Image processing with Gimp: part 6

GIMPRESSIONS

SIMON BUDIG

Gimpressionist is a flexible plugin by Vidar Madsen which enables the user to turn ordinary images into works of art. Linux Magazine takes a closer look at this handy tool.

One of the classic applications in image processing is that of adding a hand-painted touch to photos. Every program has at least one function for making photos into oil paintings or suchlike. Gimp is no exception to the rule – the relevant plugin can be found under <Image>/Filters/Artistic/Oilify....

Now you may be wondering why we don't sound especially enthusiastic. There is a simple reason for this: It's just boring. Great, I can turn my picture into an oil painting – but anyone can do that with any old program. The results from Gimp's filter are not particularly wonderful either. The true reason for my boredom, though, is that there is something a great deal more exciting.

Curtain up for Gimpressionist

In Gimpressionist Vidar Madsen has written a plugin that can be used very flexibly to turn images into works of art. The basic idea is simple: The image is reassembled from small paintbrush images, which can adapt to the image.

Gimpressionist is one of the plugins which have so many parameters that it is necessary to save any especially effective combinations so as to be able to load them in again later. Start <Image>/Filters/Artistic/GIMPressionist; these delay settings are the first thing you see (Figure 1). To make use of them, click on the name of the combination – for example *Dotify* and then on *Apply*. Now the default settings are distributed over the individual index cards.

Since Gimpressionist does not work especially quickly, you should ask for a preview, by clicking on *Update*. In the little box, you can roughly assess the effect. With a click on *OK* the image is then edited.

The default setting *Dotify* has the effect of making the image look like a confetti mosaic (Figure 2). In order to show which ideas are behind Gimpressionist, we will now convert the effect step by step into a sort of chalk drawing on a wall.

The wall is a sort of structure, onto which the whole image is to be stamped. The setting for this can be found on the tab *Paper*. Select the "paper" *bricks2.pgm*. To be able to see the effect (this was not required for the confetti), drag the sliding controller *Relief* up to about 70. The structure of the wall becomes visible after a click on *Updtae*.

On the card tab *Brush* you will now see where the round basic pattern for the confetti came from. We would rather have chalk strokes, so select the paintbrush *chalk01.pgm*. If you have the preview drawn again, the preview image is black apart from the structure of the wall. The

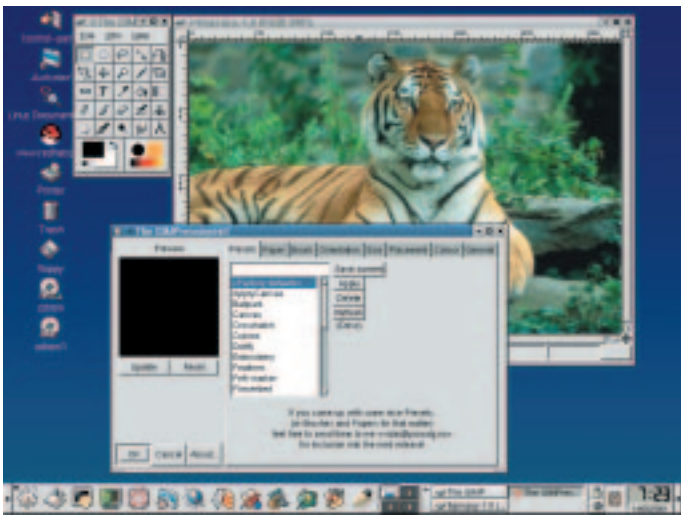

Figure 1: Gimpressionist – the starting place

Figure 2: Paper tiger

paintbrush is drawn a bit too small. So change to the *Size* tab and set the minimum and maximum size to 30. In the preview, the image becomes visible again. Obviously it is not desirable for all the strokes to run in the same direction.

In order to make up for this, change to the index card *Orientation* and under *Start angle* select 360 degrees and under *Directions* about 10. Now Gimpressionist can turn the paintbrush from the starting angle in 10 steps by 360 degrees and adapt the image. This adaptation can function according to various criteria, and one nice option is that of adapting the rough contours in the image (*Adaptative*). Obviously Gimpressionist can adapt the paintbrush strokes all the more precisely, the more *Directions* there are available.

Now you have probably noticed the large black marks, which are due to the Random *Placement* strategy. In the case of Evenly distributed, the black marks are less common.

And the size can also be adapted to the image. If, on the *Size* index card you set the minimum size to 15, the maximum size to 30 and (similar to the directions) set the number of *sizes* to 10, Gimpressionist has various sizes of paintbrush at its disposal. Here, too, it is possible to define a selection strategy. *Adaptative* orientates itself to the structures in the image.

Be aware that for many directions and many sizes the computing time rises steeply and the result could keep you waiting a long time. You should avoid combining the maximum setting of 30 in the sizes with the *Adaptative* strategy.

Now we have a tiger, which has been painted onto the wall with chalk (Figure 3). In order to get a feel for the options, you should try out the various default settings and then see what settings have been made in order to achieve this effect. With a bit of creativity it is possible to turn an image not only into an oil painting, but also into other – jollier – things.

Animation

In part 3 of the Workshop we asked you to submit suggested topics for a continuation of the series (to *sbudig@linux-user.de*). The most frequent question was how to create GIF animations with Gimp – or somewhat more generally – how Gimp can cope with films.

To answer the last question first: Gimp is not the right tool for editing films of several minutes in length. You can certainly read in films using the plugins under *<Image>/Video/Split video into frames*, but they are then stored on the hard disk as individual images uncompressed – anyone who doesn't exactly have terabytes of space to spare will very soon run out.

When it comes to making more complicated animations, you will notice that Gimp is not the ideal tool. The method of showing individual images as layers is a fast hack, to enable GIF animations without overwhelming the internal data structures. In particular, one now no longer has the option of working with layers for animations. Especially in animations, where objects are moved back and forth and may thereby overlap each other, this would be very useful, though. This is where the plugins from the domain of *<Image>/Video/...* come in. But even with the classic methods it is possible to make GIF animations with lots of effects.

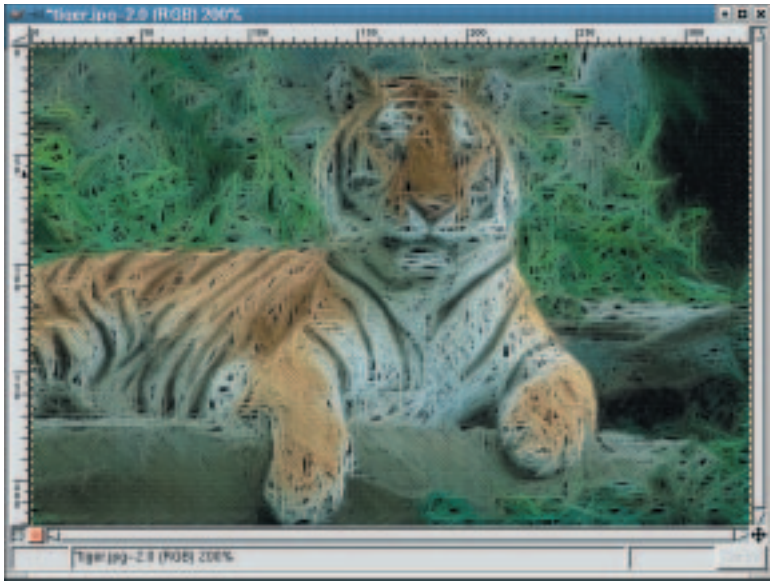

Figure 3: Tiger on the wall

GIF in motion

GIF is still the only format supported by almost all Web browsers for animations. This is why, despite the licensing problems concerning the LZW compression, it is still popular for use in Web design. You have to have a licence from Unisys to be able to legally distribute GIF images produced with Gimp on the Internet. For this reason, in many Gimp packages the GIF (and TIFF) plugin is not installed as standard. You must then install another package (gimp1.2-nonfree) to be able to create GIF images. The same applies for the Windows version of Gimp, and you can find out more on this at *http://www.gimp.org/win32/*.

Animations are, in reality, nothing more than a collection of images shown one after the other. To this extent it seems obvious that Gimp should save the individual images of an animation in layers. If you load any old animated GIF from the Web into Gimp (you can simply give a URL as filename, Gimp then downloads the image using *wget* from the Net), you will see that in the layer dialog the individual stages of the animation are visible.

How long an individual image is visible, is something you can determine from the name of a layer. If it is called *layer (500ms)*, the image will be shown for half a second. Bear in mind that only *ms* (milliseconds) is permitted as a unit. If you do not specify a period, when the finished image is saved a standard period will be asked for..

Figure 4: Giiiiimp!

A big yawn

One fast method of creating an animation is the Iwarp plugin, which you will find under *<Image>/Filters/Blur/Iwarp*. With this plugin you can blur images freehand, in a similar to that of the Goo programs. It is especially easy to create caricatures out of faces, by exaggerating distinctive facial features. You can displace image areas, blow them up, shrink them and turn them clockwise and anticlockwise. With *Delete* you take an image back to its original condition. The two sliding controls define the size of the affected area and the intensity of the effect. With the mouse you can then blur in the preview image, and you will get the hang of it after a few tries.

The Iwarp plugin now has the option of creating an animation from unblurred to blurred image (and back again if you like). To do this, simply click on the index card *Animation* and select the number of the intermediate steps. With *Reverse* the animation goes, not from the original to the blurred image but – surprise – the other way round. With ping-pong, after blurring it animates back to the original image.

Using the tiger image from last month we have made our tiger yawn using this method (Figure 4). Maybe you can even get your mother-in-law to grin...

You can view the finished work of art using *<Image>Filters/Animation/Animation playback*. The plugin is easy to use, but there is also another neat trick here: You can click on the display of the animation and drag it out of the window. This is especially practical, if you want to quickly assess how the finished animation would look in the website, and do not feel like faffing around in HTML code. Simply drag the animation over the Web browser. It disappears again when the window is closed.

Keep the *Layers & Channels* dialog open all the time, when you are working with animations. It is a very useful tool, to quickly duplicate a layer, change the sequence of the layers (thus the order of the individual images) and to combine two layers with each other. As a little example we can make a little text appear.

Fading in text

Create a new image 500x100 in size. Select the text tool and create the text which is to appear, in a layer of its own. Now duplicate the background and the text layer 10 times each. Using drag and drop you can now sort the copies of the text layer between the copies of the background layers (Figure 5).

When you play the animation back, you will see a flashing white text against a black background. But we would rather have a text which is faded in, so select values of between 0% and 100% as covering power of the text layers, using the slide control in the layer dialog in 10%-steps. For technical reasons, you cannot see this effect yet in the animation preview, but in principle it is still a flashing text. To get rid of this, merge every two sequential layers. To do this you must click the mouse to activate the text layers network after another and press Shift+Ctrl+M to trigger the command *Combine downwards* (Figure 6). Now our text fades in gently.

We can now save this image as an animated GIF. Simply specify a filename, ending in *.gif*. The export dialog will appear automatically (Figure 7), which informs you that several layers can be combined before saving – but we don't want to do that in this case. So click on *Save as animation*. Since the GIF plugin only supports indexed colours, the image is automatically converted into such a format. Then click on *Export*. In the dialog which will appear, just

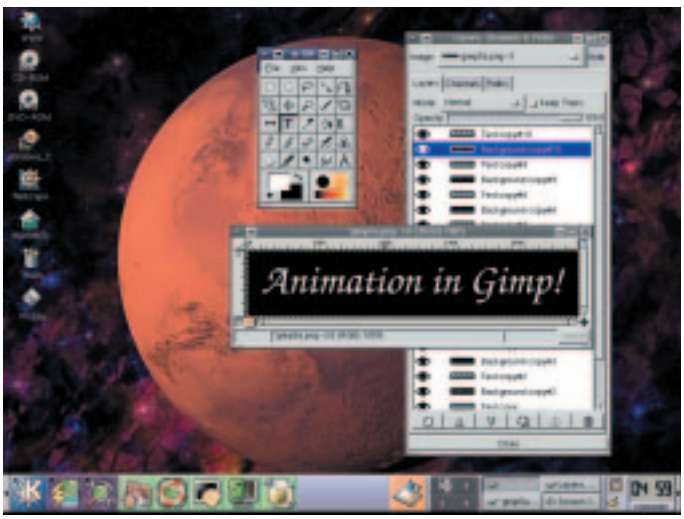

Figure 5: Sorting the layers

Figure 6: Merging layers

click on *OK*, the default settings are reasonable. If you now look at this image in Netscape, you will see the effect.

Web design

GIF animations are sometimes a nice enhancement for a website – but if they are used to excess and there is something flashing and moving wherever you look, visitors will be put off. Please be sparing with the use of GIF animations. Sometimes a small effect is much more effective than all that flashing. For example on Slashdot after a report on Gimp there are always astonished comments that the eyes of Wilber (the Gimp mascot) can move – and yet one had never noticed it before. They do actually move by one or two pixels, and that will never change. But a Wilber who rotates about his own axis, who changes his colour and at the same time hops up and down, would never trigger this Aha! affect.

Size matters

The other thing you should bear in mind is the size of files. GIF animations can become very large and drastically increase the loading time for a Web page. If the animation is better designed right from the start and some effects are made slightly differently, you can save a lot of space. The animation we have just created (Figure 8), is about 34 KB in size, via an ISDN connection it would take five seconds to get onto your home PC. But since it is only one second long, it will run too slowly and be jumpy.

In order to reduce the size, you should do two things: Firstly, index the image by hand (as few colours as possible and if possible without colour scanning) and then select the menu item *<Image>/Filters/Animation/Animation optimize*. This command tries to remove redundancies from the image and to exploit a couple of special features of the GIF format, in order to save a bit more space. This is especially worthwhile when large areas remain the same from one image to the next and therefore do not have to be saved again. In our case, we have got it down to about 24KB.

Figure 8: Fading in text – large files

But the main problem with file size is home-made. Since our text fades in slowly, many pixels change colour from one image to the next. It would be better if only small areas were to change each time – and then less image data would be transferred, too. As an experiment I have redesigned our animation so that it appears letter by letter. This means that only a small area ever changes from one image to the next. If you want to do this, it is worthwhile starting with the full text, making a copy of the layer and then deleting one letter. Repeat this process until the text is blank. The result looks something like Figure 9.

Figure 9: Fading in text – small files

Once the image has been indexed and optimised it is just 4KB in size. So the slight adjustment to the animation has certainly paid off. Generally it is possible to say that movements and fading in take up more space than the appearance of parts of an image.

Obviously there is a lot more to discover, but that's enough for this time. Have fun! ■

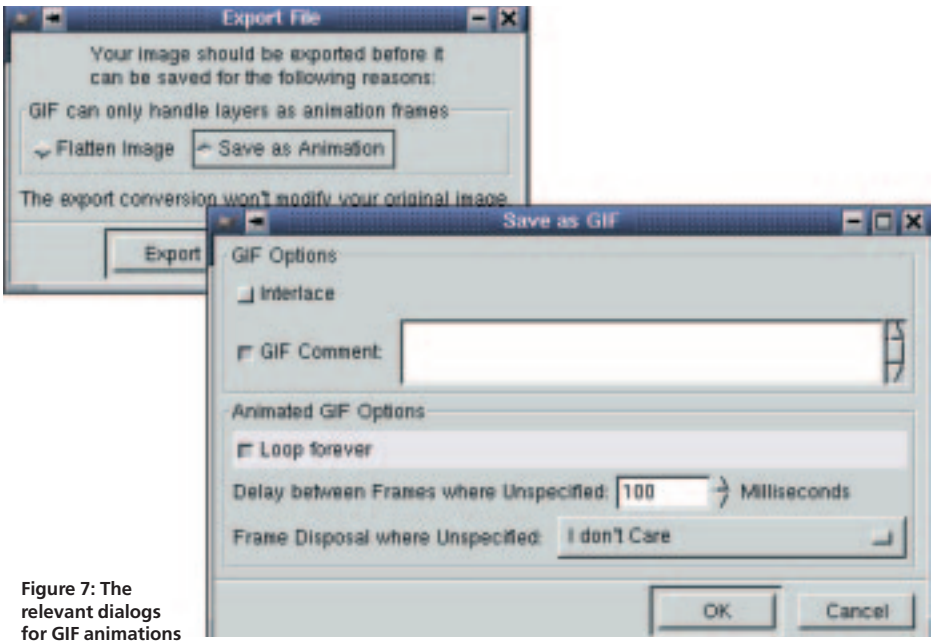

Figure 7: The relevant dialogs for GIF animations

The author

Simon Budig is now battling with compiler construction. That's why there are no philosophical comments this time. *A parser is an algorithm, formal proof of...*