit with status 1 when encountering a very erroneous GIF. Default is to muddle on.

--version
Print the version number and some short non-warranty information and exit.

--careful
Write slightly larger GIFs that avoid bugs in some other GIF implementations. Some Java and Internet Explorer versions cannot display the correct, minimal GIFs that Gifsicle produces. Use the **--careful** option if you are having problems with a particular image.

--conserve-memory
Conserve memory usage at the expense of processing time. This may be useful if you are processing large GIFs on a computer without very much memory. Or say **--no-conserve-memory**.

--nextfile
: Allow input files to contain multiple concatenated GIF images. If a filename appears multiple times on the command line, **gifsicle** will read a new image from the file each time. This option can help scripts avoid the need for temporary files. For example, to create an animated GIF with three frames with different delays, you might run "**gifsicle --nextfile -d10 - -d20 - -d30 - > out.gif**" and write the three GIF images, in sequence, to **gifsicle**'s standard input.

--multifile
: Like **--nextfile**, but read *as many GIF images as possible* from each file. This option is intended for scripts. For example, to merge an unknown number of GIF images into a single animation, run "**gifsicle --multifile - > out.gif**" and write the GIF images, in sequence, to **gifsicle**'s standard input. Any frame selections apply only to the last file in the concatenation.

Frame Selections

A frame selection tells **gifsicle** which frames to use from the current input file. They are useful only for animations, as non-animated GIFs only have one frame. Here are the acceptable forms for frame specifications.

#*num*
: Select frame *num*. (The first frame is ‘**#0**’. Negative numbers count backwards from the last frame, which is ‘**#-1**’.)

#*num1***-***num2*
: Select frames *num1* through *num2*.

#*num1***-**
: Select frames *num1* through the last frame.

#*name*
: Select the frame named *name*.

The ‘#’ character has special meaning for many shells, so you generally need to quote it.

For example,

```
gifsicle happy.gif "#0"
```

uses the first frame from happy.gif;

```
gifsicle happy.gif "#0-2"
```

uses its first three frames; and

```
gifsicle happy.gif "#-1-0"
```

uses its frames in reverse order (starting from frame #-1—the last frame—and ending at frame #0—the first).

The action performed with the selected frames depends on the current mode. In merge mode, only the selected frames are merged into the output GIF. In batch mode, only the selected frames are

modified; other frames remain unchanged. In explode mode, only the selected frames are exploded into output GIFs.

Frame Change Options

Frame change options insert new frames into an animation or replace or delete frames that already exist. Some things—for example, changing one frame in an animation—are difficult to express with frame selections, but easy with frame changes.

--delete *frames* [*frames*...]
: Delete *frames* from the input GIF.

--insert-before *frame other-GIFs*
: Insert *other-GIFs* before *frame* in the input GIF.

--append *other-GIFs*
: Append *other-GIFs* to the input GIF.

--replace *frames other-GIFs*
: Replace *frames* from the input GIF with *other-GIFs*.

--done
: Complete the current set of frame changes.

The *frames* arguments are frame selections (see above). These arguments always refer to frames from the *original* input GIF. So, if ‘a.gif’ has 3 frames and ‘b.gif’ has one, this command

```
gifsicle a.gif --delete "#0" --replace "#2" b.gif
```

will produce an output animation with 2 frames: ‘a.gif’ frame 1, then ‘b.gif’.

The *other-GIFs* arguments are any number of GIF input files and frame selections. These images are combined in merge mode and added to the input GIF. The *other-GIFs* last until the next frame change option, so this command replaces the first frame of ‘in.gif’ with the merge of ‘a.gif’ and ‘b.gif’:

```
gifsicle -b in.gif --replace "#0" a.gif b.gif
```

This command, however, replaces the first frame of ‘in.gif’ with ‘a.gif’ and then processes ‘b.gif’ separately:

```
gifsicle -b in.gif --replace "#0" a.gif --done b.gif
```

Warning: You shouldn’t use both frame selections and frame changes on the same input GIF.

Image Options

Image options modify input images—by changing their interlacing, transparency, and cropping, for example. Image options have three forms: ‘**--X**’, ‘**--no-X**’, and ‘**--same-X**’. The ‘**--X**’ form selects a value for the feature, the ‘**--no-X**’ form turns off the feature, and the ‘**--same-X**’ form means that the feature’s value is copied from each input. The default is always ‘**--same-X**’. For example,

-background="#0000FF**"** sets the background color to blue, **--no-background** turns the background color off (by setting it to 0), and **--same-background** uses input images' existing background colors. You can give each option multiple times; for example,

```
gifsicle -b -O2 -i a.gif --same-interlace b.gif c.gif
```

will make 'a.gif' interlaced, but leave 'b.gif' and 'c.gif' interlaced only if they were already.

-B *color*
--background *color*
Set the output GIF's background to *color*. The argument can have the same forms as in the **--transparent** option below.

--crop *x1,y1-x2,y2*
--crop *x1,y1+widthxheight*
Crop the following input frames to a smaller rectangular area. The top-left corner of this rectangle is (*x1*,*y1*); you can give either the lower-right corner, (*x2*,*y2*), or the width and height of the rectangle. In the *x1,y1+widthxheight* form, *width* and *height* can be zero or negative. A zero dimension means the cropping area goes to the edge of the image; a negative dimension brings the cropping area that many pixels back from the image edge. For example, **--crop** 2,2+-2x-2

will shave 2 pixels off each side of the input image. Cropping takes place before any rotation, flipping, resizing, or positioning.

--crop-transparency
Crop any transparent borders off the following input frames. This happens after any cropping due to the **--crop** option. It works on the raw input images; for example, any transparency options have not yet been applied.

--flip-horizontal
--flip-vertical
Flip the following frames horizontally or vertically.

-i
--interlace
Turn interlacing on.

-S *widthxheight*
--logical-screen *widthxheight*
Set the output logical screen to *widthxheight*. **--no-logical-screen** sets the output logical screen to the size of the largest output frame, while **--same-logical-screen** sets the output logical screen to the largest input logical screen. **--screen** is a synonym for **--logical-screen**.

-p *x,y*
--position *x,y*
Set the following frames' positions to

(*x*,*y*). **--no-position** means **--position** 0,0. Normally, **--position** *x*,*y* places every succeeding frame exactly at *x*,*y*. However, if an entire animation is input, *x*,*y* is treated as the position for the animation.

--rotate-90
--rotate-180
--rotate-270
Rotate the following frames by 90, 180, or 270 degrees. **--no-rotate** turns off any rotation.

-t *color*
--transparent *color*
Make *color* transparent in the following frames. *Color* can be a colormap index (0-255), a hexadecimal color specification (like "#FF00FF" for magenta), or slash- or comma-separated red, green and blue values (each between 0 and 255).

Extension Options

Extension options add non-visual information to the output GIF. This includes names, comments, and generic extensions.

--app-extension *app-name extension*
Add an application extension named *app-name* and with the value *extension* to the output GIF. **--no-app-extensions**

removes application extensions from the input images.

-c *text*
--comment *text*
Add a comment, *text*, to the output GIF. The comment will be placed before the next frame in the stream. **--no-comments** removes comments from the input images.

--extension *number extension*
Add an extension numbered *number* and with the value *extension* to the output GIF. *Number* can be in decimal, octal, hex, or it can be a single character like ‘n’, whose ASCII value is used. **--no-extensions** (or **+x**) removes extensions from the input images.

-n *text*
--name *text*
Set the next frame’s name to *text*. This name is stored as an extension in the output GIF (extension number 0xCE, followed by the characters of the frame name). **--no-names** removes name extensions from the input images.

Animation Options

Animation options apply to GIF animations, or to individual frames in GIF animations. As with image options, most animation options have three forms, ‘**--X**’, ‘**--no-X**’, and ‘**--same-X**’, and you

can give animation options multiple times; for example,

```
gifsicle -b a.gif -d50 "#0" "#1" -d100 "#2" "#3"
```

sets the delays of frames 0 and 1 to 50, and frames 2 and 3 to 100.

-d *time*
--delay *time*
Set the delay between frames to *time* in hundredths of a second.

-D *method*
--disposal *method*
Set the disposal method for the following frames to *method*. A frame's disposal method determines how a viewer should remove the frame when it's time to display the next. *Method* can be a number between 0 and 7 (although only 0 through 3 are generally meaningful), or one of these names: **none** (leave the frame visible for future frames to build upon), **asis** (same as "none"), **background** (or **bg**) (replace the frame with the background), or **previous** (replace the frame with the area from the previous displayed frame). **--no-disposal** means **--disposal**=**none**.

-l[*count*]
--loopcount[=*count*]
Set the Netscape loop extension to *count*. *Count* is an integer, or **forever**

to loop endlessly. If you supply a **--loopcount** option without specifying *count*, Gifsicle will use **forever**. **--no-loopcount** (the default) turns off looping.

Set the loop count to one less than the number of times you want the animation to run. An animation with **--no-loopcount** will show every frame once; **--loopcount**=1 will loop once, thus showing every frame twice; and so forth. Note that **--loopcount**=0 is equivalent to **--loopcount**=forever, not **--no-loopcount**.

-O[*level*]
--optimize[=*level*]
Optimize output GIF animations for space. *Level* determines how much optimization is done; higher levels take longer, but may have better results. There are currently three levels:

-O1
Stores only the changed portion of each image. This is the default.
-O2
Also uses transparency to shrink the file further.
-O3
Try several optimization methods (usually slower, sometimes better results).

Other optimization flags provide finer-grained control.

-Okeep-empty
Preserve empty transparent frames (they are dropped by default).

You may also be interested in other options for shrinking GIFs, such as **-k**, **--lossy**, and **--no-extensions**.

-U
--unoptimize
Unoptimize GIF animations into an easy-to-edit form.

GIF animations are often optimized (see **--optimize**) to make them smaller and faster to load, which unfortunately makes them difficult to edit. **--unoptimize** changes optimized input GIFs into unoptimized GIFs, where each frame is a faithful representation of what a user would see at that point in the animation.

Image Transformation Options

Image transformation options apply to entire GIFs as they are read or written. They can be turned off with '**--no-**

option'.

--resize *widthxheight*
Resize the output GIF to the given *width* and *height*. If *width* or *height* is an underscore '_', that dimension is chosen so that the aspect ratio remains unchanged. Resizing happens after all input frames have been combined and before optimization. Resizing uses logical screen dimensions; if the input stream has an unusual logical screen (many GIF displayers ignore logical screens), you may want to provide **--no-logical-screen** (or **+S**) to reset it so **gifsicle** uses image dimensions instead. See also **--resize-method**.

--resize-width *width*
--resize-height *height*
Resize to a given width or height, preserving aspect ratio. Equivalent to **--resize** *width*x_ or **--resize** _x*height*.

--resize-fit *widthxheight*
--resize-touch *widthxheight*
Resize the output GIF to fit within a rectangle with dimensions *widthxheight*. The aspect ratio remains unchanged. The **--resize-fit** option only shrinks the image—no resize is performed if the GIF already fits within the rectangle. Either *width* or *height* may be an underscore '_', which

leaves that dimension unconstrained.

--resize-fit-width *width*
--resize-fit-height *height*
--resize-touch-width *width*
--resize-touch-height *height*
Like **--resize-fit** and **--resize-touch**, but constrains only one dimension.

--scale *Xfactor*[x*Yfactor*]
Scale the output GIF's width and height by *Xfactor* and *Yfactor*. If *Yfactor* is not given, it defaults to *Xfactor*. Scaling happens after all input frames have been combined and before optimization.

--resize-method *method*
Set the method used to resize images. The 'sample' method runs very quickly, but when shrinking images, it produces noisy results. The 'mix' method is somewhat slower, but produces better-looking results. The default method is currently 'mix'.

Details: The resize methods differ most when shrinking images. The 'sample' method is a point sampler: each pixel position in the output image maps to exactly one pixel position in the input. When shrinking, full rows and columns from the input are dropped. The other methods use all input pixels, which generally produces

better-looking images. The ‘box’ method, a box sampler, is faster than the more complex filters and produces somewhat sharper results, but there will be anomalies when shrinking images by a small amount in one dimension. (Some output pixels will correspond to exactly 1 input row or column, while others will correspond to exactly 2 input rows or columns.) The ‘mix’ method is a full bilinear interpolator. This is slower and produces somewhat blurrier results, but avoids anomalies.

Gifsicle also supports more complex resamplers, including Catmull-Rom cubic resampling (‘catrom’), the Mitchell-Netravali filter (‘mitchell’), a 2-lobed Lanczos filter (‘lanczos2’), and a 3-lobed Lanczos filter (‘lanczos3’). These filters are slower still, but can give sharper, better results.

--resize-colors *n*

Allow Gifsicle to add intermediate colors when resizing images. Normally, Gifsicle’s resize algorithms use input images’ color palettes without changes. When shrinking images with very few colors (e.g., pure black-and-

white images), adding intermediate colors can improve the results. Example: **--resize-colors** *64* allows Gifsicle to add intermediate colors for images that have fewer than 64 input colors.

Color Options

Color options apply to entire GIFs as they are read or written. They can be turned off with '**--no-option**'.

-k *num*
--colors *num*
Reduce the number of distinct colors in each output GIF to *num* or less. *Num* must be between 2 and 256. This can be used to shrink output GIFs or eliminate any local color tables.

Normally, an adaptive group of colors is chosen from the existing color table. You can affect this process with the **--color-method** option or by giving your own colormap with **--use-colormap**. Gifsicle may need to add an additional color (making *num*+1 in all) if there is transparency in the image.

--color-method *method*
Determine how a smaller colormap is chosen. '**diversity**', the default, is **xv**(1)'s diversity algorithm, which uses a strict subset of the existing

colors and generally produces good results. ‘**blend-diversity**’ is a modification of this: some color values are blended from groups of existing colors. ‘**median-cut**’ is the median cut algorithm described by Heckbert. **--method** is a synonym for **--color-method**.

-f
--dither[=*method*]
When **--dither** is on and the colormap is changed, combinations of colors are used to approximate missing colors. This looks better, but makes bigger files and can cause animation artifacts, so it is off by default.

Specify a dithering algorithm with the optional *method* argument. The default, ‘**floyd-steinberg**’, uses Floyd-Steinberg error diffusion. This usually looks best, but can cause animation artifacts, because dithering choices will vary from frame to frame. Gifsicle also supports ordered dithering algorithms that avoid animation artifacts. The ‘**ro64**’ mode uses a large, random-looking pattern and generally produces good results. The ‘**o3**’, ‘**o4**’, and ‘**o8**’ modes use smaller, more regular patterns. The ‘**ordered**’ mode chooses a good ordered dithering algorithm. For special effects, try the halftone modes ‘**halftone**’, ‘**squarehalftone**’, and

‘**diagonal**’. Some modes take optional parameters using commas. The halftone modes take a cell size and a color limit: ‘**halftone,10,3**’ creates 10-pixel wide halftone cells where each cell uses up to 3 colors.

--gamma *gamma*
Set the gamma correction to *gamma*, which can be a real number or ‘**srgb**’. Roughly speaking, higher numbers exaggerate shadows and lower numbers exaggerate highlights. The default is the function defined by the standard sRGB color space, which usually works well. (Its effects are similar to **--gamma**=2.2.) Gifsicle uses gamma correction when choosing a color palette (**--colors**) and when dithering (**--dither**).

--lossy[=*lossiness*]
Alter image colors to shrink output file size at the cost of artifacts and noise. *Lossiness* determines how many artifacts are allowed; higher values can result in smaller file sizes, but cause more artifacts. The default *lossiness* is 20.

--change-color *color1 color2*
Change *color1* to *color2* in the following input GIFs. (The *color* arguments have the same forms as in the **-t** option.) Change multiple colors by giving the option multiple times.

Color changes don't interfere with one another, so you can safely swap two colors with '**--change-color** *color1 color2* **--change-color** *color2 color1*'. They all take effect as an input GIF is read. **--no-change-color** cancels all color changes.

--transform-colormap *command*
Command should be a shell command that reads from standard input and writes to standard output. Each colormap in the output GIF is translated into text colormap format (see **--use-colormap** below) and piped to the command. The output that command generates (which should also be in text colormap format) will replace the input colormap. The replacement doesn't consider color matching, so pixels that used color slot *n* in the input will still use color slot *n* in the output.

--use-colormap *colormap*
Change the image to use *colormap*. Each pixel in the image is changed to the closest match in *colormap* (or, if **--dither** is on, to a dithered combination of colors in *colormap*). *Colormap* can be **web** for the 216-color "Web-safe palette"; **gray** for grayscale; **bw** for black-and-white; or the name of a file. That file should either be a text file (the format is described below) or a GIF file, whose

global colormap will be used. If **--colors**=*N* is also given, an *N*-sized subset of *colormap* will be used.

Text colormap files use this format:

```
; each non-comment line represents one color, "red green blue"
; each component should be between 0 and 255
0 0 0           ; like this
255 255 255
; or use web hex notation
#ffffff         ; like this
```

EXAMPLES

First, let’s create an animation, ‘anim.gif’:

```
gifsicle a.gif b.gif c.gif d.gif > anim.gif
```

This animation will move very quickly: since we didn’t specify a delay, a browser will cycle through the frames as fast as it can. Let’s slow it down and pause .5 seconds between frames, using the **--delay** option.

```
gifsicle --delay 50 a.gif b.gif c.gif d.gif > anim.gif
```

If we also want the GIF to loop three times, we can use **--loopcount**:

```
gifsicle -d 50 --loop=3 a.gif b.gif c.gif d.gif > anim.gif
```

(Rather than type **--delay** again, we used its short form, **-d**. Many options have short forms; you can see them by running '**gifsicle --help**'. We also abbreviated **--loopcount** to **--loop**, which is OK since no other option starts with 'loop'.)

To explode 'anim.gif' into its component frames:

```
gifsicle --explode anim.gif

ls anim.gif*

anim.gif  anim.gif.000  anim.gif.001  anim.gif.002  anim.gif.003
```

To optimize 'anim.gif':

```
gifsicle -b -O2 anim.gif
```

To change the second frame of 'anim.gif' to 'x.gif':

```
gifsicle -b --unoptimize -O2 anim.gif --replace "#1" x.gif
```

--unoptimize is used since 'anim.gif' was optimized in the last step. Editing individual frames in optimized GIFs is dangerous without **--unoptimize**; frames following the changed frame could be corrupted by the change. Of course, this might be what you want.

Note that **--unoptimize** and **--optimize** can

be on simultaneously. **--unoptimize** affects *input* GIF files, while **--optimize** affects *output* GIF files.

To print information about the first and fourth frames of ‘anim.gif’:

```
gifsicle -I "#0" "#3" < anim.gif
```

To make black the transparent color in all the GIFs in the current directory, and also print information about each:

```
gifsicle -bII --trans "#000000" *.gif
```

Giving **-I** twice forces normal output to occur. With only one **-I**, the GIFs would not be modified.

To change ‘anim.gif’ to use a 64-color subset of the Web-safe palette:

```
gifsicle -b --colors=64 --use-col=web anim.gif
```

To make a dithered black-and-white version of ‘anim.gif’:

```
gifsicle --dither --use-col=bw anim.gif > anim-bw.gif
```

To overlay one GIF atop another—producing a one-frame output GIF that looks like the superposition of the two inputs—use **gifsicle** twice:

```
gifsicle bottom.gif top.gif | gifsicle -U "#1" > result.gif
```

BUGS

Some optimized output GIFs may appear incorrectly on some GIF implementations (for example, Java's); see the **--careful** option.

Please email suggestions, additions, patches and bugs to ekohler@gmail.com.

SEE ALSO

For a tutorial on GIF images and animations, you might try some of the resources listed on-line at webreference.com:
http://www.webreference.com/authoring/graphics/animation.html

AUTHORS

Eddie Kohler <ekohler@gmail.com>
http://www.read.seas.harvard.edu/~kohler/
He wrote it.

Anne Dudfield <annied@frii.com>
http://www.frii.com/~annied/
She named it.

Hans Dinsen-Hansen <dino@danbbs.dk>
http://www.danbbs.dk/~dino/
Adaptive tree method for GIF writing.

Kornel Lesiński
--lossy option.

http://www.lcdf.org/gifsicle/
The **gifsicle** home page.

Back to gifsicle // **Eddie Kohler**