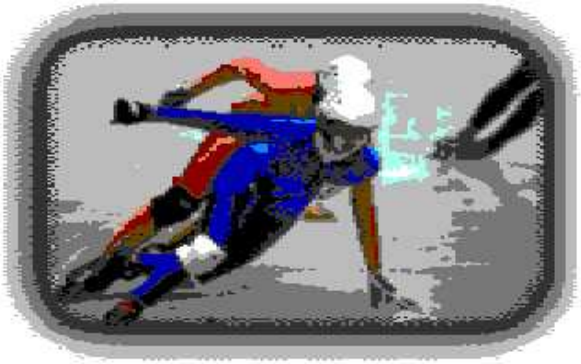
How to GoDot

By Arndt Dettke

Working on clips (Part 3, Overlaying Images)

GoDot provides three different methods of combining images: overlaying, masking, and alpha-channeling. This issue will cover how to use the overlaying method using images with transparent areas in them. And what nice should we create today? Pictures which leave the all-time path of rectangularity, wouldn't that be nice? Look here.

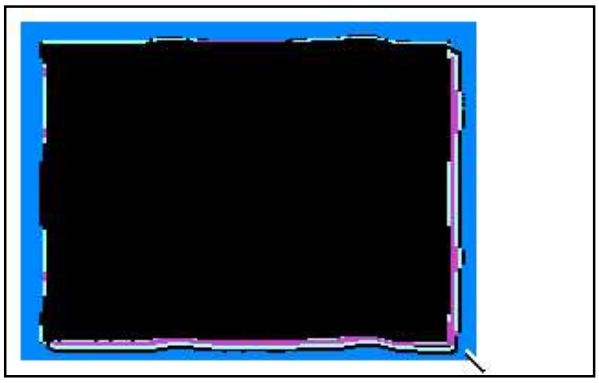


These are the modules we use in this issue of our workshop: mod.ClipWorks, mod.StretchClip, mod.Squeeze2Clip, mod..Histogram, svr.-4BitGoDot, and ldr.4BitGoDot.

I prepared some useful, and I hope uncommon frames images for you. I took them from a PC alpha-channel archive and converted them to GIF, so GoDot could easily import them to its own 4bit format. You can download these borders from my site.

Most of the provided border images are not in a ready-to-use condition. I deemed it better to let them be imperfect. Leaves us things to learn about… For instance image "paper2.4bt": load it and render it. You'll see that it is much smaller than the screen. To take full advantage of it, we will scale it up to screen size now.

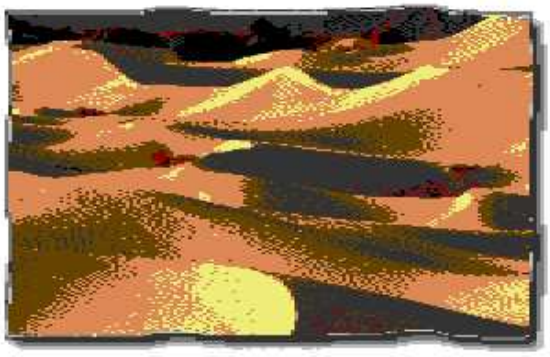
For this, execute **ClipWorks** and click on "***Clip***" to visually set the clip values. Click off the upper left corner of the black "hole" on screen and move the mouse pointer to the bottom right corner. You'll see that GoDot marks the area of the clip (all colors but the background color get inverted). This way it is easier to match the correct position (one tile off the corner). After clicking, the ClipWorks requester reappears and should show the values 1, 1, 31, 23. Edit the values manually if they don't match. "***Accept***" and leave ClipWorks.



Install **mod.StretchClip** and execute it. It doesn't open another window but immediately starts working which you can see by the white bar wandering down the preview window. StretchClip is finished when the bar disappears.

"***Display***" (render) the image to see what has happened. The black hole now covers the whole image leaving only a small white border on all sides. Save this image as a 4Bit file. If you are owner of an REU, I'd recommend you save it to "4Bit Undo" on unit RAM. Saving the frame image, you have repeatable access to it for many further images to process.

These were the preps, step into the art of image processing now. Start loader **4BitGoDot** and enter "***Compose***" mode. Click on "***Background***" thus to cause GoDot to load the next image only to those places in the current image (the black hole) which are black. So, black is transparent now!



Load any 4bit image you want and render it to see

how it looks. You'll be happily astonished! And the best is (you see it on these pages) when you print these images: they have a new kind of border, they aren't strictly rectangular anymore!

Reload the black hole image and try for more. You'll like it!

Ahead we go for the next frame image. Load "papersheet1.4bt" and render it. Hm, looks nice, but something is wrong with it. Wait… Yes, there's no "black hole"! We have no decent transparency area! You guess why that is the case? Yes, I intended it. Of course. Come, follow me to analyze the image and find a solution.

Install **mod..Histogram** and execute it. The first you see is it's working. When done it presents you with the amount of pixels of every one color in the image. You recognize it has only five colors in it, and these are all the C=64's preset grays. (Every gray scale GIF gets converted to these colors by ldr.GIF.) Now, to invert the image we just have to exchange black and white and gray1 and gray3. Click on "***Swap***" to do so. First pass: black and white, second pass the other two. "***Exit***" Histogram and render the image. We have what we needed and can continue like we did with the first frame images.

Operate accordingly on the third frame image "oval.4bt". Results could look like this one here (showing my daughter and my older dog):

But what if I wanted to add a "sheet" of paper to my image instead of framing it? Good question and easily answered. Load your image and determine where to insert the paper (set a clip there). Then load the paper image (not a "black hole"), scale it up to screen size and then apply **mod.Squeeze2Clip** to it. This will scale the screen down to the clip's size. At last, you "***Compose***" the wanted image as a "***Background***". And that's it.

Mod.StretchClip – Module which magnifies every chosen clip to fullscreen size. To keep the resulting image undistorted you should be sure to set a clip with an aspect ratio not far from 8/5 (or 1.6), which is the aspect ratio of the C=64 screen (40/25=1.6).

Mod.Squeeze2Clip – This module scales a whole screen down to the clip size. To achieve an undistorted result care for an aspect ratio of 8/5 when setting the clip.

Mod..Histogram – A very powerful module to affect pixels of one color as a whole. First, it gives you information (amount of pixels of a certain color), then, you can completely exchange two colors. Finally, Histogram lets you join one or more colors with a selectable target color.

Command history

Scaling up (the image is already in memory)
Inst: ClipWorks
Execute
Clip row, column, width, height
Leave
Inst: Stretchclip
Execute

Scaling down (image fills the screen area)
(just change the second but last line above to:)
Inst: Squeeze2Clip

Inverting colors (in a gray scale image)
Inst: .Histogram
Swap blk, wht
Swap gr1, gr3
Exit

Framing an Image (frame already in memory)
Load: 4BitGoDot
Compose Background “imagetoframe.4bt”

The option “Foreground” in ldr.4BitGoDot is useful if you want to apply the frame as a *final* step of processing the image. Next time we’ll cover the masking method of combining two images.

Again I updated two GoDot modules (download from my site www.godot64.de): mod.Squeeze2Clip (bugfix) and mod.Scroll (bugfix). Also, the frame images used in this issue of HtG are for download there. Looking forward to see you visiting me!

Have fun using GoDot!