aFraktal™

an Integrated Shading Network

overview

Welcome to aFraktal, an Integrated Shading Network

Why an Integrated Shading Network?

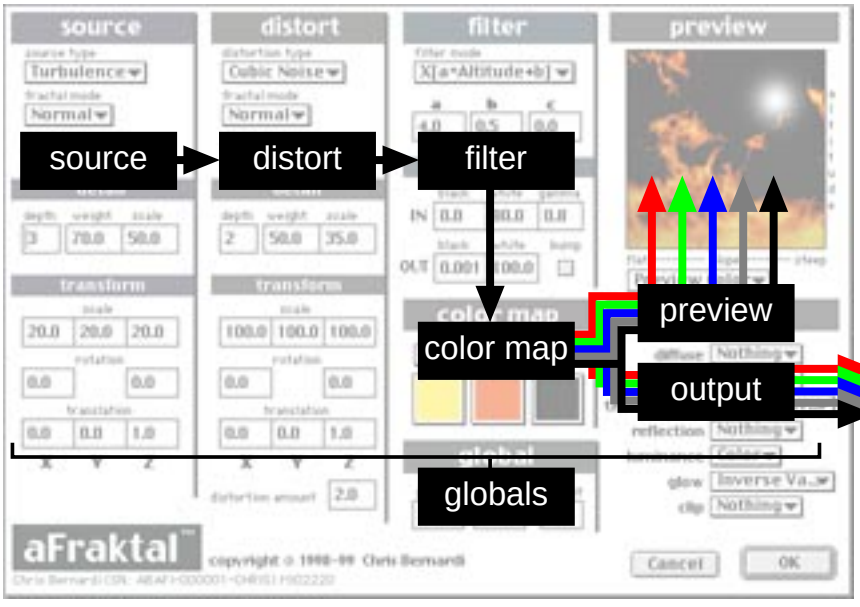
Most shaders for ElectricImage™ are very specific. The shader will do one specific thing (and often quite well). While this can be efficient, it comes at the expense of flexibility. Such shaders only work in the channels they are called up in (though they often write to channels you don't even suspect). If you need to use the same shader in another channel, you will have to copy and paste it. Calling up multiple shaders with the same settings can lead to unnecessarily long render times.

By building several shading modules together into an integrated network and combining powerful filters and flexible output routing, aFraktal takes you to the next level in shading technology. Now you can build your own functions, fine tune them with the integrated filter section and route them to as many outputs as you desire, all from a single shader! The result is an unprecedented level of control over your final image. No more hunting for the right shader, because you have it all right here in aFraktal.

Sound complicated? It's not!

While the inner workings of aFraktal are rather complicated, it's remarkably easy to use once you understand how it works. aFraktal is most easily viewed as a series of shaders, working together to generate a final image. Let's take a peek at how this works.

Source
Distort
Filter
Color Map
Preview
Output
Globals



aFraktal can be broken down into its 7 major modules:

The shading information starts at the Source and travels through each of the individual modules in order, until it reaches the output section. If you use Adobe Photoshop, you can think of the source as your starting image and the various modules as filters that modify this image.

Notice how there are only colored lines at the very end of the network? Prior to reaching the Color Map module, all information that is fed through aFraktal can be thought of as greyscale. We'll be referring to this as a Value. This will become important to remember when we get to the output section.

Let's take a look at the modules one at a time, and see how they work together to give you the final result.

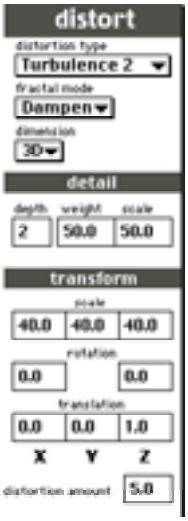
Source: Where it all begins

This is the most important module in the network. This is where you generate the fractal function that you will be modifying in the modules that follow. You have a variety of sources to choose from (11 in all) and a mix of different fractal modes (8 to choose from). This leads to 88 different base fractals you can generate, and we haven't gotten past the first 2 fields yet! You can generate your fractals in 1, 2 or 3 dimensions.

There is a detail section that gives you absolute control over every element of the fractal.

The last section is a transformation section that lets you scale, rotate and translate this source however you wish (you'll see why you'd want to do this in a later section).

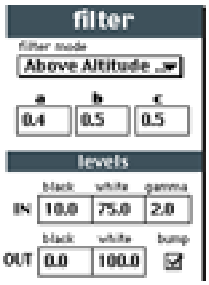
This module generates Value information that is fed into the Distort module.



Distort: Because life has very few straight lines

The distort section is a duplicate of the source section with one additional field, "distort amount." This puts a second fractal at your disposal strictly for distorting the output you made in the Source module. It works very much like the displace filter in Adobe Photoshop or After Effects™ (except it happens in 3 dimensions!). Wherever the Distort fractal gets light, the source gets pushed over in X, Y and Z. Wherever Distort fractal is black, the Source gets pushed in negative X, Y and Z. Wherever the Distort is 50% grey, the Source is left alone. The "distort amount" controls how much distortion occurs.

This module generates Value information that modifies the source value.



Filter: Making your shader free from impurities

The filter section is actually two filters in one.

The first filter is really a collection of 14 filters. These range from math functions, to filters that modify the Value based on environmental factors such as slope and altitude. You could use these filters to generate grass that only occurs when the model flattens out, or snow that only occurs above a certain altitude. These filters operate on both the Value and the Bump. The three boxes (a, b and c), represent parameters for the filters. The parameters that these boxes control will change depending on the filter that is selected. Several of the filters may only use 2 parameters. In this case, the number entered in box c will have no effect.

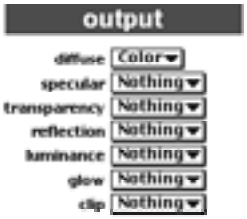
The second filter is a standard Levels filter. It mimics the controls from the Photoshop™ Levels filter. This is great for tweaking the output until it's "just right." You also have the option of applying the levels filter to the bump as well.

This module generates Value information that is fed to the Color Map and Output modules.

Color Map: Because you live in color

Up till this point, the shader has been working strictly with Value (or greyscale) information. The Color Map module is where you bring your shader into living color. This module will map the greyscale values into colors in a variety of ways. You can map to anything from no color, to 3 colors using either linear functions or splines. Black Values tend to take on the colors to the left, while white Values take on the colors to the right. In 2 color modes, the right color chip is not used. You can use the RGBA color chip in ElectricImage to set alpha values for the colors, to show through to underlying textures.

This module generates RGB color data (with alpha) that is fed, along with the original Value information, to the Output and Preview modules.



Output: Getting connected

Now that you've made your shader, you want to get it out to ElectricImage. Well, here's where you do it. You can take the values and colors that you've created, and map them out to almost any ElectricImage parameter. Diffuse, Specular, Transparency, Reflection, Luminance, Glow and Clip can all be reached from within a single shader. You can assign the color you've generated, the Value you generated (before you turned it into a color), the inverse of the Value or any combination of the above into any combination of ElectricImage values.

Global: Because the world won't stop and melt with you

The global sections contains controls that operate across the entire shader.

The gScale (or Global Scale) scales the entire shader (both the Source and Distort modules) so you can easily tweak the size of the shader from a central location.

The Bump controls how deep the bumps are that the shader generates. These can be positive or negative values.

The aa Limit is a control for anti-aliasing. Smaller values lend more detail to the final image, but at the risk of longer render times and possible aliasing. Higher values prevent aliasing and speed up the rendering, but at the expense of detail. You can usually leave this value at 1.0 for animations and 0 for stills, but situations may require different settings.

Preview: Show me! I'm from Missouri

With a shader this complicated, it can be a little tricky to see what elements are contributing to the final output. The preview menu allows you to view various aspects of the shader (Value, Color, Bump, Distort) without having to change any values in the shader. This can very helpful in developing complex shaders. The preview box itself has indicators to help you work with the environmental filters that use slope and altitude, which might not show up in the preview. **This popup only affects the preview and not the final output in Camera. Remember to set this back to "Preview All" when you are done, or you're just asking for confusion.**

With all of this control, it can be very tempting to start using all of the options that the shader provides. Exercise restraint. Using more parameters does not mean that your result will be any more appealing. Likewise, all of the modules are intelligent (as far as modules go) and when not used, are taken out of the processing chain to yield maximum performance at render time.