animaGIF

builds an animated GIF image from a series of png, jpg, or bmp images or from a live figure

Syntax

```
animaGIF(files, outgif)
animaGIF(files, outgif, delay)
animaGIF(files, outgif, delay, loops)

idGif = animaGIF(idFig, outgif)
idGif = animaGIF(idFig, outgif, delay)
idGif = animaGIF(idFig, outgif, delay, loops)
animaGIF(idFig, idGif)
animaGIF(idGif)
```

Arguments

files

provides the series of input images files. Two specifications are possible:

1. **filesmask** : If `files` is a single string including a * wildcard character, it represents the path and filenames pattern of images files to be selected and processed. Usually * is used as the generic index of images.

   The corresponding vector of matching filenames is sorted in increasing *numerical* order. Hence, the set `im_1.png`, `im_02.png`, `im_10.png`, `im_11.png`, `im_21.png`, `image_100.png` will be sorted as 1, 2, 10, 11, 21 and 100, while the alphabetical order would sort them as 02, 1, 10, 100, 11, 21.

   The `files` pattern must be non-ambiguous to match numbering digits. If the pointed directory contains only files named like `ima3d_#.png` where "#" stands for the numbering digits, the trivial "*" pattern could match "3" as well as "12" and so would fail, while "_*" would be OK to consider just "12" (or so) as numbering digits.

   If the directory contains some other unrelated files, then the full explicit `ima3d_*.png` mask will be mandatory.

2. **filenames** : If `files` is a vector of strings, it provides the explicit pathnames of files of images to be stacked.

   By default, images will be bundled in the order their file appears in the `files` vector. However, if `files(1)` includes the "*" character, it is considered as the numbering mask and this mask will be used to sort the `files` components in numerical order.

The set of input images can mix different image encodings: png with jpg etc, in any order.

Predefined TMPDIR, SCI, SCIHOME, home... constant paths are supported.

⚠ The GIF encoding is poorly supported for input images.

idFig

Handle of the graphical figure whose snapshot must be added to the animated GIF.

delay

positive number: sleeping time after each image when the animated GIF will be played, in milliseconds. The delay resolution is 10 ms. Default value = 200.

loops

integer in [2, 255], or 0: number of consecutive times the animated GIF will be played when displayed. 0 stands for infinite looping, and is the default value.

⚠ Some rendering softwares ignore this number and always play the GIF indefinitely.

outgif

Single string: Pathname of the animated GIF file to build. The predefined TMPDIR, SCI, SCIHOME, home... constant paths are supported as part of **outgif**.

idGif

Handle of the opened animated GIF as returned by a (previous) call to animaGIF(…)

Description

animaGIF() is a stand-alone Scilab function using only the JVM embedded in Scilab. It does not require any external software like *ImageMagic* or *Gimp*. It does not involve any hard-coded part and so is expected to be stable accross Scilab versions.

Since it uses the JVM, animaGIF() can be used in standard STD Scilab desktop and in No-Window NW (advanced console) modes, but is not available in NWNI batch mode.

Animated GIF from existing images files

animaGIF(files, outgif)

- selects images files according to the given mask and sorts them in increasing numerical indices, or considers given files and possibly sorts them in increasing numerical indices,
- builds the animated GIF image, setting a default pause of 200 ms after each image,
- sets the looping mode to infinite iterations,
- and records the animated GIF image in the file **outgif**.

animaGIF(files, outgif, delay) does the same, but sets the pause's duration as specified by **delay**.

animaGIF(files, outgif, delay, loops) does the same, but sets the chosen number of times the GIF will be played after each loading. To use the default **delay** value, just run animaGIF(files, outgif, , loops).

Animated GIF directly from a figure

This on-the-fly mode is incremental and avoids saving a series of snapshots in possibly numerous files.

idGif = animaGIF(idFig, outgif)

- initiates the animated GIF file **outgif** with the default inter-image pause of 200 ms value and the default infinite looping mode.
- makes a snapshot of the given figure, and adds it to the animated GIF,
- keeps open the animated GIF,
- and returns the handle of the animated GIF image.

idGif = animaGIF(idFig, outgif, delay) and **idGif = animaGIF(idFig, outgif, delay, loops)** do the same but use the specified **delay** (and **loops** number of iterations) instead of the default ones.

idGif = animaGIF(idFig, idGif)

- makes a snapshot of the given figure,
- adds the snapshot to the open animated GIF specified by its **idGif** handle as returned by a previous call,
- and returns the handle of the updated animated GIF image.

animaGIF(idGif) closes the given animated GIF and concludes the job.

- In the animated image, the size of each image is the size of the figure when it is snapshot. Therefore, the figure must not be resized during the GIF stream.
- The figure may be hidden (gcf().visible = "off") when animaGIF(…) is called.
- Before building on-the-fly an animated GIF including many images, the **delay** must be carefully chosen. For the time being, animaGIF(…) does not allow to change the **delay** of an existing animated GIF.

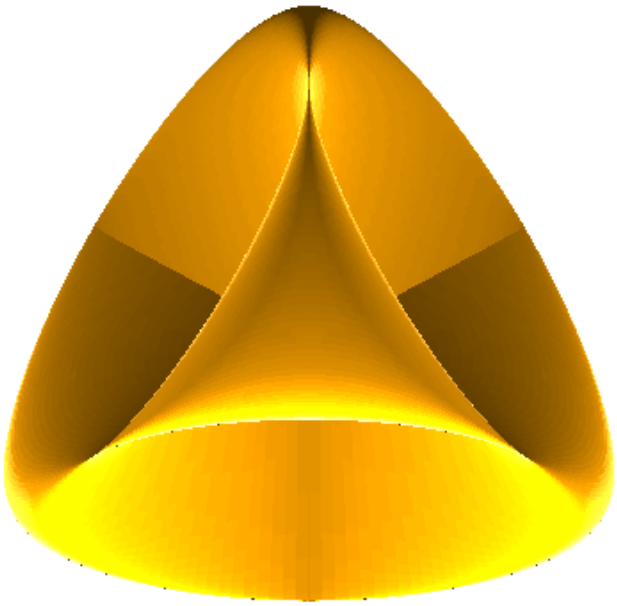
⚠ No variable named *File* must exist anywhere in the calling level of animaGIF() (and in the upper callers, if any).

Examples

Using animaGIF() to bundle existing images files:

```
Dir = TMPDIR + "/animaGIF/";
mkdir(Dir);

// Example of images files selection with a mask:
outgif = Dir + "romanSurf.gif";
animaGIF(animaGIFpath()+"tests/images/romanSurf_*.png", outgif, 300, 3);

// Play the animated GIF:
winopen(outgif);        // in a dedicated image viewer
animaGIFinHTML(outgif); // .. and in our web browser. 3 revolutions are expected.
```

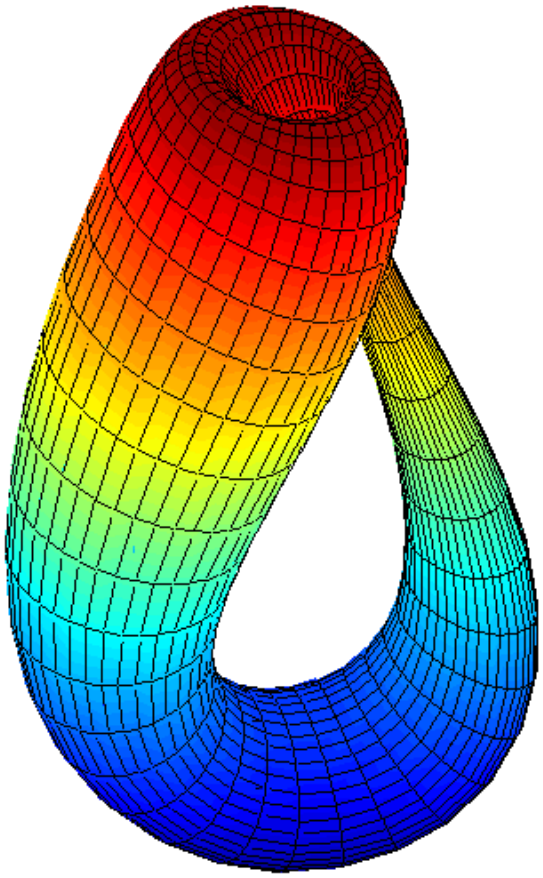
The Scilab help browser ignores the loops number and iterates indefinitely. Any web browser loops as expected.



Using animaGIF() on-the-fly with a live figure:

```
// Creation of the surface data
u = linspace(0,%pi,40);
v = linspace(0,2*%pi,40);
[u,v] = meshgrid(u,v);
k = 0.10;
r = k*(2 + sin(2*u) + sin(4*u)/2);
x = sin(2*u)/2 - sin(4*u)/4 + r.*cos(v).*cos(u-sin(2*u))*1.5;
y = cos(2*u) - r.*cos(v).*sin(u-sin(2*u));
z = r.*sin(v);

// Creation of the figure
f = scf();
```

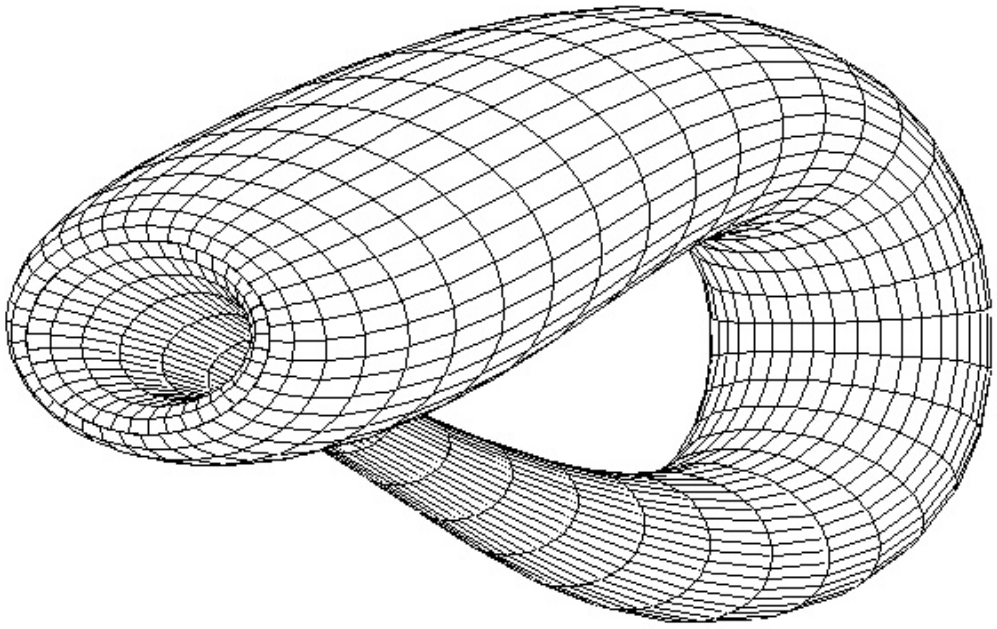
```
drawlater
f.color_map = jetcolormap(64);
f.axes_size = [400 630];
surf(x, z, y)
xtitle("","","","")
a = gca();
set(a, "box","off", "axes_visible","off", "rotation_angles",[75 80]);
a.margins = [1 1 1 1]*0;
a.children.color_flag = 3;
drawnow

// Creation of the animated GIF
// ----------------------------
// Creation and configuration of the GIF stream (default delay, 2 iterations)
outgif = "TMPDIR/anima.gif";
idGif = animaGIF(gcf(), outgif, ,2);
// Animation of the figure
n = 36;
b = waitbar(1/n);
for i = 2:n
    gca().rotation_angles(2) = gca().rotation_angles(2) + 360/n;
    idGif = animaGIF(gcf(), idGif); // Adds the current figure to the GIF stream
    waitbar(i/n,b);
end
animaGIF(idGif);    // Closes the GIF stream
close(b)            // Closes the progression bar

// Play the animated GIF:
winopen(outgif);
animaGIFinHTML(outgif); // .. 2 revolutions expected
```



A **mixed series of png, jpg and bmp images files** can be processed.

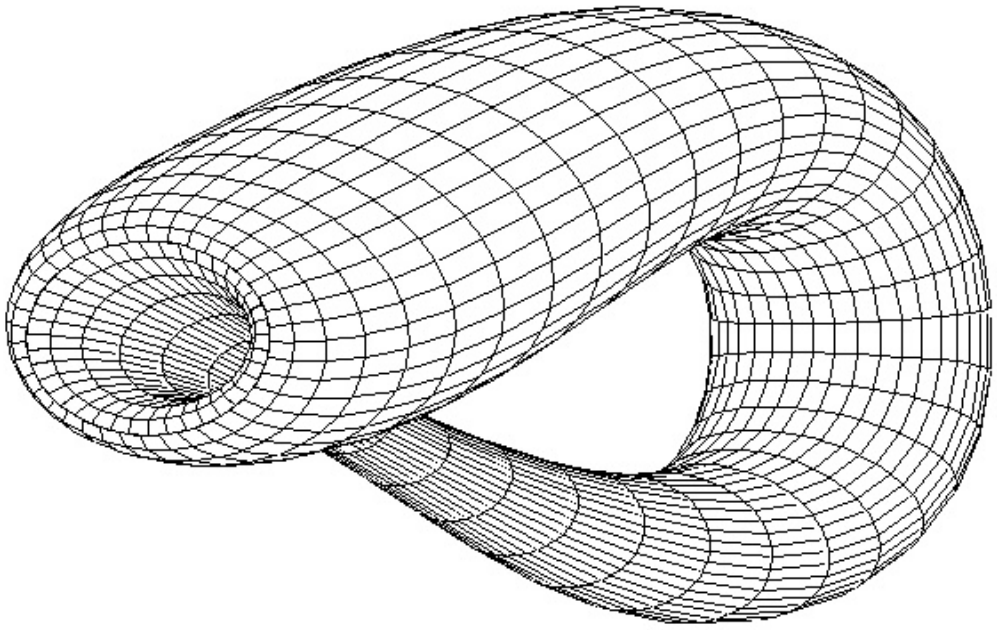
The first image of the animation sets the standard size of the whole animated GIF. If some forthcoming images are smaller, some margins are left and remain. In the opposite, wider images are clipped (in web browsers. In image viewer softwares, this is software-dependent).

```
Dir = TMPDIR + "/animaGIF/"; mkdir(Dir);
outgif = Dir + "animix.gif";
apath = animaGIFpath();
L1 = listfiles(apath+"tests/images/animaMesh_*.jpg");
L2 = listfiles(apath+"tests/images/romanSurf_*.png");
L3 = listfiles(apath+"tests/images/animaV_*.bmp");
L = [L1(1:$/3) ; L2(2*$/3:$) ; L3];
size(L)
animaGIF(L, outgif, 300, 3);
animaGIFinHTML(outgif); // in the web browser
winopen(outgif);        // in the dedicated image viewer software
```

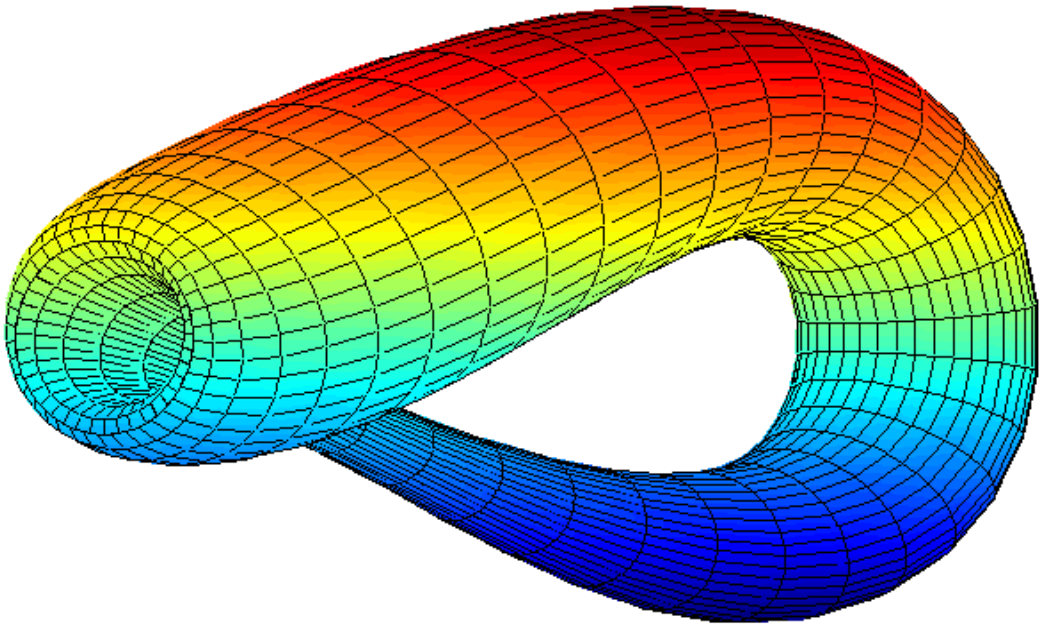


A series of numbered files can be **automatically resorted in numerical order**, instead of alphabetical order:

```
Dir = TMPDIR + "/animaGIF/"; mkdir(Dir);
outgif = Dir + "animix.gif";
apath = animaGIFpath();
L1 = listfiles(apath+"tests/images/animaMesh_*.jpg");
L2 = listfiles(apath+"tests/images/animaH_*.png");
L = [L1(1:2:$) ; L2(2:2:$)];

messagebox(strsubst(L,apath,"./"))

// AS IS, this list of images files
//  - Define 2 subsequences, one with the mesh, then the colored one.
//  - Each sequence goes twice faster, since one angle step over 2
//     is missed (20°/step).
animaGIF(L, outgif, 300, 3);
animaGIFinHTML(outgif); // in the web browser
```



```
// Now we force numerical ordering of files,
// according to the only index present in filenames:
L = ["*"; L];
animaGIF(L, outgif, 300, 3);
animaGIFinHTML(outgif); // in the web browser
```



Author

- Samuel GOUGEON

See also

- xs2png
- xs2jpg
- xs2bmp