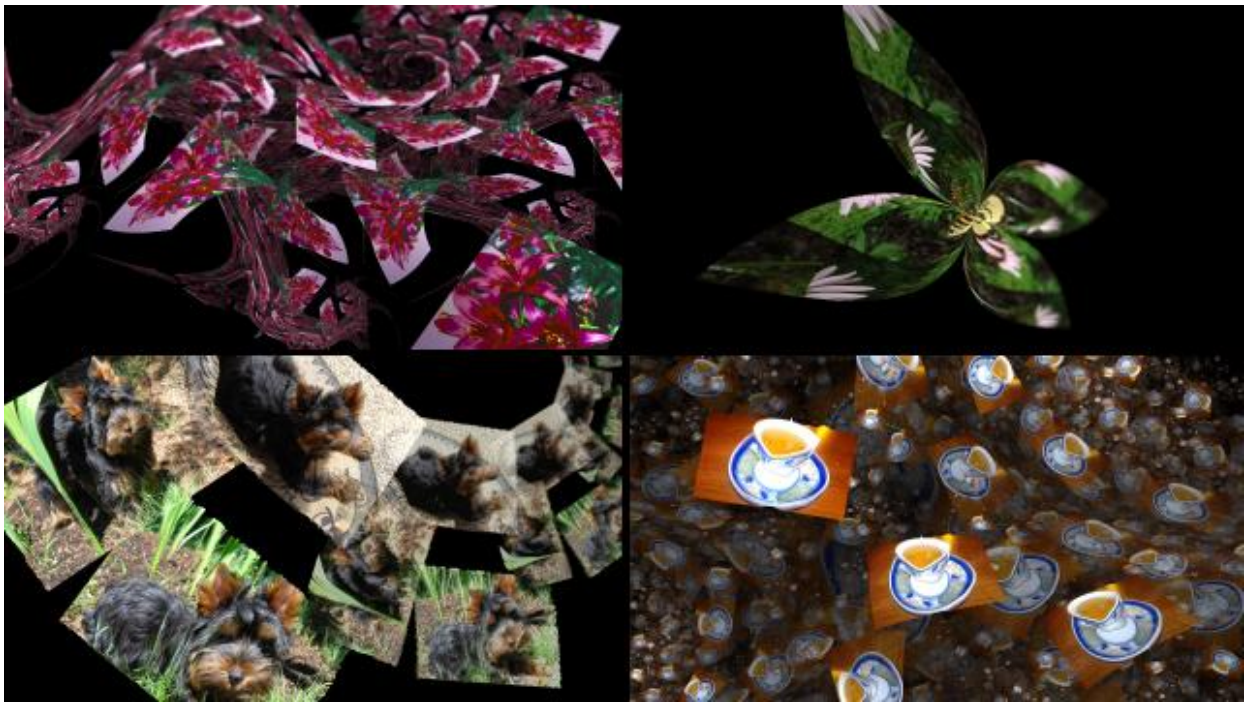


JWildfire tutorial on: image mapping on fractal flames - using the (post_)colormap_wf variation

by thargor6 (http://thargor6.deviantart.com/)

Version 0.1

The following tutorial describes how to create amazing - and refreshingly different looking - fractal flames by applying bitmap images to it - either as color map or bump map or both.

All you will need is a recent copy of JWildfire (Version 0.50 or higher is recommended) and some own images (*.png or *.jpg) to play with.

Table of Contents

1 Introduction

JWildfire is the spiritual successor of the image processing software *Wildfire\7 PPC* for the Amiga. So dealing with bitmap graphics is more than natural for *(J)Wildfire* because it's originally designed to do that. And if you explore the menu items and windows beside the fractal flame editor you will face lots of functions which deal with changing or generating bitmap graphics.

So it was only a question of time when I would try to combine fractal frames and bitmap graphics - the result are **post_colormap_wf** and **colormap_wf** variations and some additions to the renderer.

1.1 The post_colormap_wf and colormap_wf variations

The **postcolormap_wf** and **colormap_wf** variations are used to modify

- the absolute color (R, G, B)
- the color index
- the z-coordinate (optional)

by reading color information from an external bitmap graphic.

The difference between this variations is just in the usage of the texture coordinates.

The **post_colormap_wf** variation takes the coordinates which were affected by all of the (non-post) variations as input and projects them onto the image to retrieve color information.

The **colormap_wf** variation takes the coordinates from the affine transform ("triangle") as input, respectively.

1.2 Coordinates

If we have an image of the dimension *width* x *height* we can define pixel coordinates in the *0...width-1* x *0...height-1*.

Those coordinates are automically mapped into the range *-0.5 ... 0.5* x *-0.5 ... 0.5* of fractal flame coordinates. I. E. if you create a fractal flame which consists of a square shape of size 1.0 x 1.0 it will fit the whole image. We wll try out this example later in this tutorial.

There are additional parameters to stretch or move this texture.

1.3 Colors

As noted above the **post_colormap_wf** and **colormap_wf** variations modify both color and color index and you my ask why.

The reason for this is the process of rendering a fractal flame. Depending on the transforms used and the weights set, they are projecting themselves each to another.
I. E. there occurs a reproduction of shapes... and colors. The reproduction of colors is driven by color index and the color *symmetry* (also known as *color speed*).

The color index is chosen accordingly to the original color of the image data and the currently used color gradient. I. E., if you use a gradient which does not reflect the colors of your image very well you may get not the expected results.

Changing the gradient may improve the results in such cases dramatically.

Note that both the computed color index and the gradient in case of the **(post_)colormap_wf** variations only applies to reproduced shapes.
On the first iteration step the original colors of the image are used! This makes it possible to use any number of bitmap graphics with any colors in a fractal flame. At the first iteration all those shapes remain the original image colors. Only at further reproduction steps the colors are computed accordingly to color index and the gradient. But not in all cases it makes sense to have reproduction enabled at all.

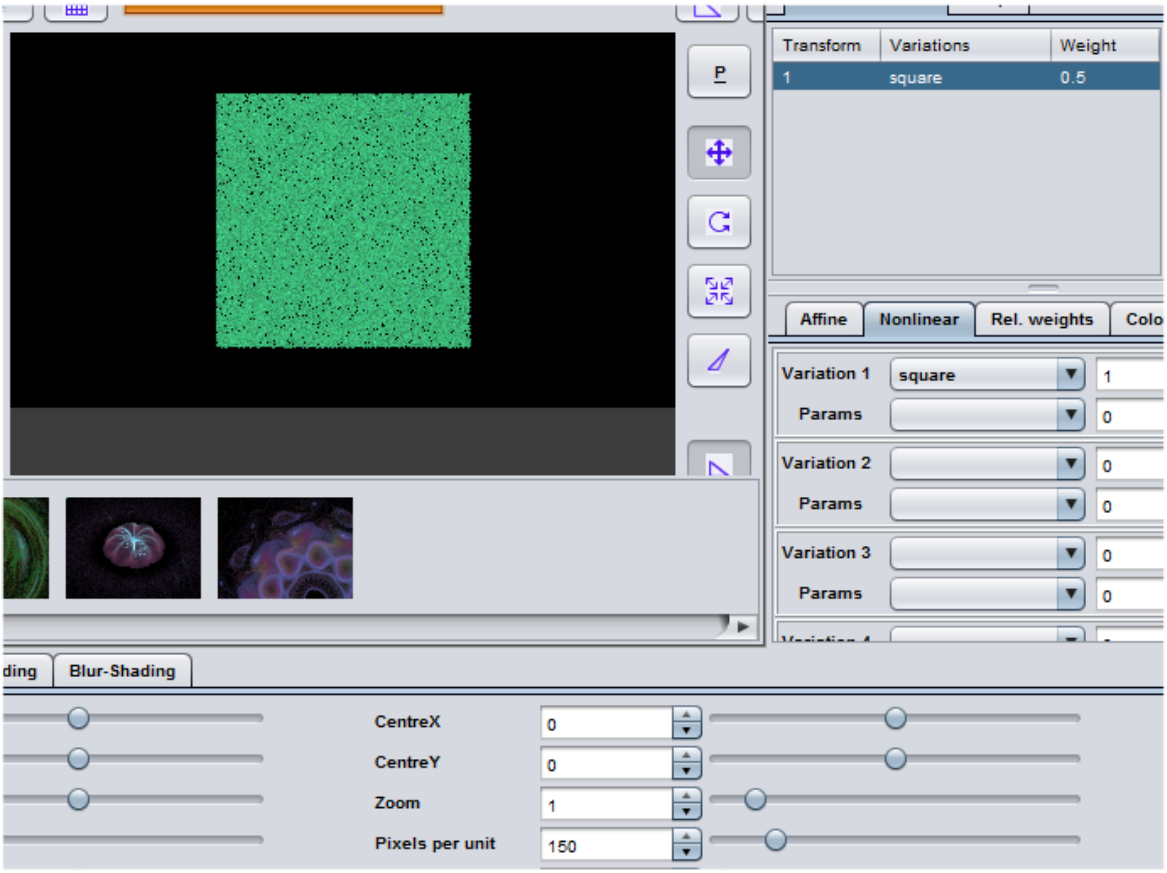
2 Examples

2.1 Getting started with a simple rectangular shape

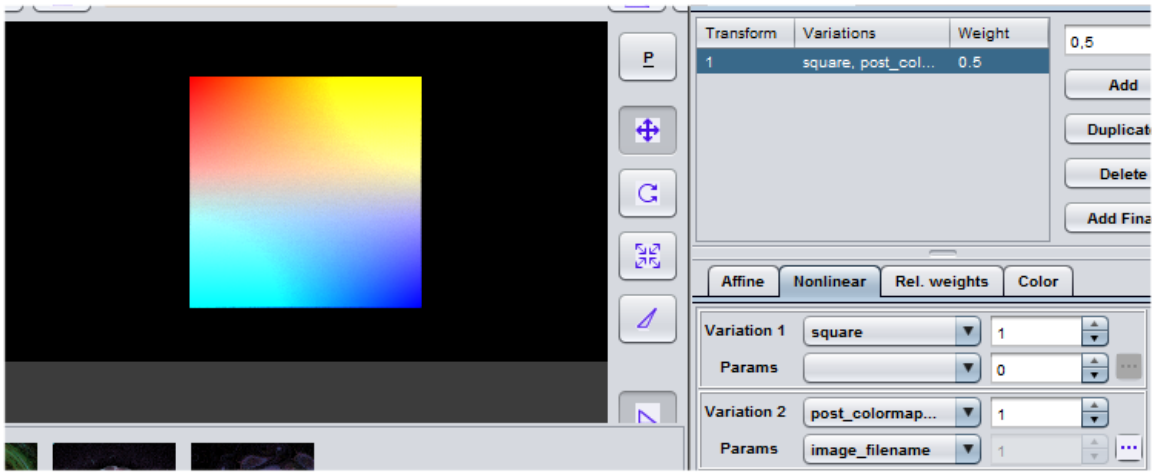
Create a new fractal by hitting the "New from scratch" button.

Change the "Variation 1" on the "Transformations/Nonlinear" tab to "square".

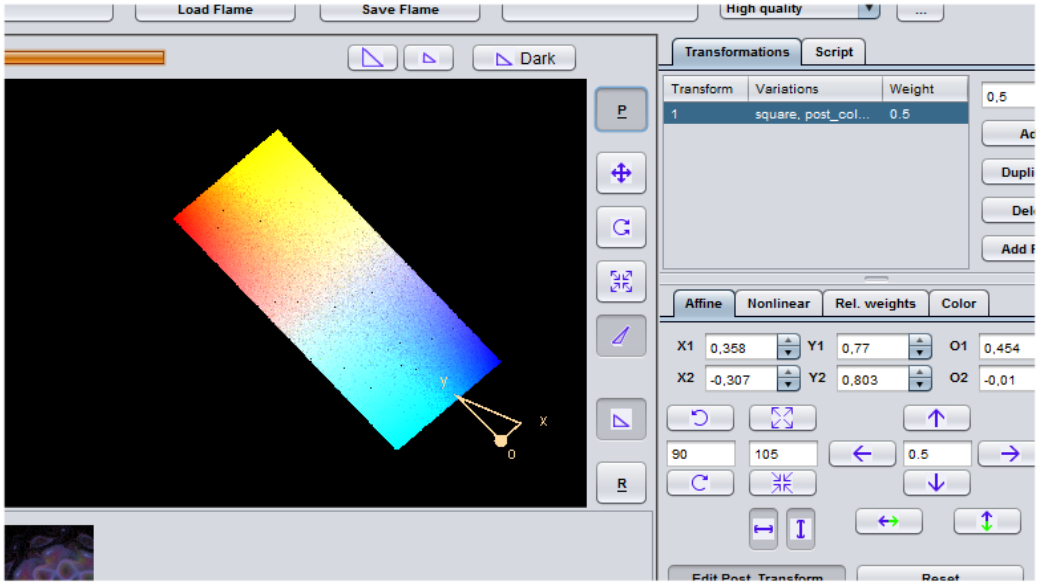
Increase "Pixels per unit" on the "Camera" tab to 150 so that the square is filling the preview area:



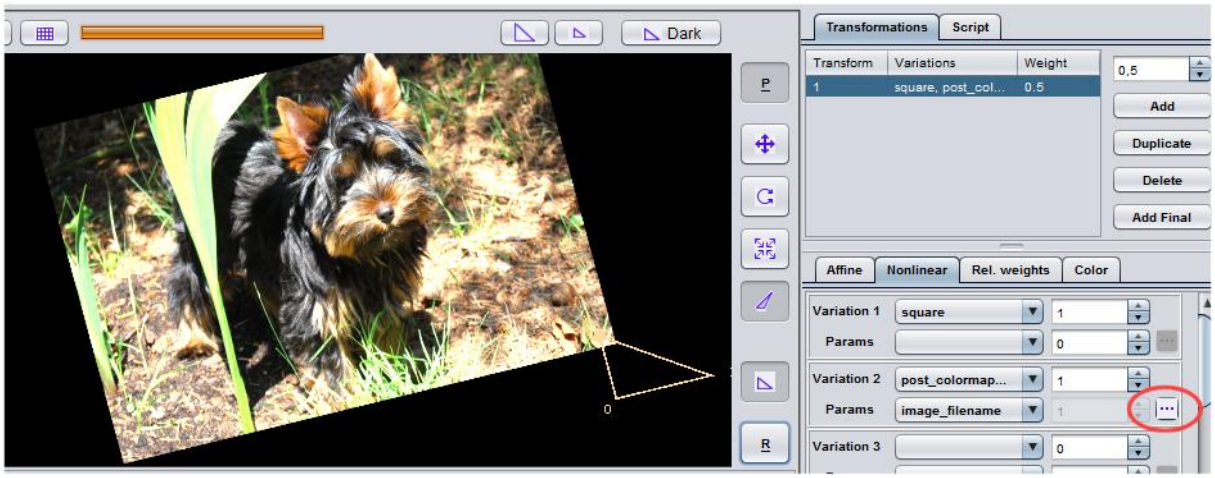
Set "Variation 2" on the "Transformations/Nonlinear" tab to "post_colormap_wf", set the value to 1.0. The rectangle is filled with the default image, a simple gradient. Note that the gradient applies with all of its colors, no matter which gradient you use:



Now enter the post transform editing mode by clicking the "P" button right of the preview area. You may freely modify the shape as usual - now with applied image texure:



Now let's use some external image, just click the "..." button right of the parameter "image_filename" of "Variation 2" on the "Transformations/Nonlinear" tab.
A file dialog appears, just select any image of your choice and keep playing with the transform controls until you are get a satisfying result, e.g.:



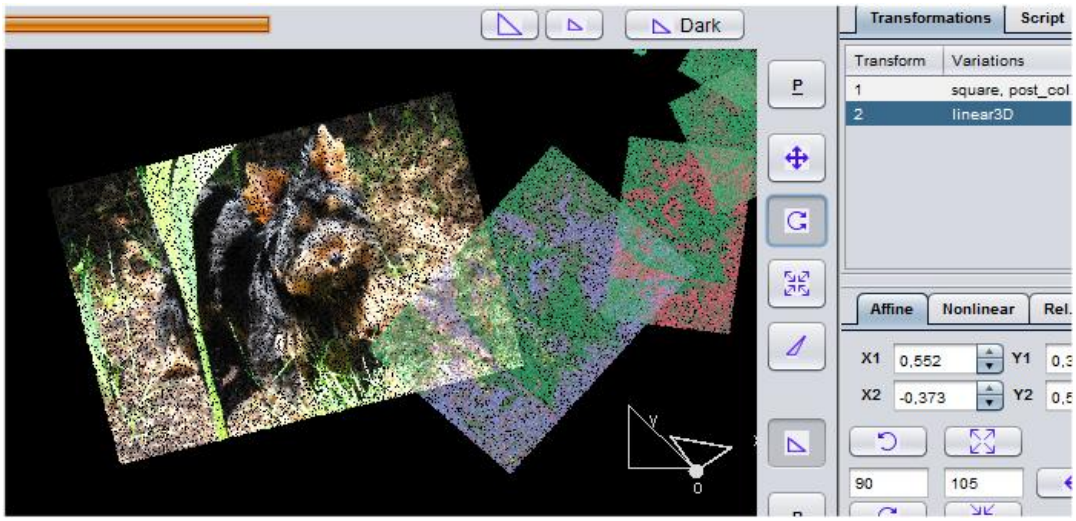
2.2 Reproducing the simple shape

Add a second transform by clicking the "Add" button on the "Transformations" tab.

Turn off the post-transform-editing mode by clicking again at the "P" button right in the preview area so that it appears "off".

Set the "Pixels per Unit" on the "Camera" tab to 150, i. E. zoom out a little bit.

Select the newly created 2nd transform and move it about unit to the right, scale it halve down and turn it a little bit so that it appears like this:

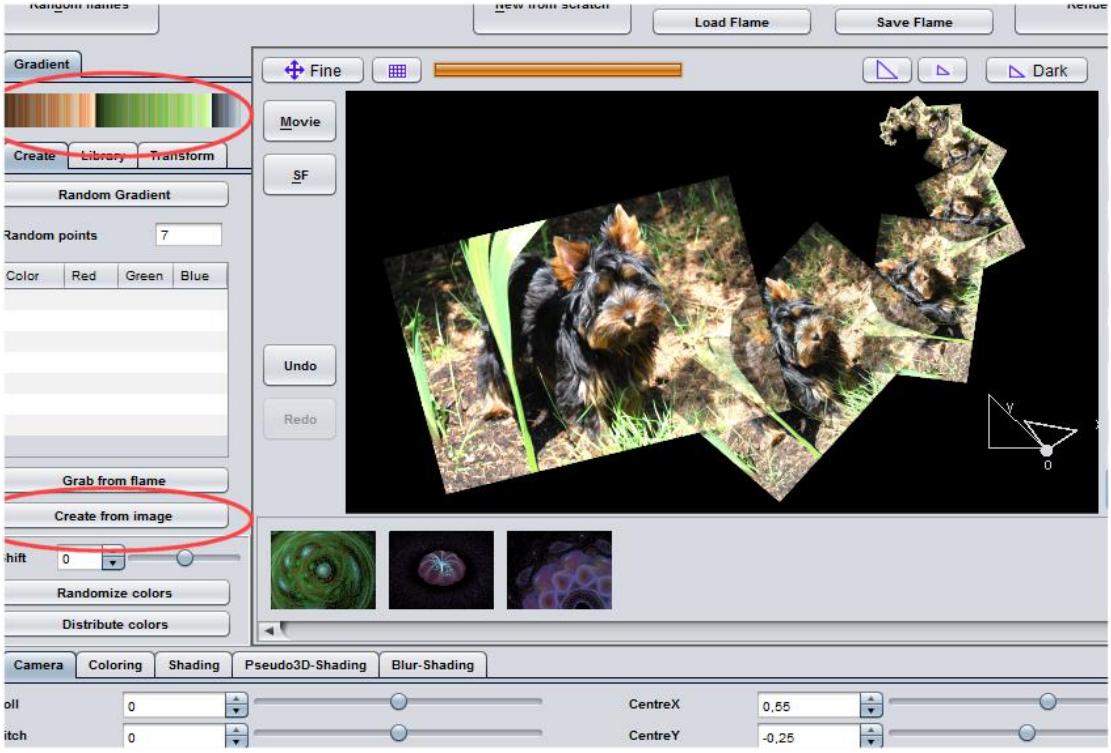


The shape now is replicated but has strange colors. Change this by changing the "Symmetry" value on the "Transformations/Color" tab to 1.0. Now JWildfire tries to find color indices fitting to the current gradient.

2.3 Generating an optimized gradient

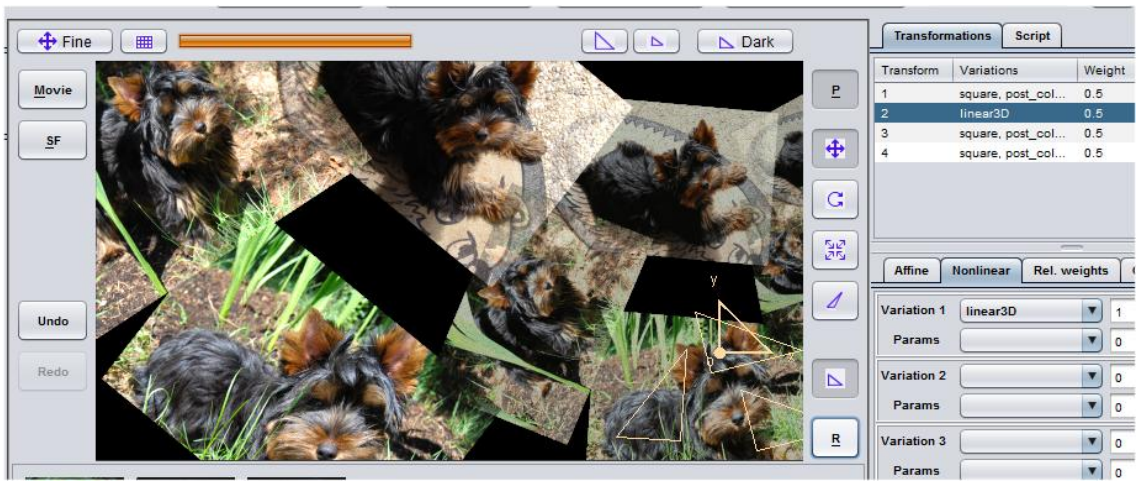
The current result may fit at this stage or may not, it just depends on the gradient.

But you can calculate a gradient from an image! Just press the "Create from image" button on the "Gradient/Create" tab and choose your image again. JWildfire creates now a new gradient by choosing the "most important" colors from this image.
You may also adjust the "CentreX" and "CentreY" to center your fractal and should now get a decent result:



2.4 Using multiple images

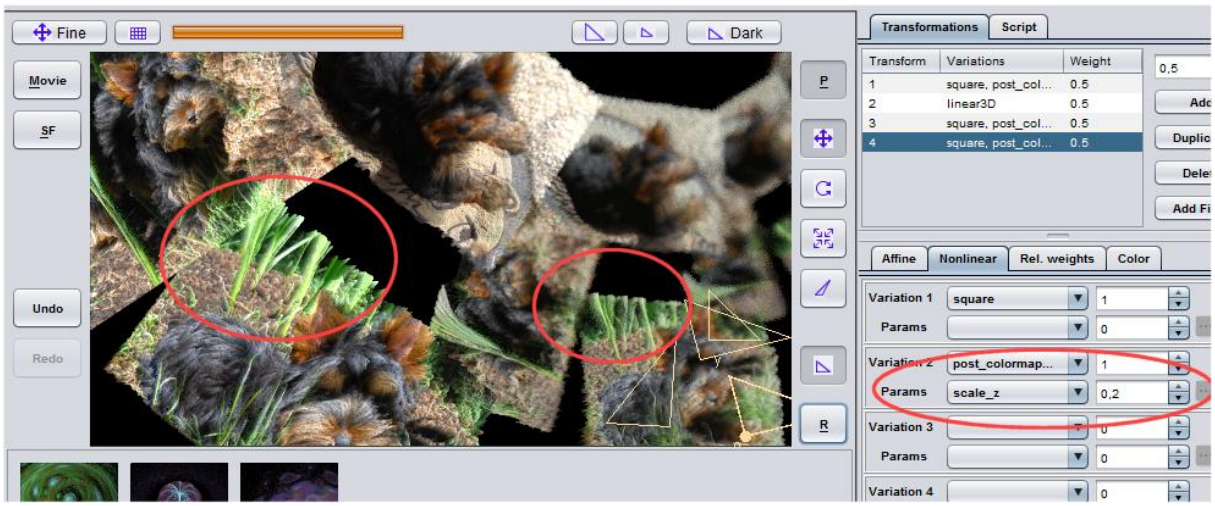
You may use as any number of images you want. In this example you should just duplicate the first transform and use different images, e.g.:



For good results the gradient should be an "average" for all used images. You may create such a gradient by just creating a larger image which contains all images uses for mapping and use "Create from image" function again

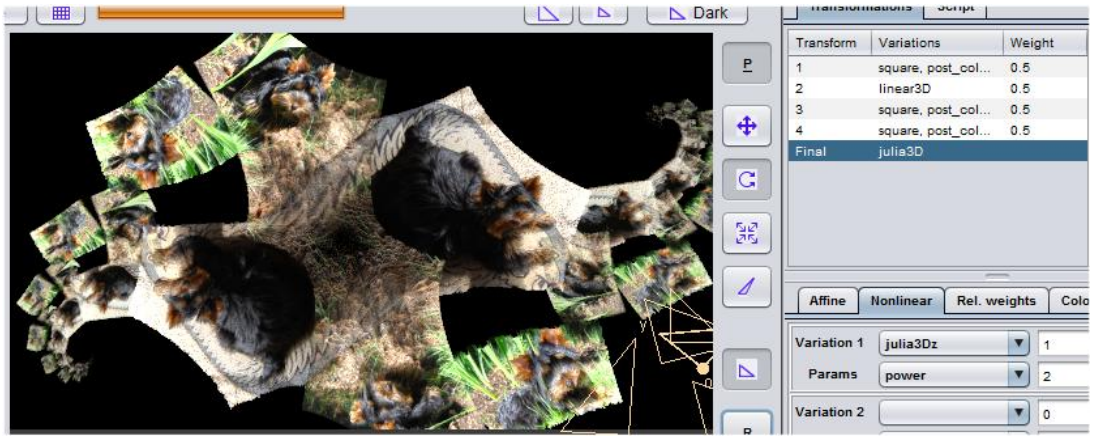
2.5 Using bump-mapping

You can also use the color information of the color map to apply additional bump-mapping. Just bring your fractal into 3rd dimension, for example by changing the pitch to 25. It should still appear to be flat. Change this by setting the "scale_z" parameter of the "Variation 2" so a value between -0.3 and 0.3. This cause the points to be moved into z-direction according to the luminosity of the image map:

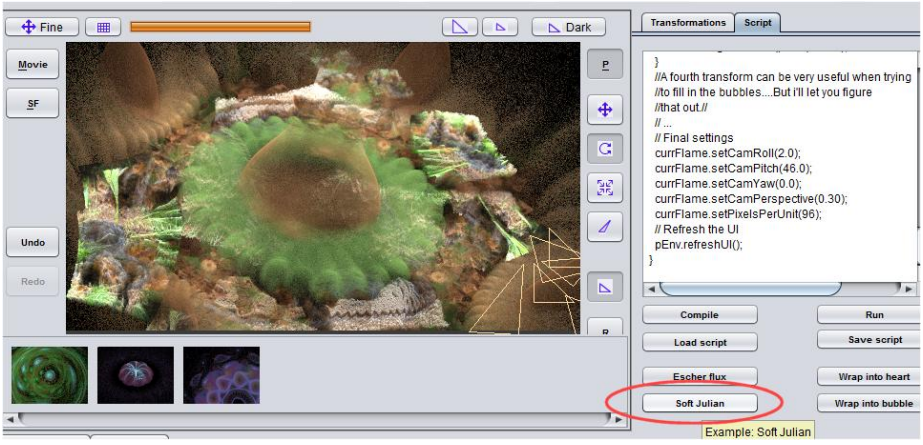


2.6 Creating more interesting shapes

You can now create more interesting shapes by using all the techniques you already know, e.g. you could apply a final transform, for example Julia3Dz:

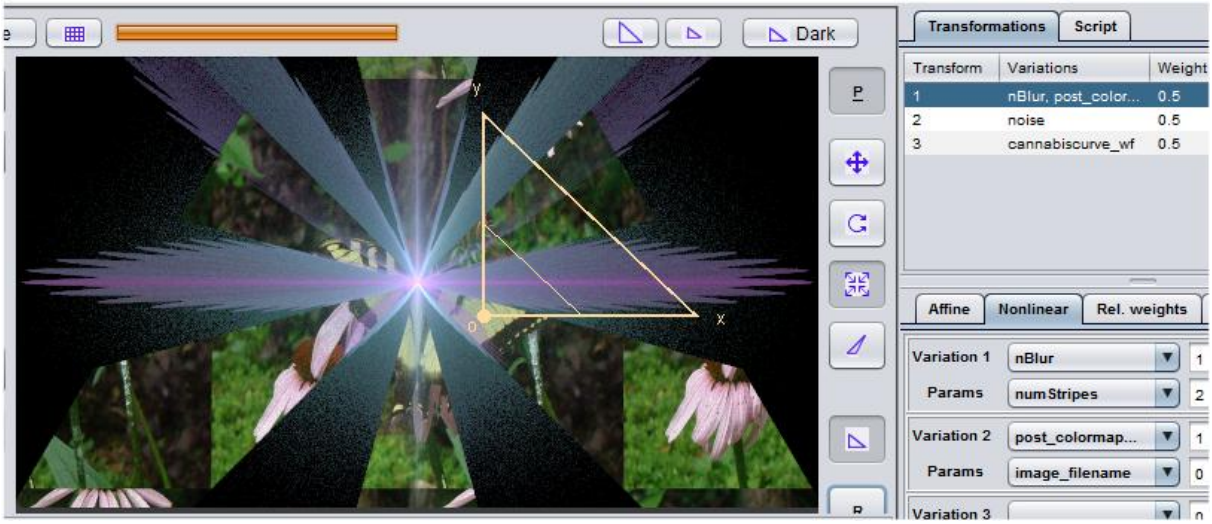


Or just apply a script to it, for example the "Soft Julian" script:



Or you could just go back to start from scratch and to create something totally different, there are no limits - you already know what now follows...

Have fun to try your own things! :-)



(The original photograph used above was supplied by Katie Shelby - thanks Katie :-) All other images were created my myself.)