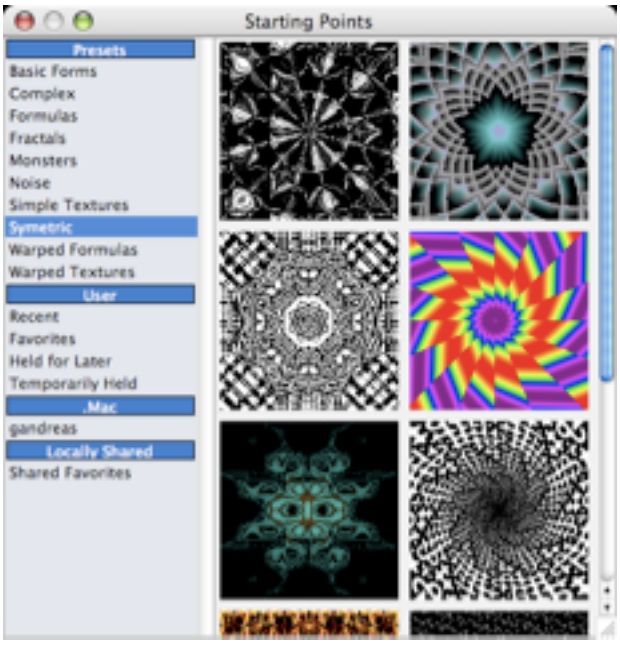
quadrium2
Tutorial

Lesson 1 - Making a Simple Image

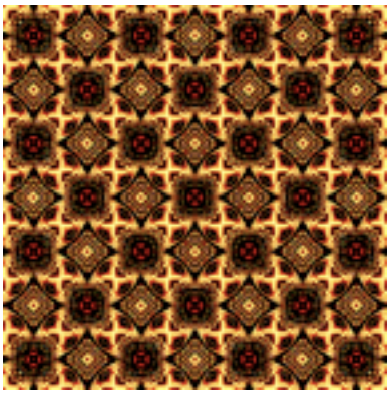
When you first launch quadrium2, you are presented with a dialog that shows various starting points:



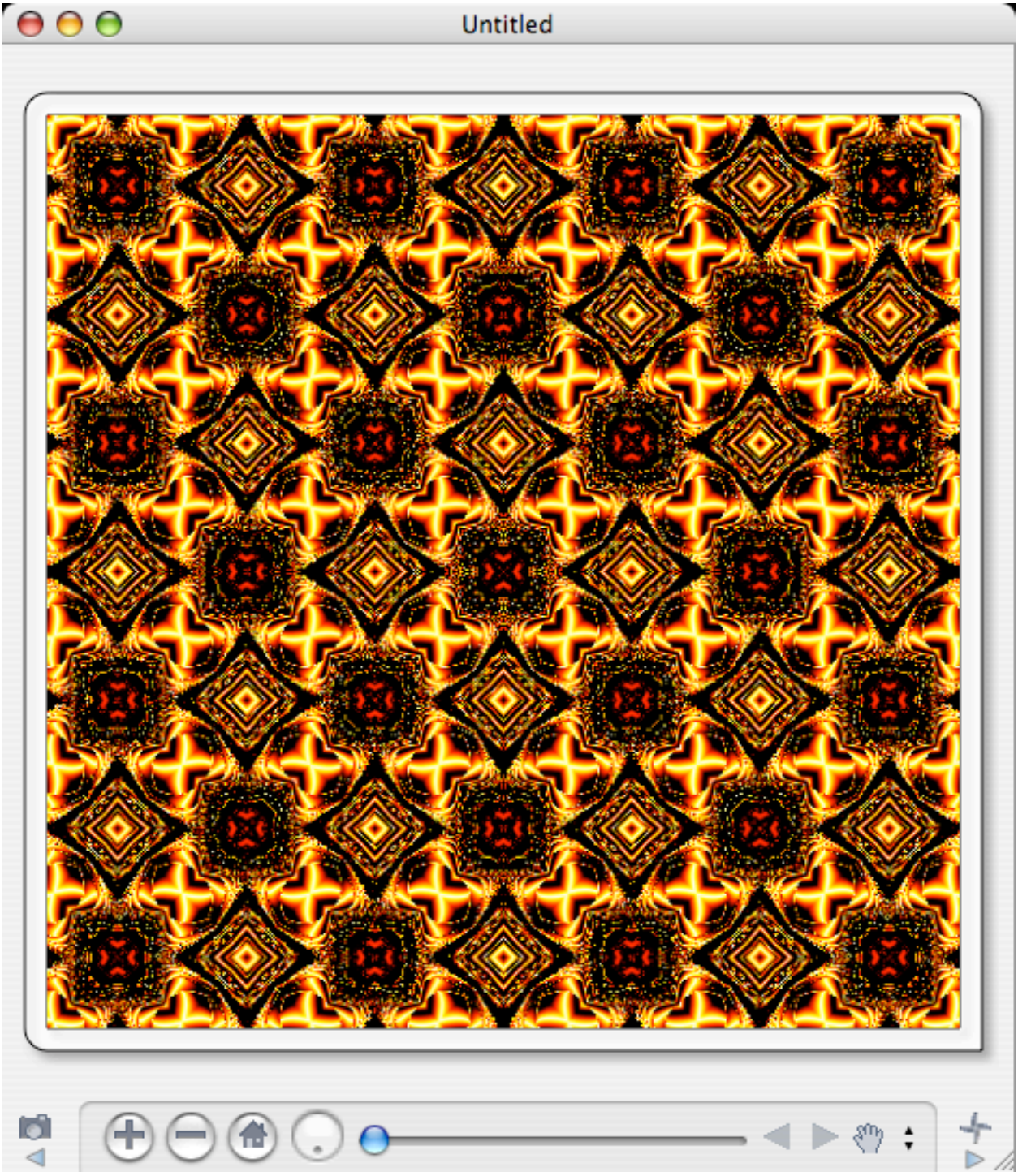
(There are also a bunch of “drop tabs” below the menu bar, but you can ignore these for the moment).

This contains a wide variety of images, sorted into different categories (called “Presets” because the come with the application). This is also where you’ll find your own personal favorites, as well as favorites found locally, or published via “.Mac” (you don’t need a .Mac account to access them, though you do if you want to publish).

The easiest way to create your own image is to start with an image similar to what you are trying to create, and let quadrium2 “evolve” the image into something more like what you desire. In our case, we’ll start with an image that looks like this:



(This is found in the “Symmetric” section, just peeking off the bottom in the above screenshot - you, of course, can use anything you want). After selecting the image, you should see the basic view window:

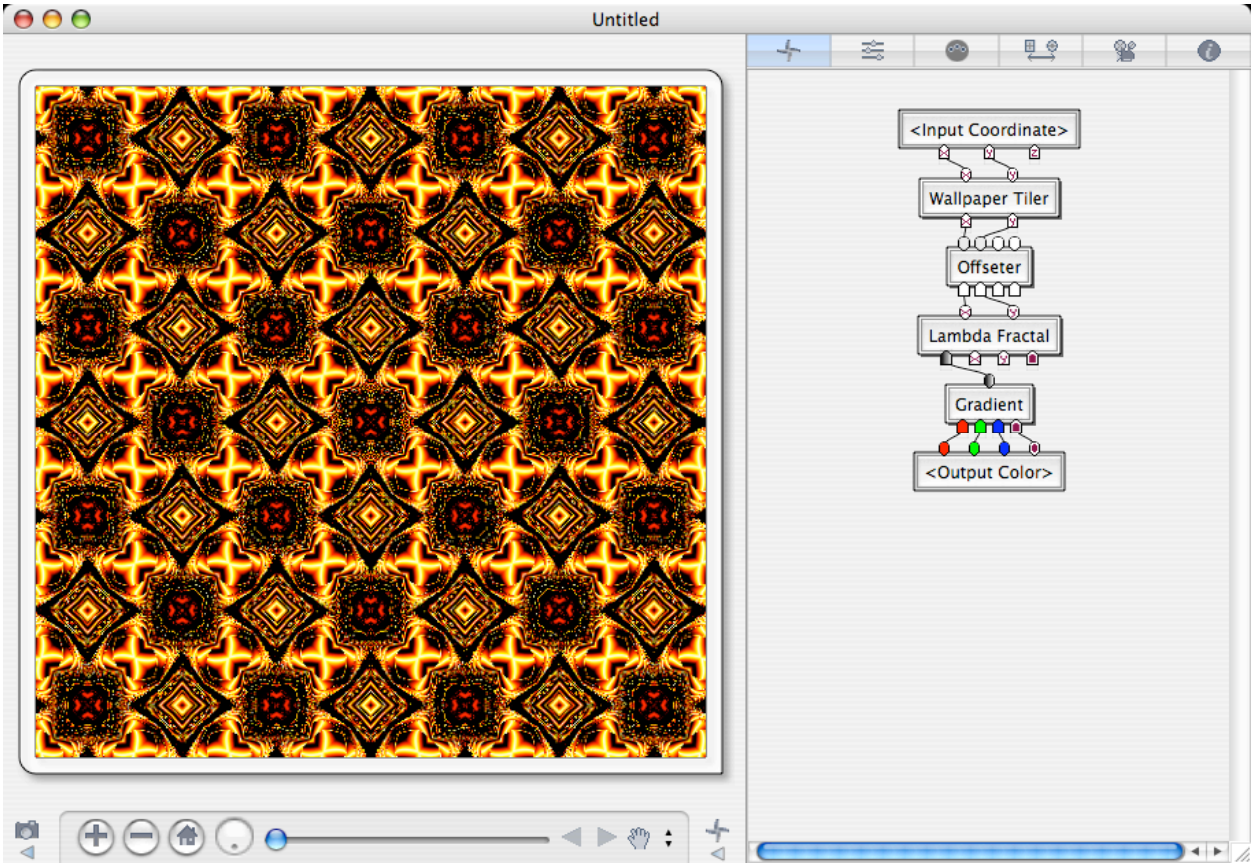


The top part, of course, is the image - you can pan around by clicking and dragging there. At the bottom are a series of navigation controls, as well as two buttons that toggle additional panels. From the left:

- Snapshot panel toggle

- Zoom In
- Zoom Out
- Return to Home view (this resets location of the view, rotation, and magnification)
- Rotates the image
- Toggles the details panel.
- The slider controls the “Z” value, which can be used to provide an easy way to change the image (this will be covered in more detail later).
- View Point History
- View Mode Popup.
- Details panel toggle

So this is interesting, but not what we wanted. So we’re going to explore the ability to randomly mutate the image. To do this, click the “details” toggle button, which will expand the window thusly:



(Clicking it a second time will hide this panel) The details panel has six different tabs within it, ordered as follows:

1) Tree view

2) Node parameters

3) Mutations

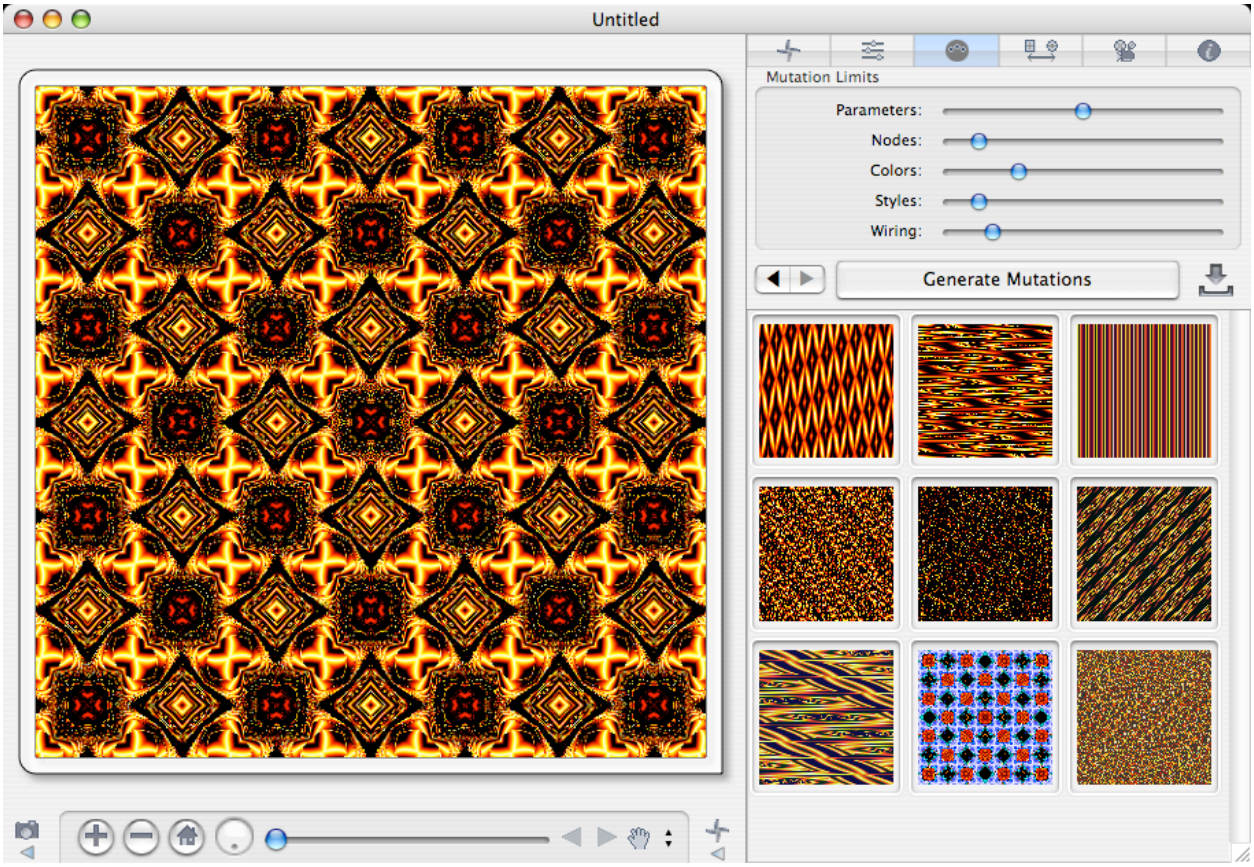
4) Tweener

5) Movies

6) Info

Tip: You can easily switch between these panels by clicking the tabs at the top, or typing the number 1..6. (You can also switch while performing drag & drop by pausing the mouse on the tab, which will switch to it - this is going to come in handy later).

Switching to the mutation panel, we see:



(The exact images you'll see will vary, since these are randomly generated)

What quadrium2 does is take the underlying "formula" for how to generate the image, and hten changes different parts of it in different ways, to produce a new image. As you can see, many of the images on the matrix are similar to the original - have a similar "checkerboard" like structure, but are slightly different, or perhaps with different colors. Others are radically different, though it may have the same basic color scheme. Some are interestings, others less so (the program tries to avoid generating something completely uninteresting, but beauty is in the eye of the beholder). All of these mutations are generated by five different techniques, which are controlled by the sliders you see. From the top:

Parameters - Controls the basic values, such as the scale of the image, intensity of the image, etc...

Nodes - Controls what is used to create the image. The more these change, the more the image changes.

Colors - This just changes the colors used in the image - the structure and shape remains the same.

Style - Controls some fundamental property of the image. Not all images include things that have “styles” but an example might be a grid of circles vs a grid of squares.

Wiring - Similar to nodes, it can have some serious changes in the image, or it could do something like change the a square full of circles into a circle full of squares.

Each of these has a slider associated with it, which limits how much the image changes when mutated - if you, for example, slide all the sliders to the left, all the generated images will be identical to the original (since you limited it to not change anything). So, for example, if you like the color scheme, you’d slide that to the left, and leave the others where they are. On the other hand, if it is really perfect except for the color scheme, slide the color slider to the right and everything else to the left.

If you don’t see anything in the list that you like, you can hit the “Mutate” button to generate another set of images. One of the more powerful approaches, however, is to “walk” through generations of mutations. If you double click on one of the small images, it will be used as the “parent” and generate mutations based on it.

Tip: If you want to see an image in more detail, you can hold down the control and option keys at the same time and click on in it (or option + right click if you have a two button mouse) - it will pop up a larger version of that image. This works just about anywhere that there are lists or matrices of images.

Once you find an image that you like, click and drag the image from the matrix and drop it on the main view - you’ll see a little thumbnail of the image in a frame as you drag it (like the quadrium logo, these frames have three rounded corners and a bottom right one that is sharp). Drag & drop is the single most important technique for quadrium2 - just about everything can be done by drag & drop.

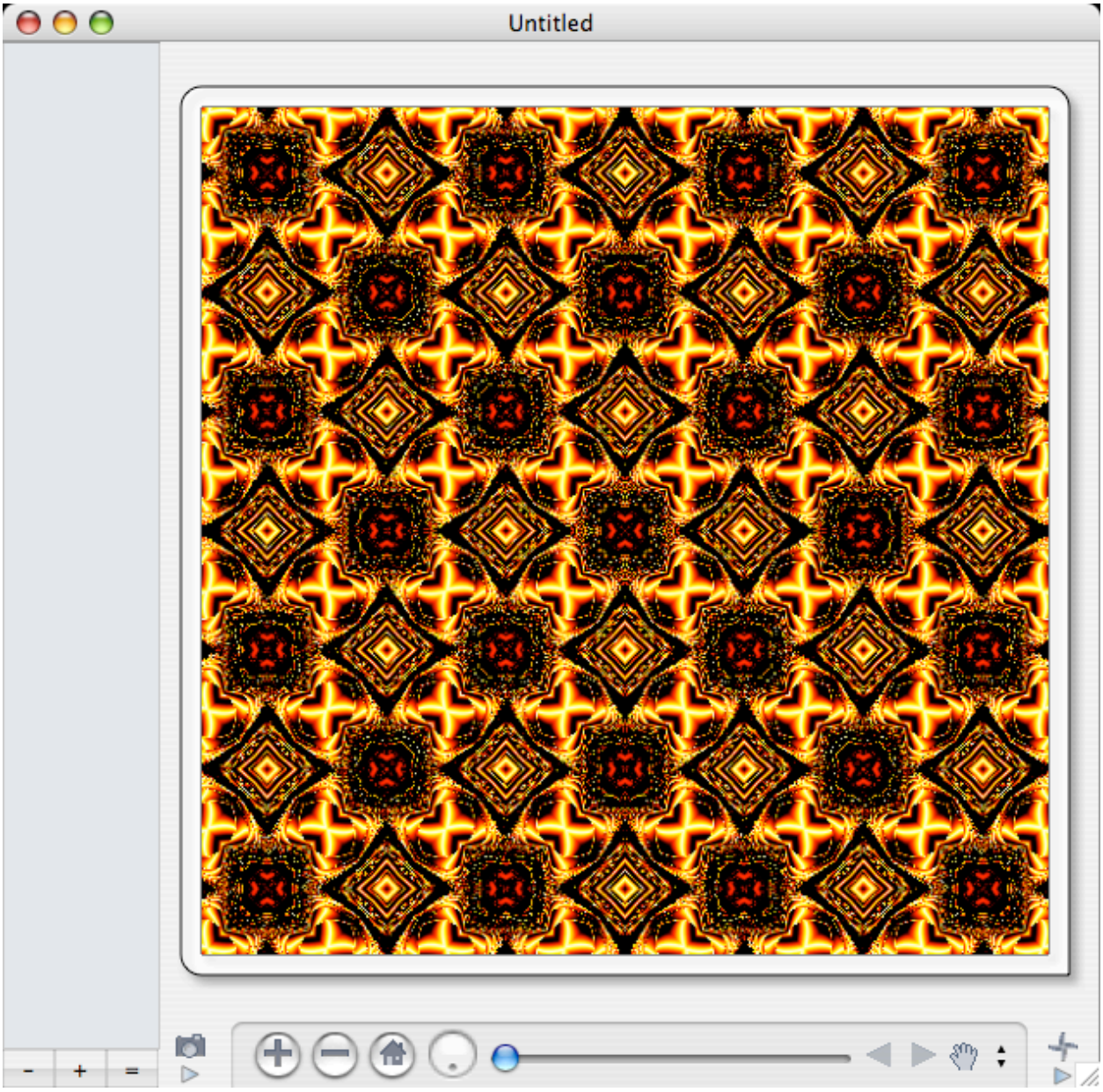
As mentioned earlier, quadrium2 provides a list of “drop targets” just below the menu:

These are used to perform an action on the image that you are dragging. From the left, this will create new document with this image, add it to the “breeder” (discussed later), hold the image for a later session, temporarily hold the image, add it to your favorites, share it locally, or publish it on your iDisk. So if made a mutation that looked good, and you wanted to get back to it later, you could drag it from the matrix and drop it on the “temp” drop target (the temporarily items are deleted when you quit quadrium2, while holding an image for later will write it to disk). These images will then be available in the corresponding category in the Starting Points browser.

While working on an image, you often come to a point where the image looks good, but not quite there, and you might want to save your progress so you don’t screw it up. You

can, like any other Macintosh program, just save the document to a new file each time, or you can take advantage of the “snapshot” feature. This allows you to keep a snapshot of the current image which you can then come back to at any time (these snapshots are also saved with the document). This also works well for coming up with a series of variations when you can’t decide between them.

To do this, click the Snapshot panel toggle button on the bottom left - this will reveal the Snapshot list (clicking it again will hide the panel).

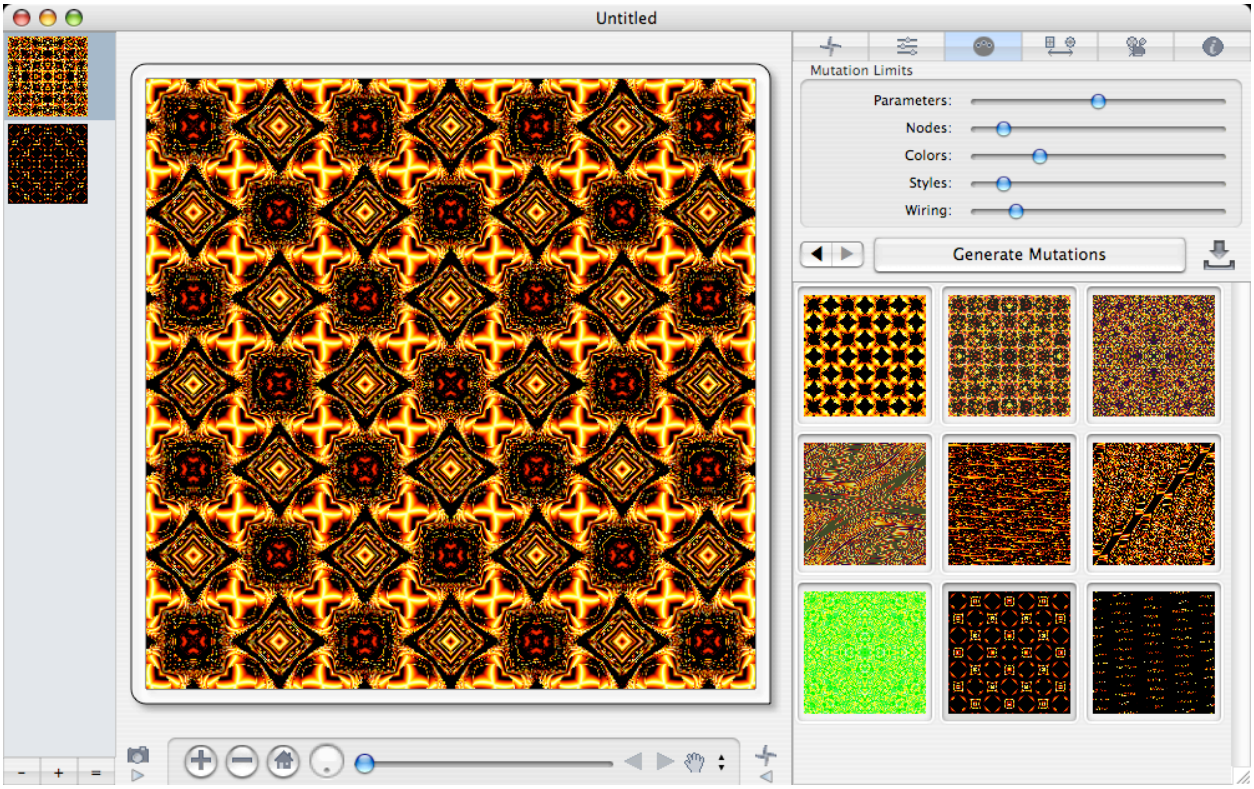


Any time you want to save your current progress, just click the “+” button in the snapshot panel and it will be added to the list. To restore from a snapshot, click the “=” button. To delete a snapshot, click the “-” button. But snapshots aren’t just limited to keep-

ing track of states of the current image - you can just as easily drag an image from the mutation matrix and drop it in that list. You can even drag snapshots between different documents. (If you want to drag the current image to another document's snapshot list, you can drag from the frame of the current image).

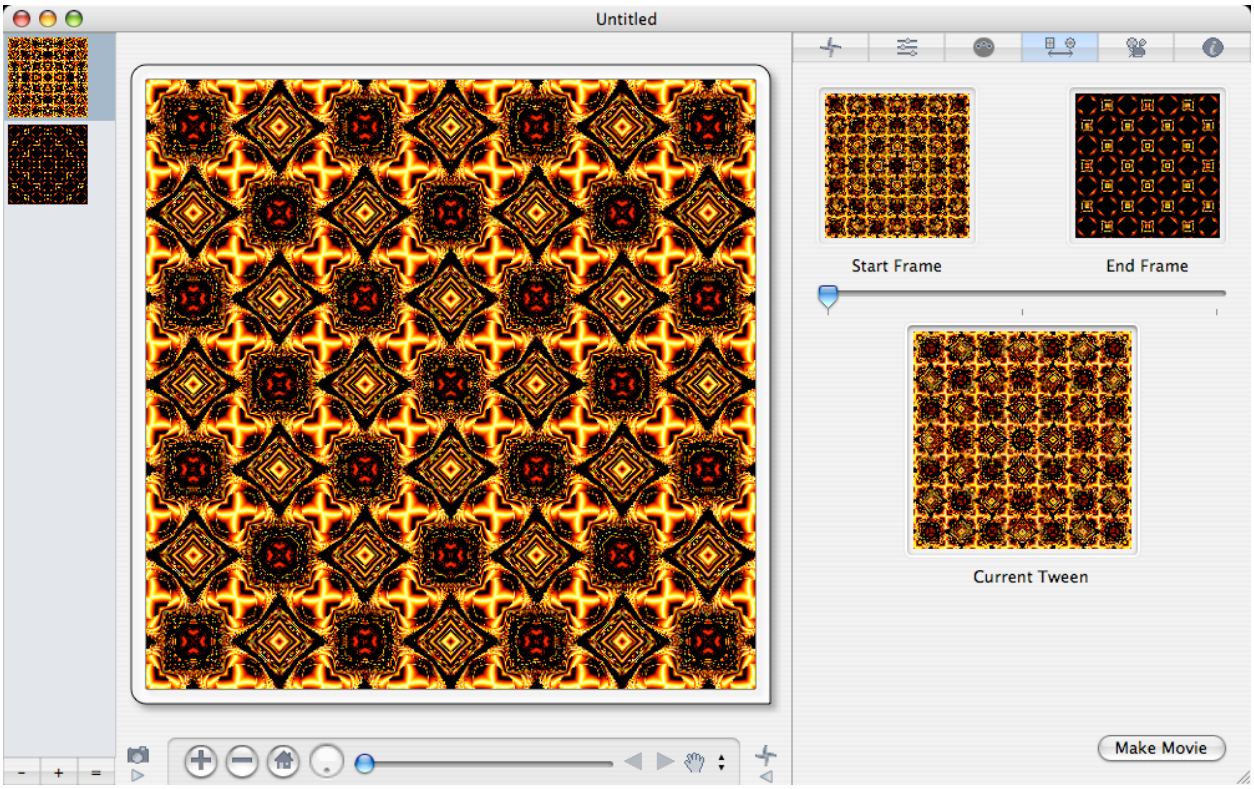
And just like the mutation matrix, you control+option click on an image in the snapshot list and get a popup larger view.

Let's suppose, now, that you've got your original image, and an interesting mutation, but wanted to have an image that was half way between these two images. That will bring us to the tweener panel. Before exploring this, find an interesting mutation, and drag it to the snapshot list. You'll also want to add the current document image to the snapshot list, so you window might look like this:

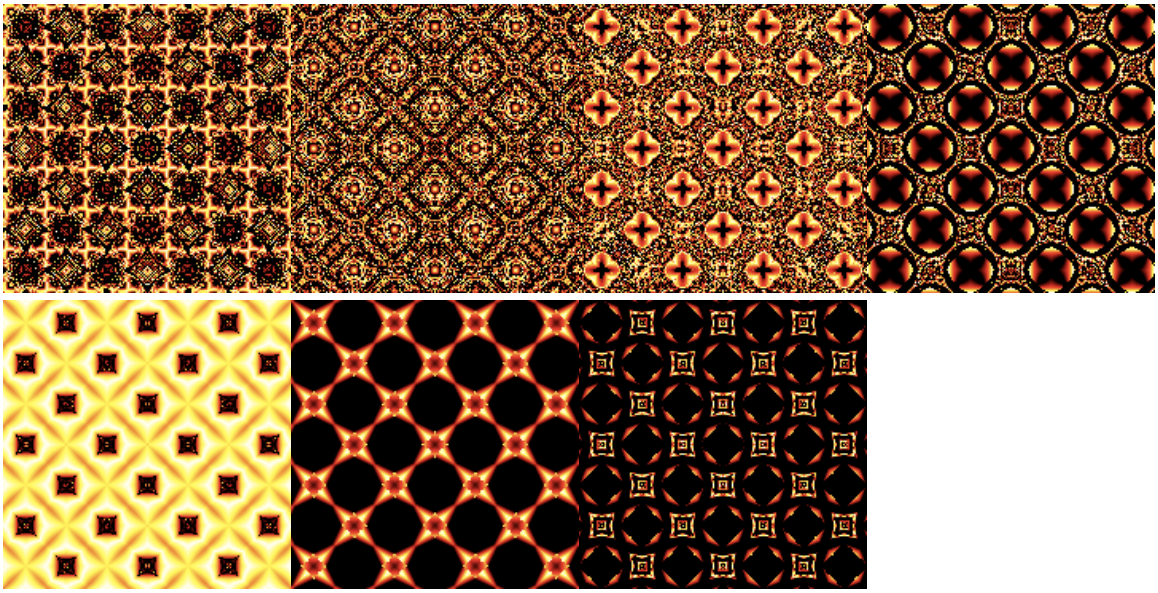


(We picked that bottom middle image as our "interesting mutation", obviously you're going to be seeing a different set of images - click "Generate Mutations" until you find something you like).

Switch now to the tweener panel (the fourth panel over, or just type a "4"), and drag the current image from the snapshot list and drop it in the left image well marked "Start Frame", and drag and drop the other one into the one marked "End Frame". It will look like this (obviously, your mutation is going to be different):



Now for the fun - drag the slider slowly from left to right, and you'll see the "Current Tween" change from one to the other:



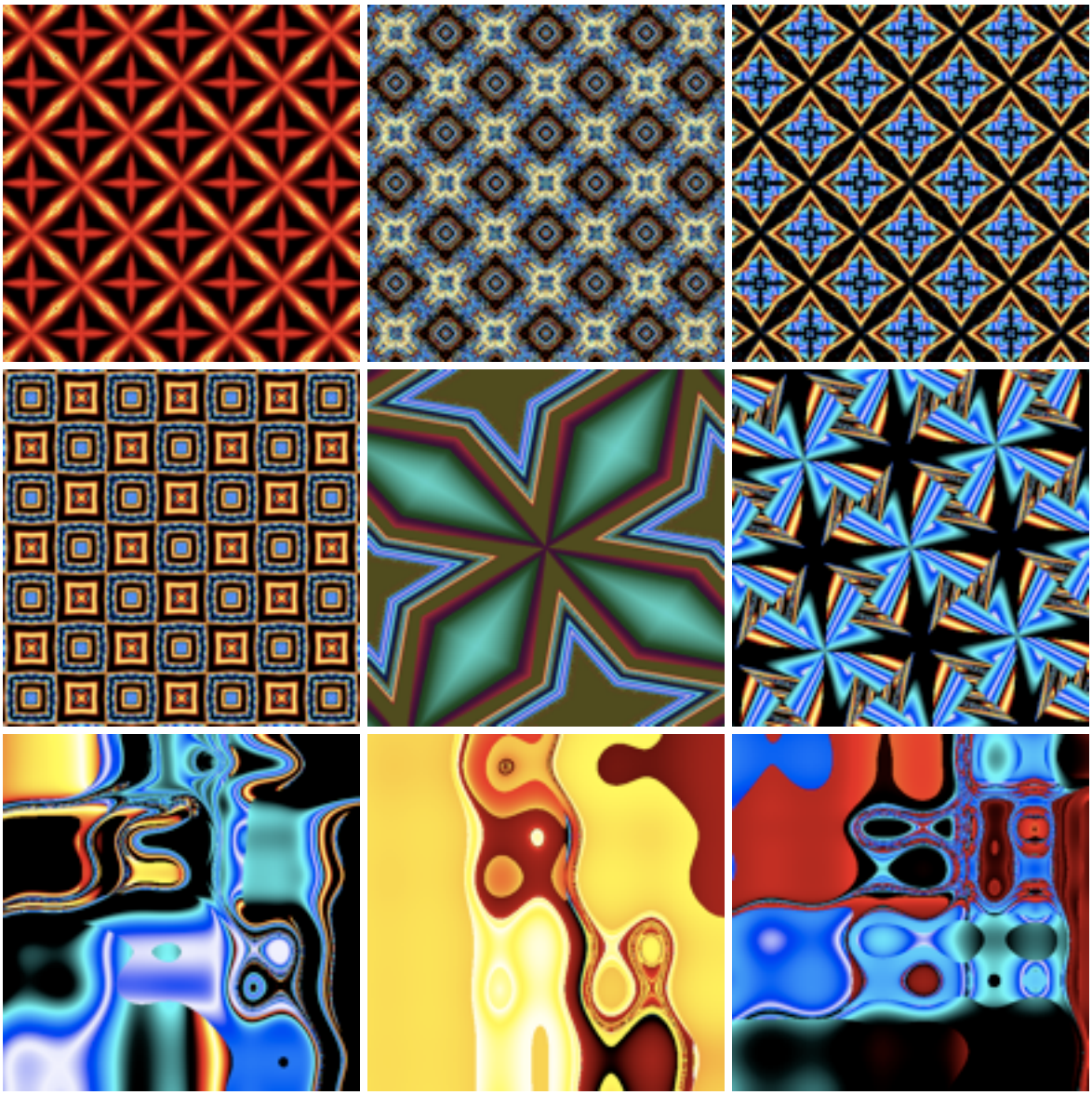
Depending on exactly what the difference between the two images is, you may notice that the rightmost position doesn't display anything that looks exactly like end frame. This is because of how "tweening" works. The idea is that we take current image, and

re-interpret it in terms of what is between the two key frames, keeping whatever we have in common, but not changing things which are different.

For example, using a real world analogy, imaging that our current document was two black cats, and the start frame was a single brown dog, and the end frame is a single white cow. Now, obviously, there isn't something that's halfway between a dog and a cow (except in the realm of a mad scientist), nor is there something between a cat and a dog or cat and a cow, much less anything between two cats and one dog. So in our example, if the slider was set the left (the single brown dog) we might have an image of two brown cats (since we can apply the color of the animal but not the species or count). And as as we slide to the right (the single white cow) the image will slowly turn from a brown cat to a white cat (since again, we can generate things "between" brown and white). The cats might also grow, since we can also interpolate between the size of the dog and the size of the cow. At no time, however, do we change from two cats to one cat, or between cats and cows or dogs.

As you can imagine, this tweening can be extremely powerful, especially since it will attempt to interpolate between a wide variety of things as best as possible. You can, for example, drag an image from the starting point browser into one of the tweener frames, and try to tween between a wide variety of completely different images. Many do not produce interesting results (what's half way between a firetruck and the concept of "truth"?), but some produce very nice results.

Using just mutation & tweening, we can produce all the following images from our starting image:



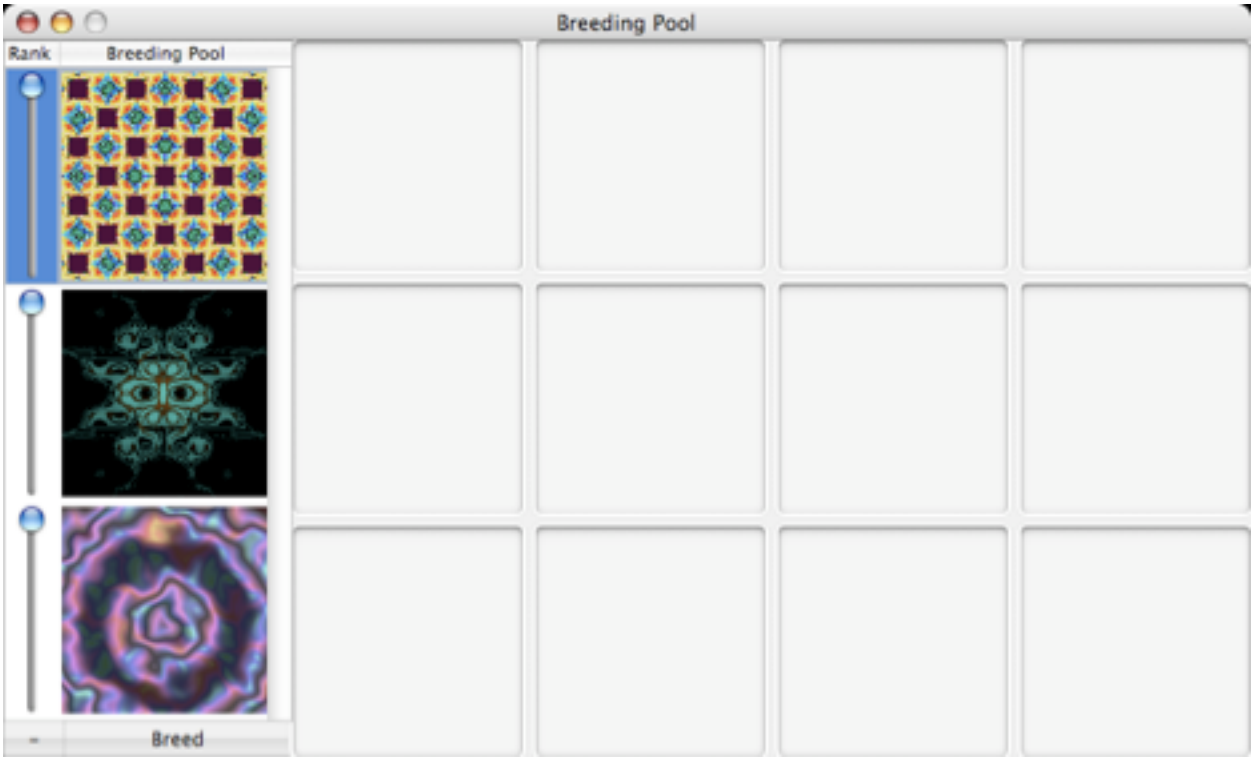
Obviously, the latter images are much further from the starting image (many more mutations) - at some point a “warped” quality appeared, and then was tweened with some of the earlier versions.

The fun part of all of this is that it is done by randomly exploring the infinite realm of possible images - quadrium2 generates the various candidates, you decide which ones you like. And we’ve only scratched the surface here...

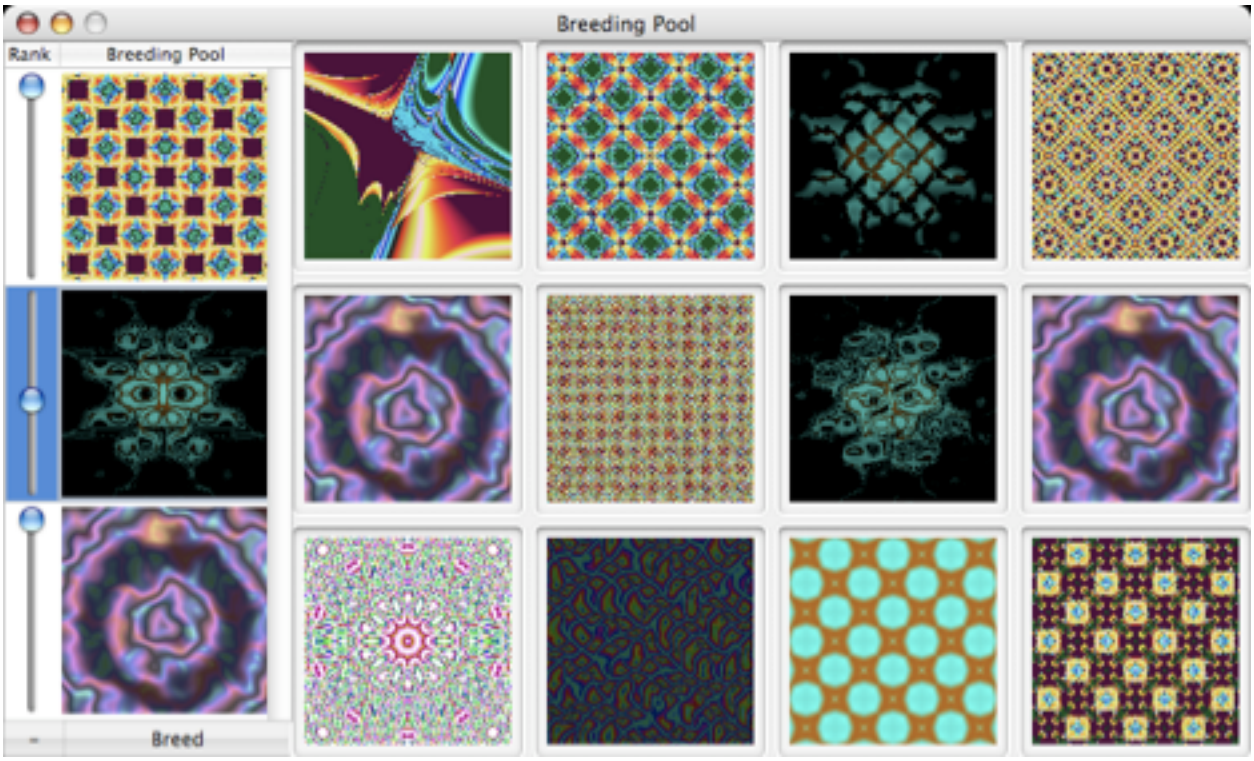
Lesson 2 - Breeding Images

In the previous lesson, we started with a built in image and mutated it into a series of new images. This was all done via mutation, but that is limited to what can be changed. Consider the variety of domesticated dogs that all come from a common wild dog ancestor - there are lots of varieties, but they are all canines with four legs, fur, etc... None of them have wings or opposable thumbs. But what if you could cross a dog with a bird and a monkey? Nature doesn't work that way, but quadrium does.

To do this, we've got a "breeding pool" - a set of images that we want to combine. To add an image to the breeding pool, simply drag it and drop it on the "Breed" drop target at the top of the screen (this image can come from any source, including regular files, snapshots, mutations, starting points, etc...) We'll take our last starting point and several others and add them to the breeding pool. Finally, we select "**Breeding Pool**" from the "**Window**" menu:

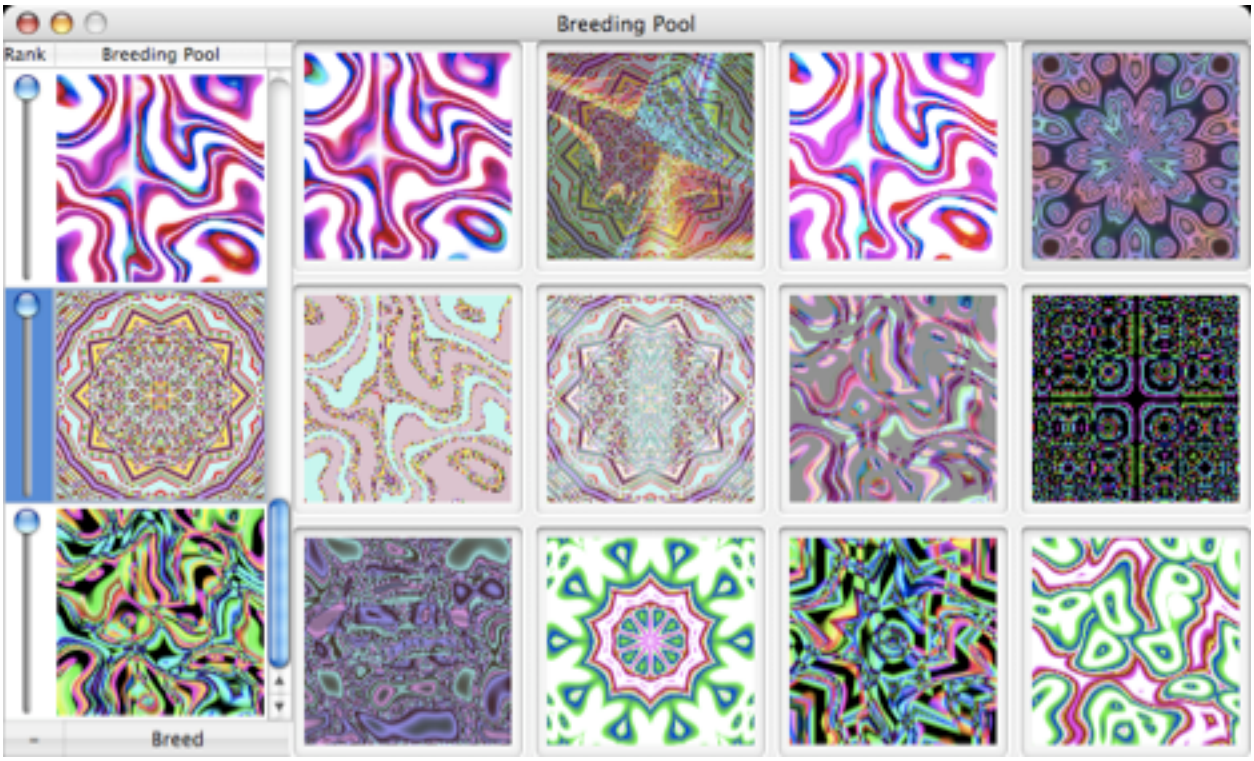


Here's the three images we selection (we removed the basic "color" image already using the "-" button). We'd like to combine the top and bottom image, and maybe the middle one as well, but we don't like it quite as much, so we'll rank it as lower. To do this, we've got a "Rank" slider that we can use to adjust how much we like the image - images with a higher "rank" are more likely to "breed" and produce new images. Once we've got things set up, we'll click "Breed" and see what we get:



As you can see, some are very similar to their “parents” (two of the purple ripple ones appear almost identical). Others are clearly combinations - you can see a combination of the grid and the face, as well as the face and the ripples. Others are all together different, such as the top left (that seems to have the coloring of the first image, but otherwise seems to have nothing in common with the rest).

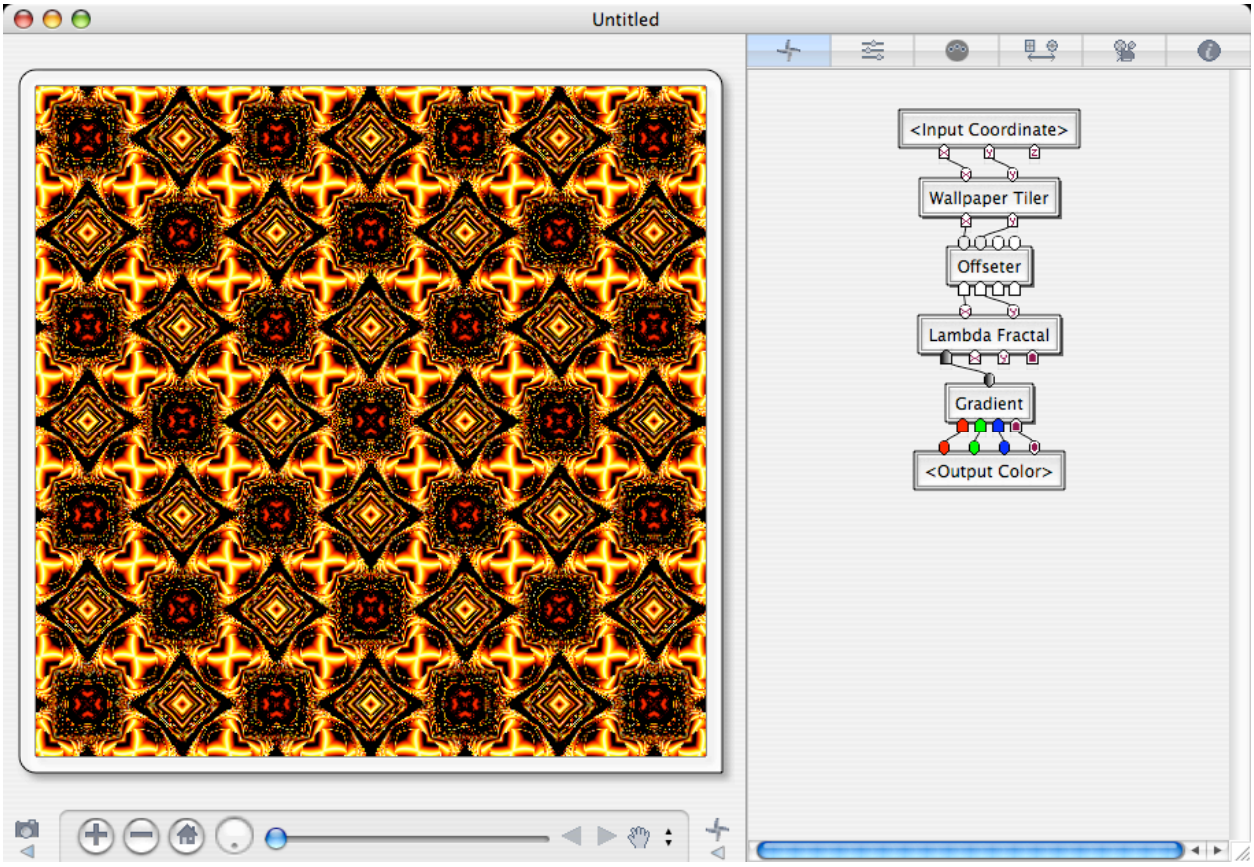
If we see any children we are interested in, we can select and drag them to any of the drop targets (to view them as a new document, or hold them for later). We can also drag them to the list on the left to add them to the breeding pool, and breed them in. Eventually we can then start removing the uninteresting parents and keep the more interesting children, eventually a wide variety of new images:



You can see elements of the starting images (such as the colors in some places, or the ripples in others) but the descendants are new and different. And this is done using only the simplest basic abilities of quadrium - mutation & breeding. In the next lesson, we'll start looking a the underlying structure of the image, and do the equivalent of manipulating the image at the "DNA" level.

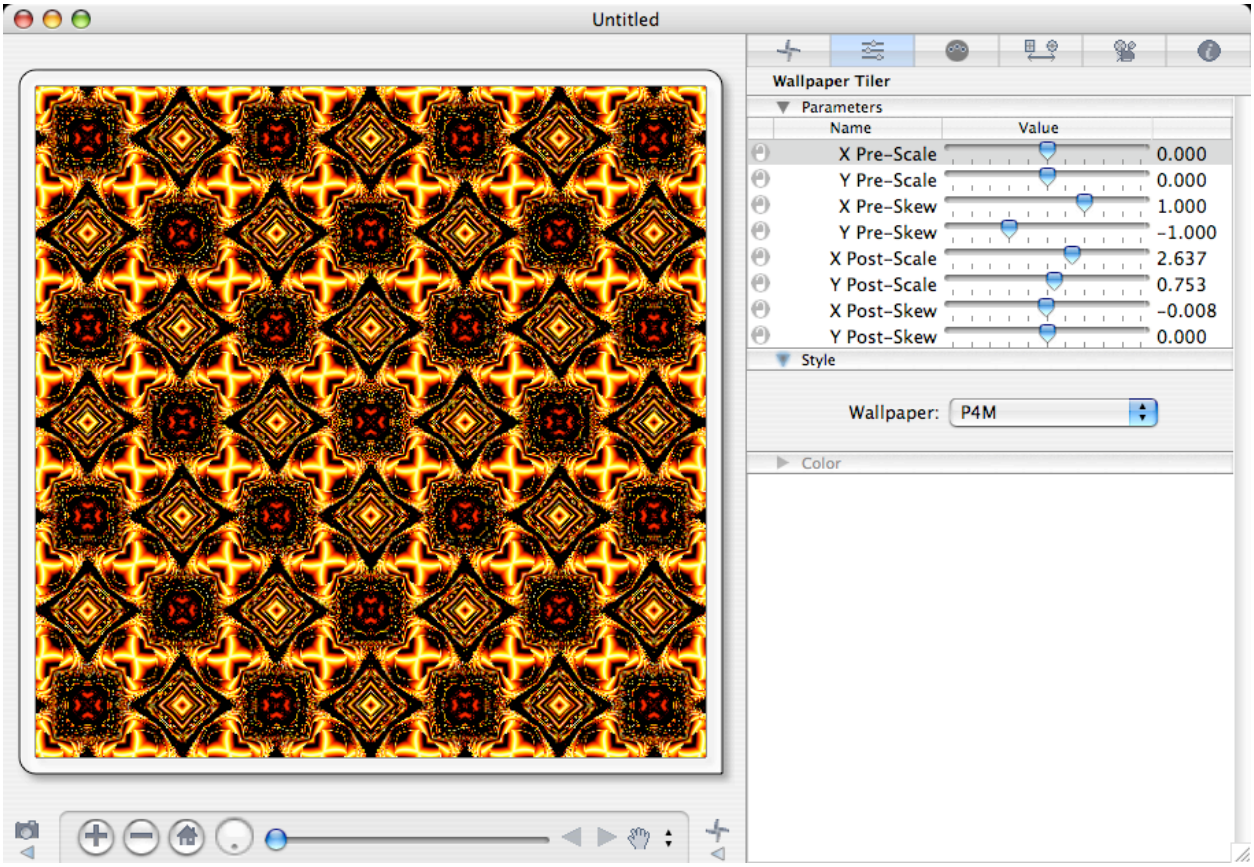
Lesson 3 - Image Structure

We'll go back to our original starting image, and rather than switching to mutations or the like, we'll just open the details panel with the first tab shown - the genetic structure:



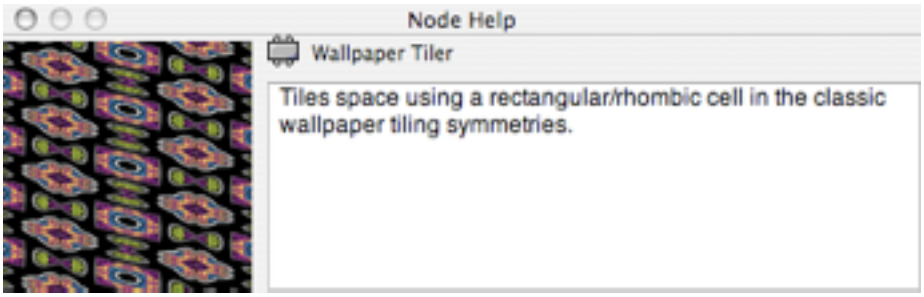
This view shows us the actual structure of the formula that we are manipulating. This is a sort of data-flow diagram. Looking at this example, quadrium takes the input coordinate (which represents where the pixels is located on the screen) and then uses that coordinate to feed the "Wallpaper Tiler" (which is responsible for making the image repeat across the screen). This then flows into an "Offseter" (which simply adds a constant to each coordinate). From there it goes to a "Lambda Fractal". This node is actually quite a bit more complicated, but ultimately, it creates a fance & detailed grayscale image. That grayscale coordinate is fed into a "Gradient" which converts from grayscale to full color. That color is the final color of the pixel at the specified coordinate. quadrium evaluates this for every pixel on the screen to produce the final image.

Each of these nodes then has potentially a whole bunch of settings associated with it (some have none, but most have at least a couple). To see these settings, double click on one of the nodes (or select a node and type 3 to switch to the third tab, or simply hit return with the node selected). We'll select the "Wallpaper Tiler" node and look at it's parameters:



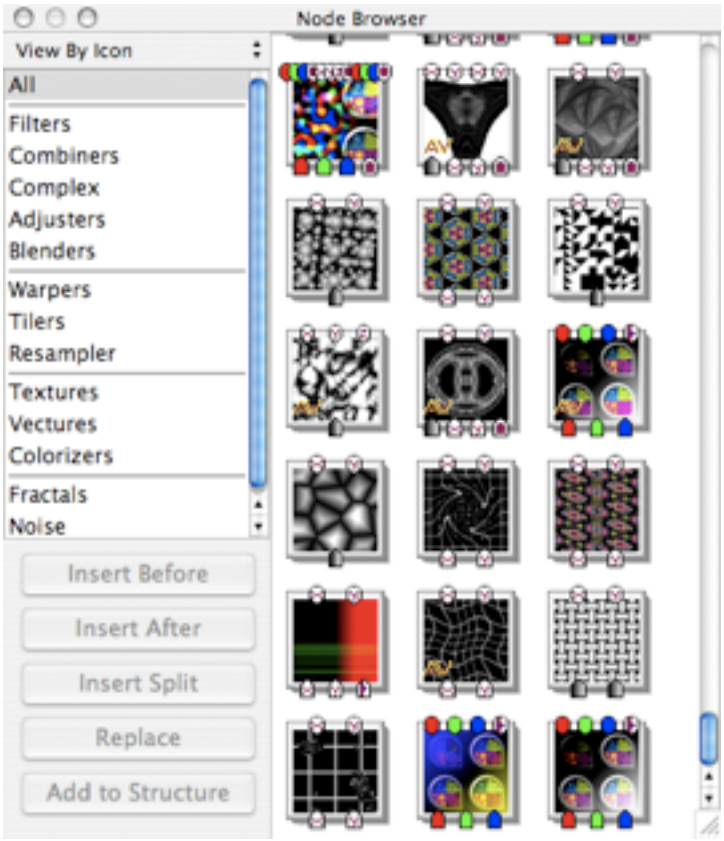
This node has eight parameter sliders, and a single additional “style” control. Some nodes also contain color information, which would appear in the third subpanel (since this node doesn’t have any, that one closed). If you select the “Gradient” node instead, you’ll a different set of information in each of these three panels.

When you select a node, you’ll normally see two additional floating panes appear:



This is the help for that specific node. It shows a preview of what sort of image/effect this node produces (in our example, it repeats a section of an image across the rest of the image), as well as a small summary. It also has a little icon showing what sort of connections it has (in this case, it takes two inputs and has two outputs). If you close this window, you can get it back by selecting the **Help > Show Node Help** menu item.

The other floating palette looks like this:



This is the node browser window (available from Window:Node Browser). For now we'll ignore it, so you can close it.

Getting back to our parameters pane, we can start tweaking the parameters. If you drag the slider labeled "Y Pre-Scale" (make sure that the Wallpaper Tiler node is selected if you don't see it) all the way to the left, the image will change to:

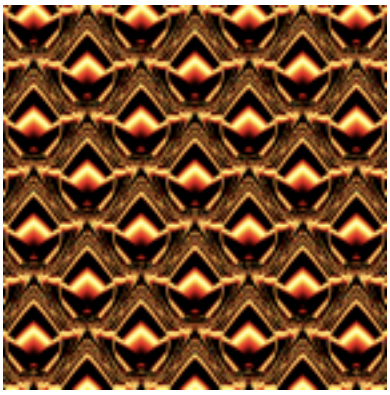
Notice how the image gets stretched out as you drag the value. You can undo this to return it to it's original state. You can play around with other sliders and observe other

actions. To make life easier to tell what does what, quadrium provides live feedback, instantly showing you what you are doing (it displays the image in a slightly reduced quality to import performance - you can adjust this in the preferences pane).

If you want to change the value to something beyond what is available in the sliders, you can edit the number in the last column. (Note that some nodes have hard-wired maxium and minimum values). This editing also makes it easy to specify specific values. When a parameter is selected, you can also copy & paste the value between parameters or external text editors.

Finally, the first column includes a little “lock” icon. If you remember back to our mutation exercise, one of the things that can be mutated are the parameters - these value that you are now playing with (one way that a mutation is generated is simply by randomly adjusting these sliders). If you click on that icon, it will lock that parameter so that mutation will not ever change things. This is very handy if you want to use mutation to fine tune an image - you might lock down the nodes that you like and leave the others unlocked, and then explore various mutations to see what looks good.

Looking at the “Style” panel, we see that node has a menu controlling what is called the “Wallpaper” (which is a cryptic letter/number combination based on the science of crystallography, in case you were wondering). This controls additional behavior of the node - if you select “CM” instead, you see the image change to something like:

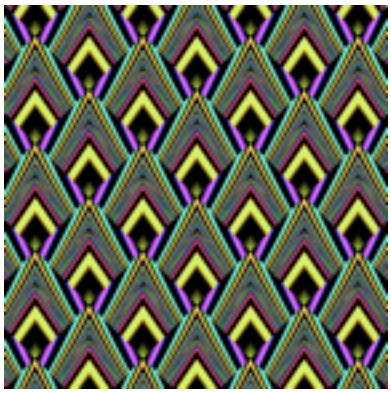


Feel free to try all the various styles - some are quite pleasing, others don’t work as well. In general, this should be your attitude when using quadrium - even if you have no idea what something is, or what it means (and, unless you’ve taken classes in crystallography or abstract math group theory, you’ll probably have no idea of the difference between CMM and P4G), experiment with it anyway. It may make something better, or maybe it won’t. You can always undo what you don’t like.

If you select the different nodes and view their parameters, you’ll see different sets of parameters, style settings, and/or color settings. The gradient node has all three. If you select the gradient in the color panel, you’ll be presented with a floating window that allows you to edit your gradient (not unlike may other drawing, painting, and illustrating applications). If you select the lambda fractal, you’ll discover an overwhelming number of style options (which are covered in more detail in Appendix B of the Reference Manual).

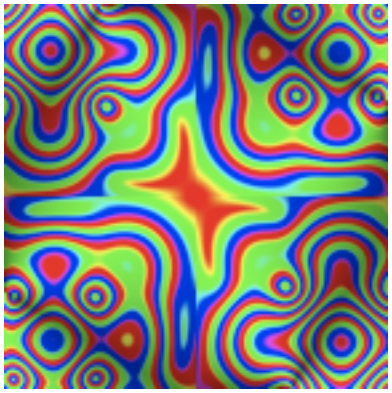
If you feel up to a real challenge, see if you can change the image to look like this:

Here's a hint or two - the wallpaper tiler parameters & style were changed, as were the parameters in the offseter and the style in the gradient node. The fractal node was untouched.

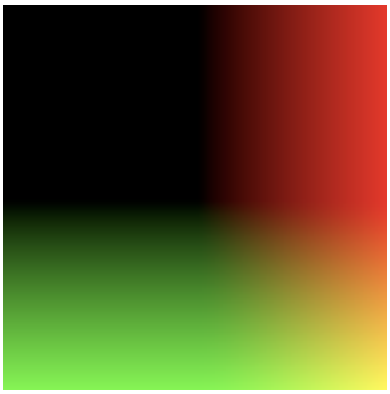


Lesson 4 - Building Structures

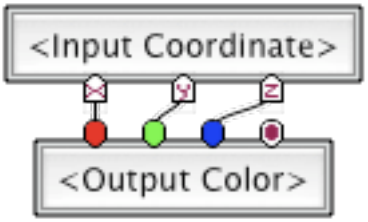
In this lession we're going to make an image completely from scratch - no starting points, no mutations, no breeding, no tweening. We'll build it up node by node until we get our final result:



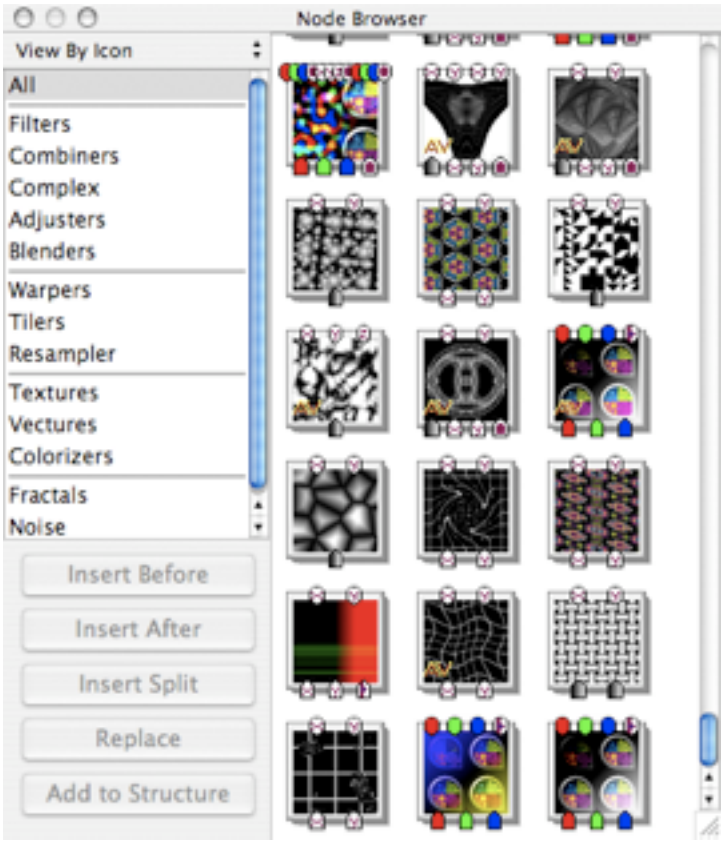
So we start with a new blank document as made by File:New Blank Document. This is our familiar color splash seen as one of the starting points:



If you look at the structure, you'll see how simple it is:



So the first things we'll do is add a node to this, so make sure that the node browser is shown (clicking on the structure panel should do it - otherwise select Window:Show Node Browser). The node browser contains two major areas:



On the left is a list of categories, and on the right are the nodes in that category. These are the current categories (for more details, see the corresponding index page in the Reference Manual):

All - Shows all nodes

Filters - Simple nodes that operate on four channels in parallel

Combiners - Takes four separate channels and combines them into one channel

Complex - Performs complex math functions. Takes either two or four inputs, produces two outputs

Adjusters - Adjusts colors. Takes RGB data and adjusts it to form different RGB data

Blenders - Combines colors. Take two RGBA sets of data, and creates a combined RGBA output

Warpers - Takes X & Y coordinates, and distorts them to produce a new coordinate

Tilers - Similar to warpers, but the output will repeat, as opposed to just being warped

Resamplers - Changes the effective resolution of the image to something like a mosaic, for example.

Textures - Takes X & Y coordinates, produces a grayscale texture/image.

Vectures - Take X & Y coordinates, produces a “two dimensional” grayscale texture

Colorizers - Take grayscale or X & Y coordinates, produces a color for that

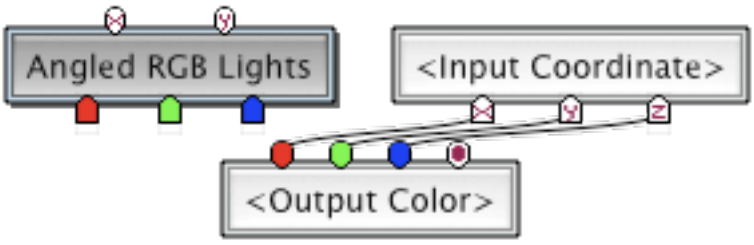
Fractals - Nodes which produce fractal effects, textures, or images.

Noise - Nodes that produce random “noise” based effects, textures, or images

The nodes on the right are displayed based on the category setting on the left. You can change how they are display by the small menu in the top left corner. For example, if screen real estate is limited (say, on a laptop), you might want to display by name only (which takes less room).

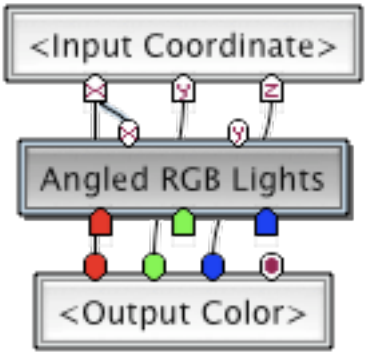
For now, we’ll start by adding some different color. So select the “Colorizers” category and select the “Angled Lights” node - it should be in the top left corner and look like this:

(If you have a hard time telling which is which, you can move the mouse over the node and leave it there until a tool tip appears, or select the node and look at the Node Help window, which will display the name and information). Once selected, click the “Add to Structure” button (which should now be enabled). Your structure will now look like this:



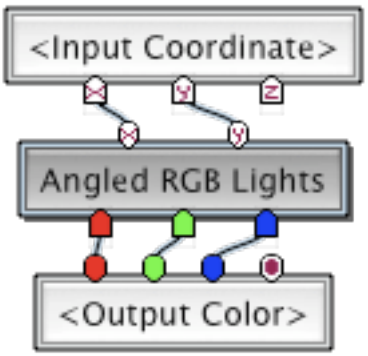
What you want to do is connect the X & Y “nubs” at the bottom (the output) of the <Input Coordinate> node to the X & Y “nubs” at the top (the input) of the new node. To do so, click on the left most output of the <Input Coordinate> and drag the resulting line to the

left most input of “Angled RGB Lights”. The structure will get laid out automatically to look like this:



We will repeat the process for the second output of <Input Coordinate> to the second input of “Angled RGB Lights”. And you’ve probably noticed, the connections have little hints as to their uses - X, Y and Z are coordinates, solid red, green and blue are colors. That purple “dot” indicates alpha transparency (and a gradient gray node indicates grayscale values).

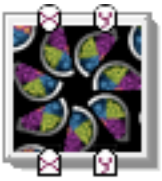
Connecting up nodes one at a time can be time consuming, especially if we have a bunch of things to hook up all in parallel, such as the output of Angled RGB Lights and the input of <Output Color> that we’re going to do next. If you hold down the shift key, it will select all the outputs from where you clicked, and doing so we can quickly connect the “Angled RGB Lights” to the <Output Color>:



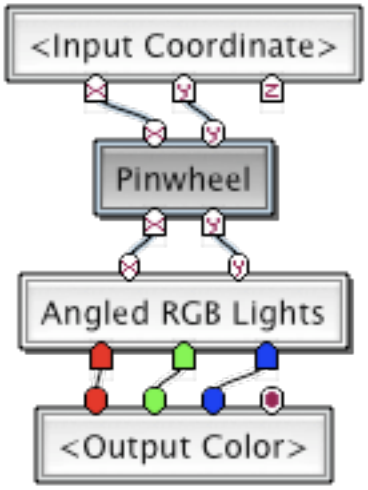
As you’ve probably noticed, this last change also changed the output display in the main view:

Obviously, parameters in the “Angled RGB Lights” node can be changed, as we saw in the last lesson.

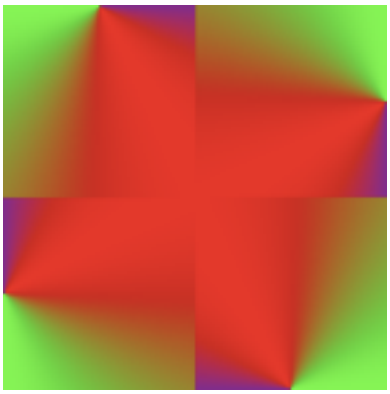
We want to add another node to our diagram. We’ll add a tiler of some sort “upstream” from our new node. While “Angled RGB Lights” is still selected, choose the “Tilers” category in the node browser, and then select the “Pinwheel” which looks like this:



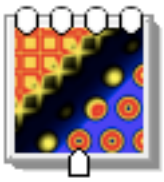
Now, rather than adding it to the structure and manually wiring it in, we can just click the Insert Before button (if this button isn’t enabled, make sure that “Angled RGB Lights” is selected in your structure pane, and that “Pinwheel” is selected in the browser). This will automatically place it in the structure and connect up the various inputs and outputs, producing the following structure and image:



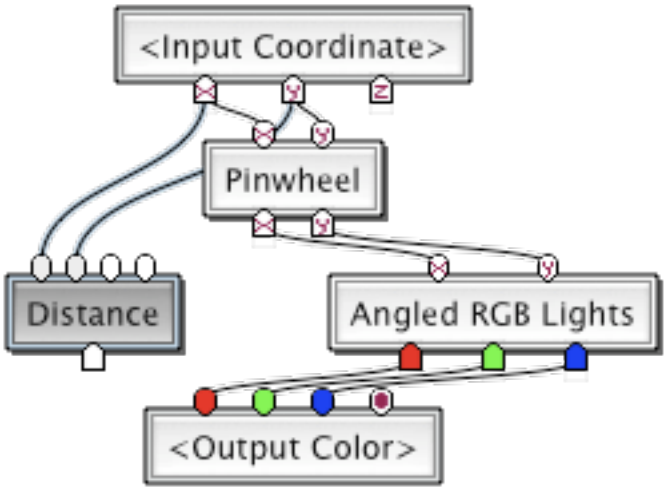
We want to change the “Y Offset” parameter to 0.5 and “Pinwheels” to 4. This will produce this:



We are then going to go to “Combiners” and select the “Distance” node:



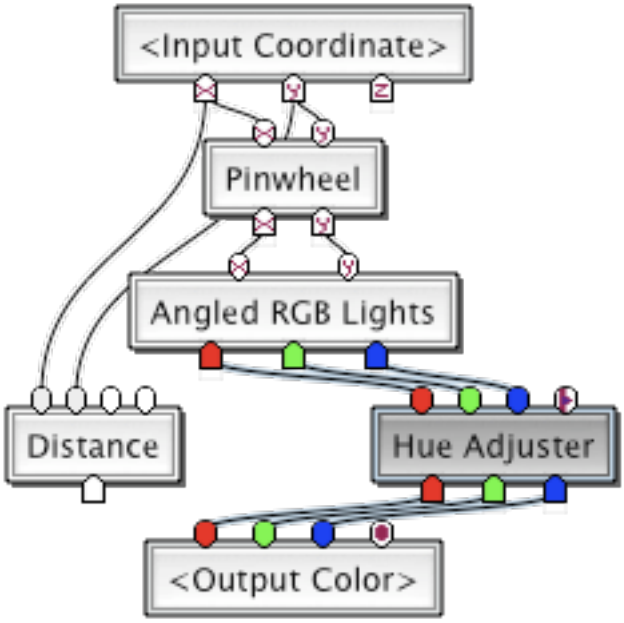
This time, with “Pinwheel” still selected in our structure, we’ll click on the Insert Split button. This cause the node to be inserted, and it will have all the same inputs as the selected node in the structure, effectively “splitting” the structure at that point:



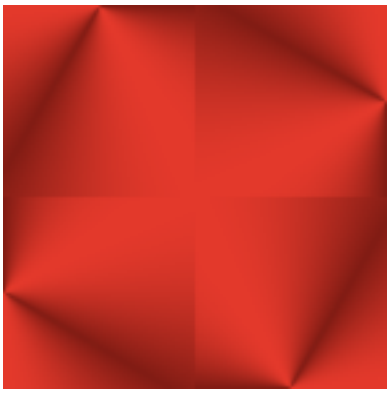
The idea is that we want an image that change it’s color based on how far away it is from the center it is - that’s what “Distance” will give us. You will also want to set the parameter “Scale” to 1.0.

Now go back and reselect “Angled RGB Lights” in the structure, and then, in the node browser, select the “Adjuster” category and find the “Hue Adjuster”:

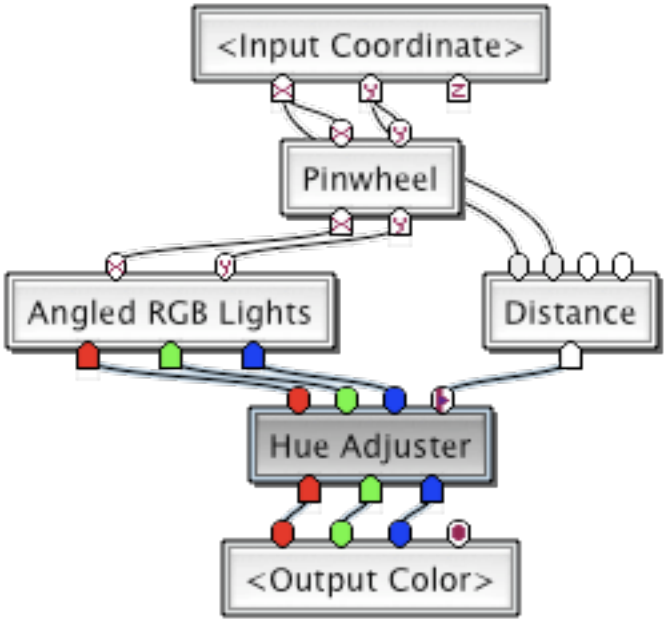
The last input with a little notched purple line stands for a control or parameter input. We will click the Insert After button to place this downstream from the “Angled RGB Lights” thusly:

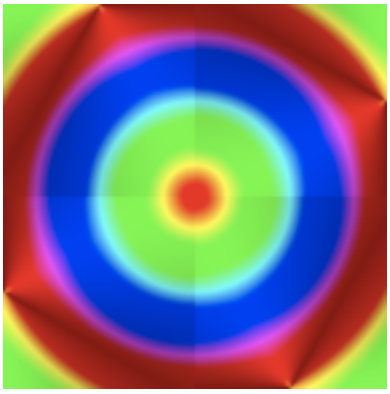


You might be surprised to see the image turn red:

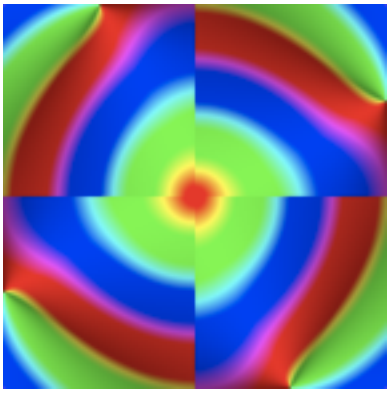


This is because the “Hue Adjuster” defaults to setting the hue to the value of that last input - in this case, since nothing is connected to that input, it makes the hue red (but leaves the saturation and lightness the same, which is why it looks similar, but just with all the colors replaced with red). So now manually connect the output of the “Distance node with that last input in the “Hue Adjuster” node:





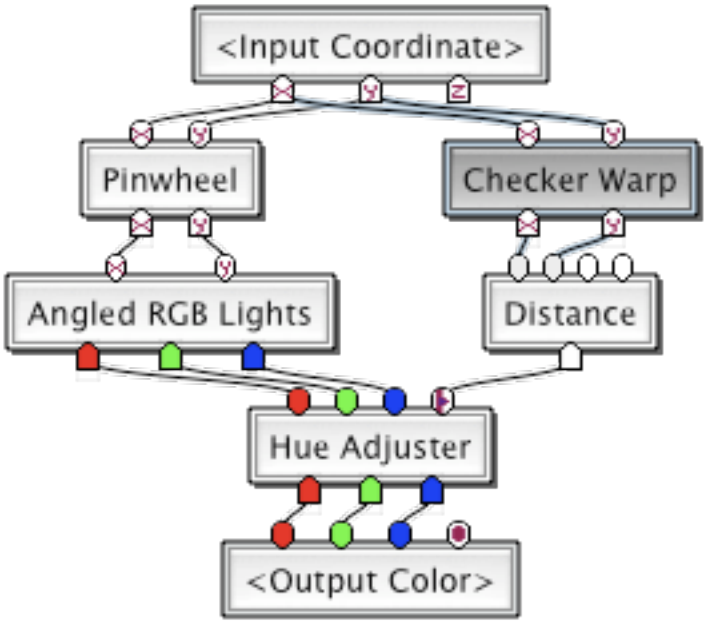
Now the hue is set based on the distance from the center (make sure that the “Distance” node’s parameter “Scale” is set to 1.0). If we change “Hue Adjuster” style to “Adjust” instead of “Set” we get:

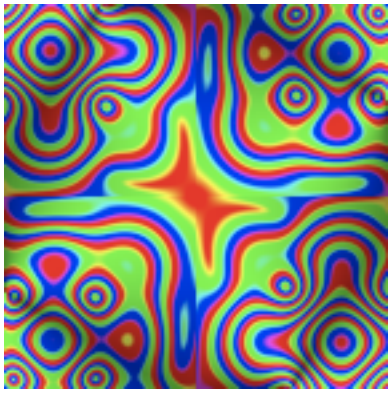


Feel free to play with the various parameters and style settings to see how we can change things around.

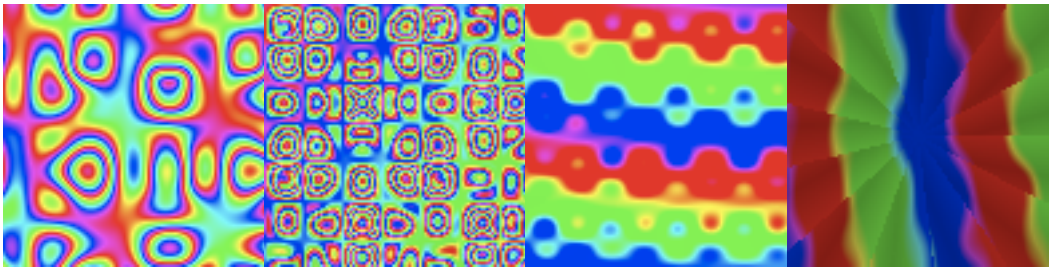
We’ll make one more change - select the “Distance” node and then in the “Warper” category, find the “Checker Warp”:

This time, click the Insert Before (placing it “upstream” of the “Distance” node), producing this:





All of this was created by adding five nodes to an image. It’s rare that more than a dozen nodes is ever added to create an image, so this simple exercise actually produced a fairly complex image. To see what sort of possibilities there are with what you’ve built, bring up the mutation panel and then move the “parameters” and “style” sliders to the right (maximum) and all the others to the extreme left (none). Click mutate and see how randomly setting the various parameters/styles of your creation produces a huge variety of images:



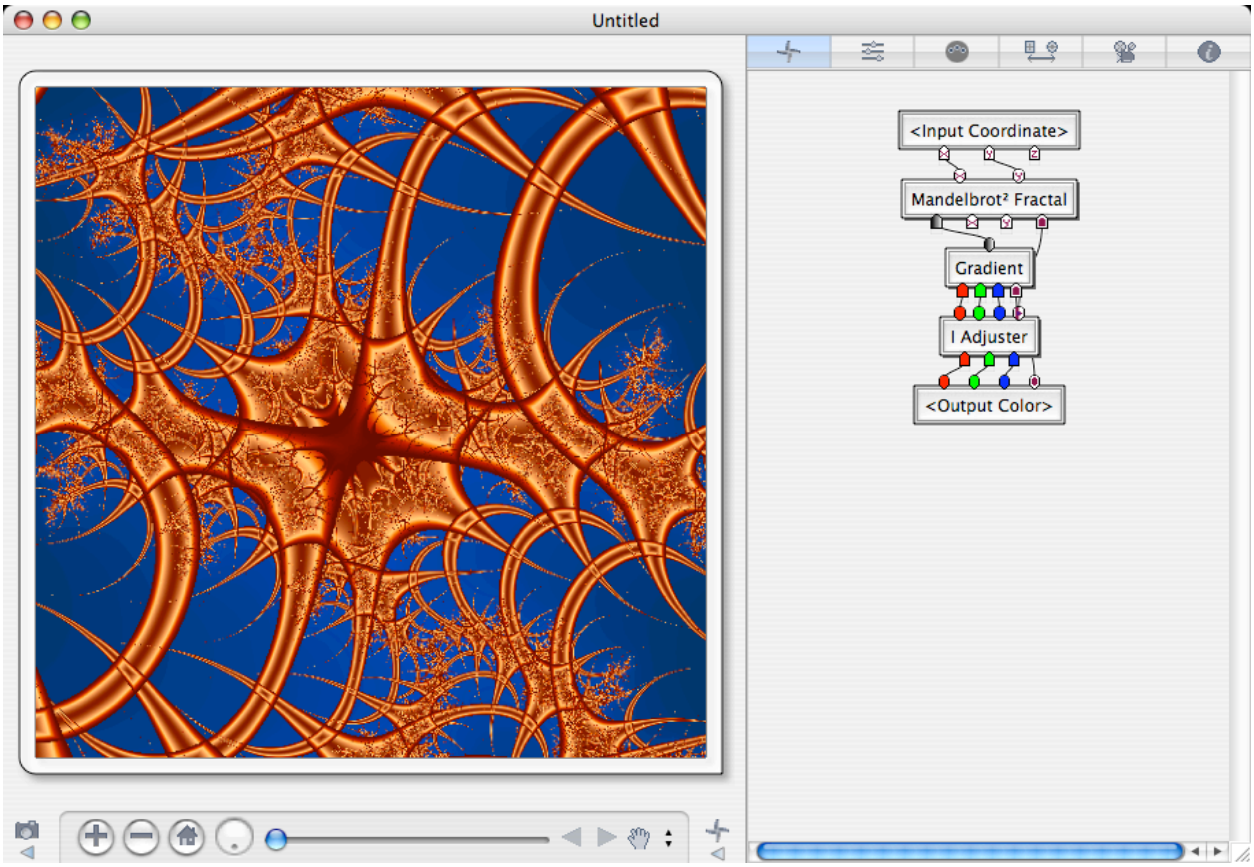
Besides clicking the various buttons in the node browser to add a node to the structure, you can also add nodes via drag & drop. If you drag the node's icon from the browser, you can drop it in your structure pane. If you drop it on top of an existing node, it will replace that node. If you drop it at the top edge of a node, it will insert before. Dropping on the bottom edge inserts after, and dropping at the left edge will split the node.

To remove a connection, you can click an drag from the input - you can drag that end to another input if you want, or let go away from any input to remove it completely. Remember - dragging from an output will make a new connection, while dragging from an input retargets that end only.

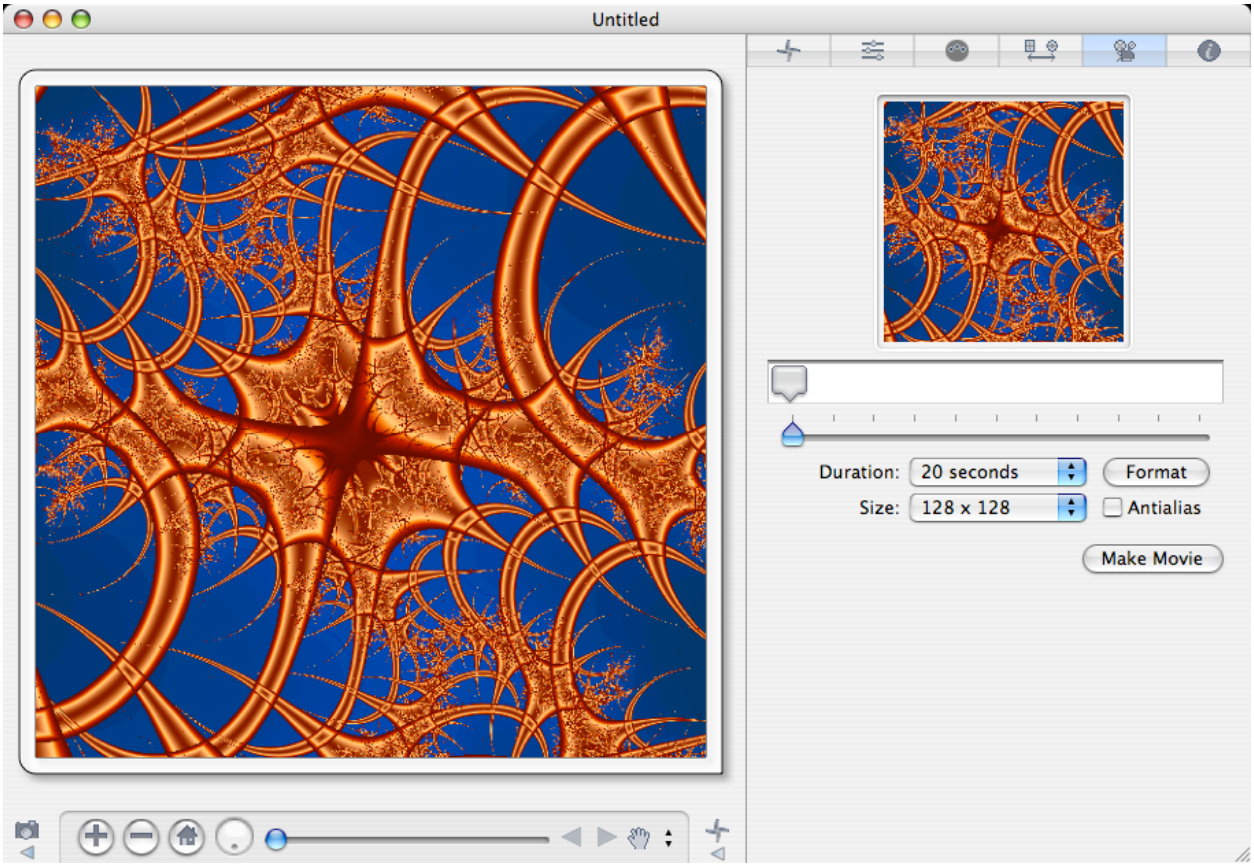
You now have all the basic tools to experiment with quadrium and create some amazing images!

Lesson 5 - Making Movies

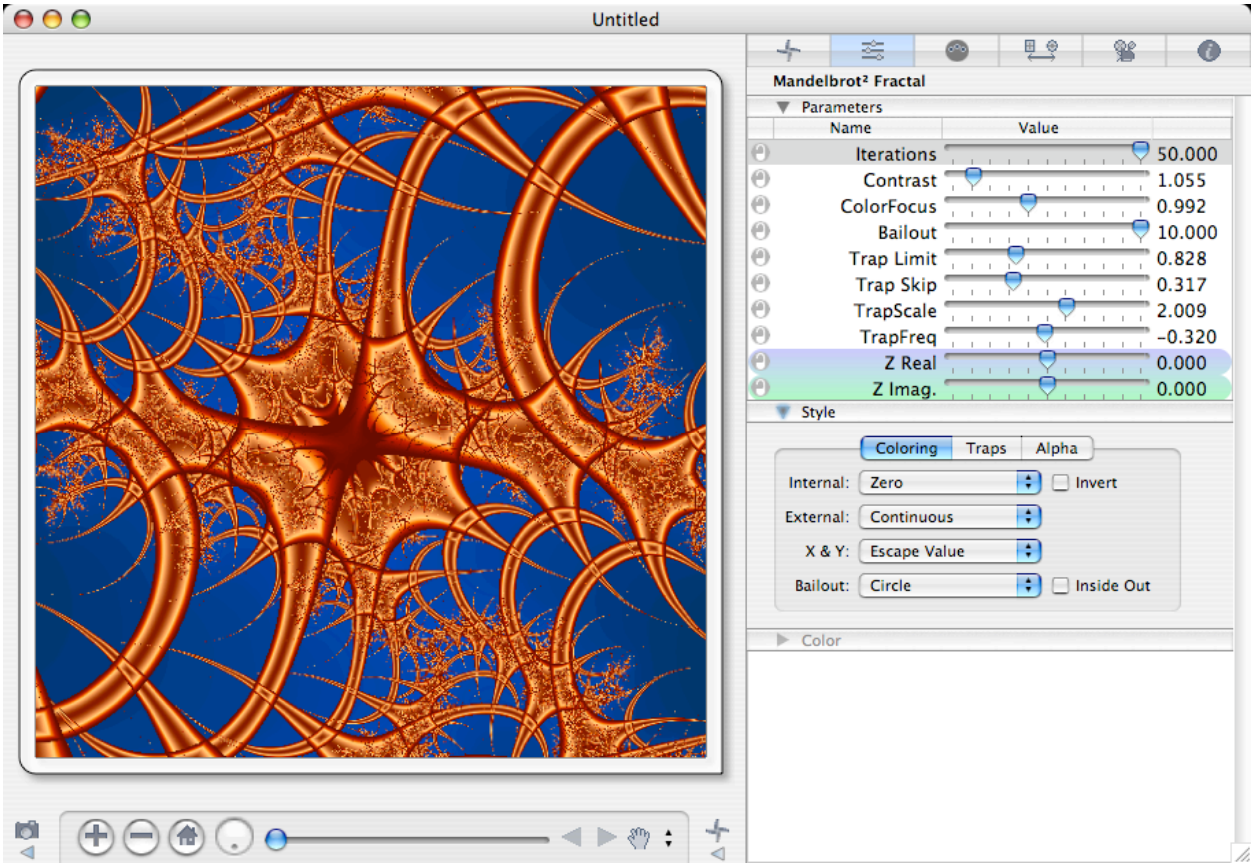
In this lesson we're going to make a simple fractal movie. From the "Starting Points" browser, select the "Fractal" category and choose the blue and brown fractal and expand the details tab:



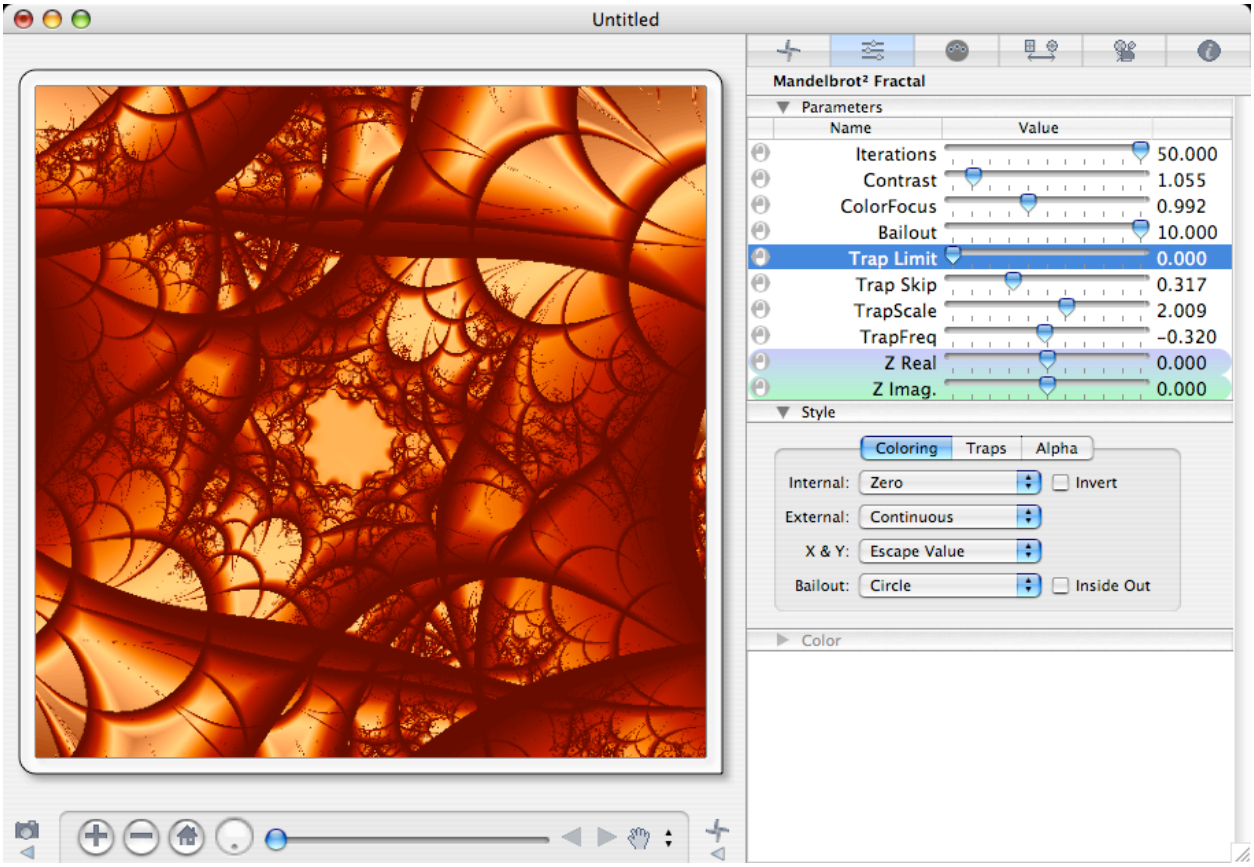
We're going to use this as the basis for our movie. This is going to be the first "frame" of the movie, so switch to the movie tab and drag the current image and drop it in the "time line" area at the very left (remember, you can drag the current image by dragging from the white border frame around the image):



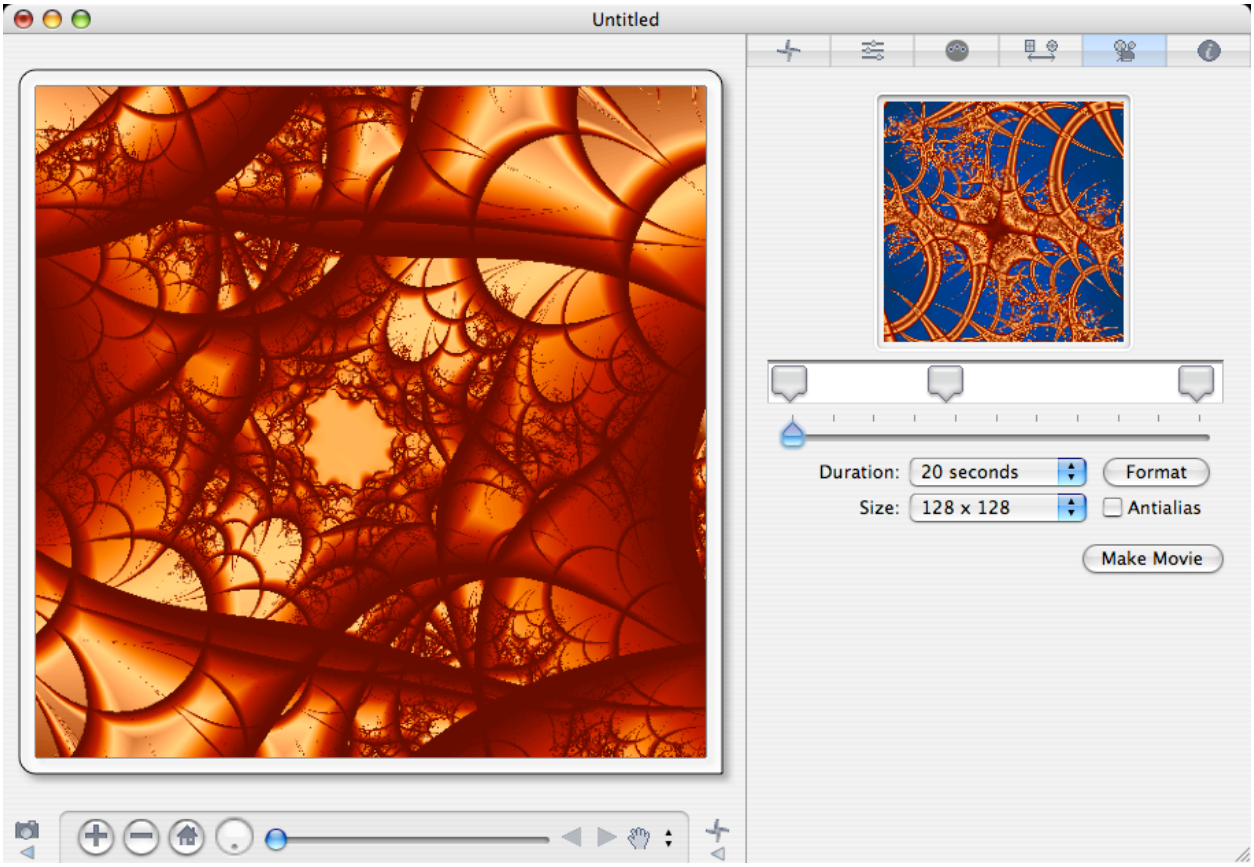
We're going to want the movie to end up being back at this state (so we can play the movie in a loop) so make a second key frame by dragging the current image and dropping at the end of the time line. Switch back to the structure view and double click the node labeled "Mandelbrot2", giving us the parameters for that:



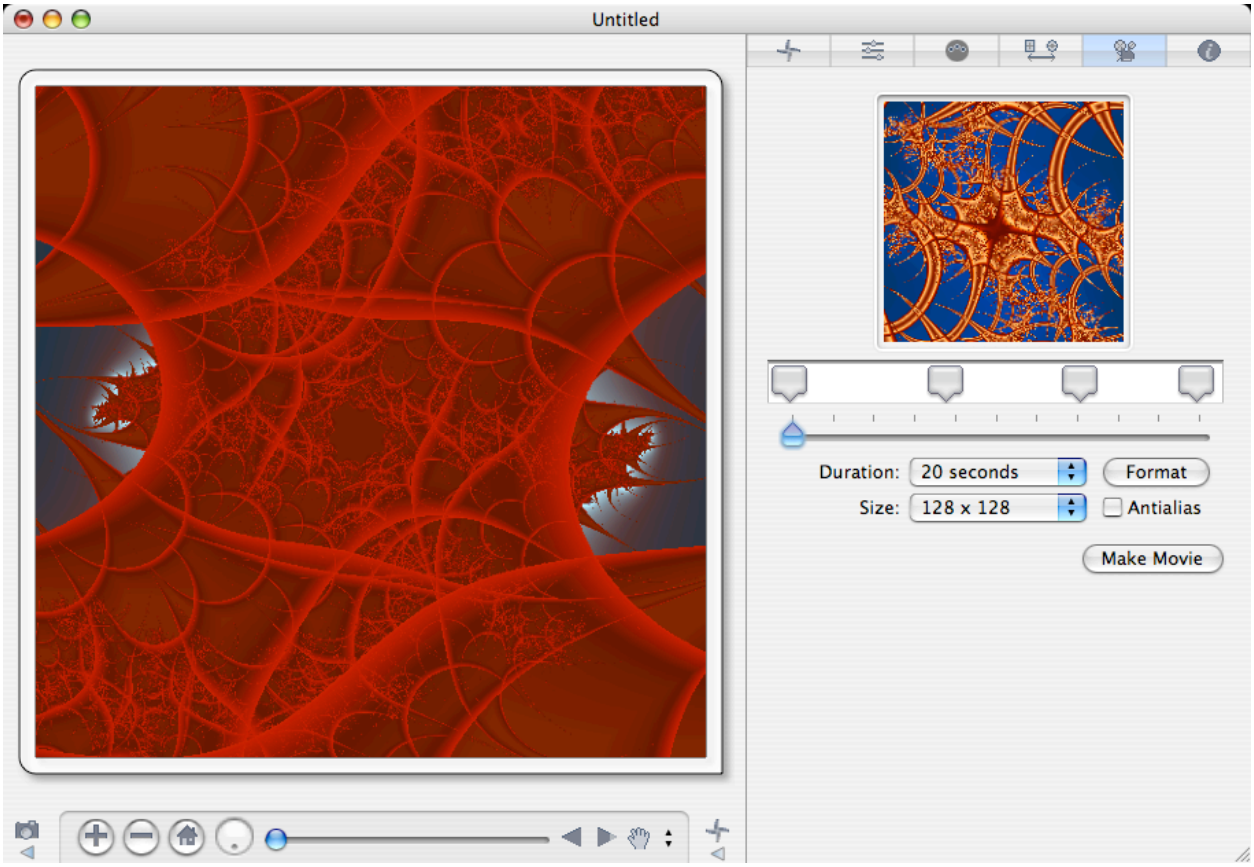
We want to change the “Trap Limit” value down to 0:



We’re going to use that as the second frame of the movie, about a third of the way along the time line:



Our third point of frame is going to have a different color, so switch to the structure panel and double click on the “gradient” node. Change “Midpoint” to 1.0. We’re also going to change the point of view slightly - click (and immediately release) the “-” zoom button 5 times. We’re going to use that result as the third frame:



Now, drag the slider from the left to right - you'll see the edges of the traps overlapping, and then the colors will begin to change as the object zooms out. Once you've gone past that third key frame, the object will zoom back in, the colors will change, and the trap edges will move - all at once.

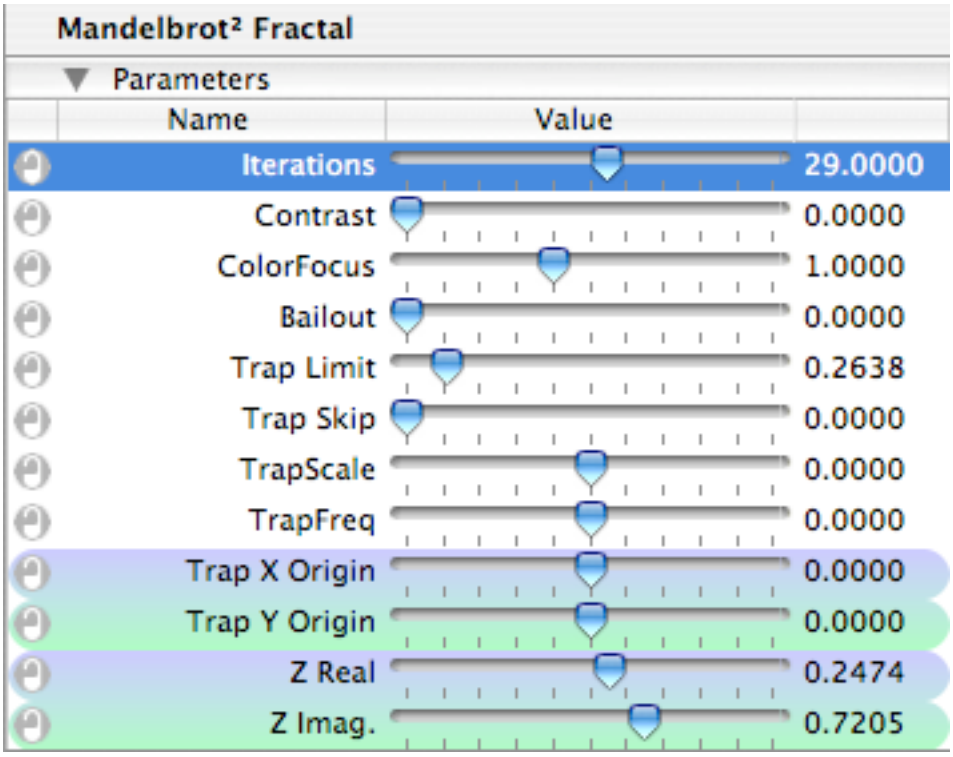
To make a movie, just click the "Make Movie" button (we'll go with the current settings). The "Job" window will appear, showing the movie being rendered. After it completes, it will be launched in QuickTime Player, ready to go:

(Note that until you register, everything you render, be it movie or high resolution file, will include the “DEMO” watermark).

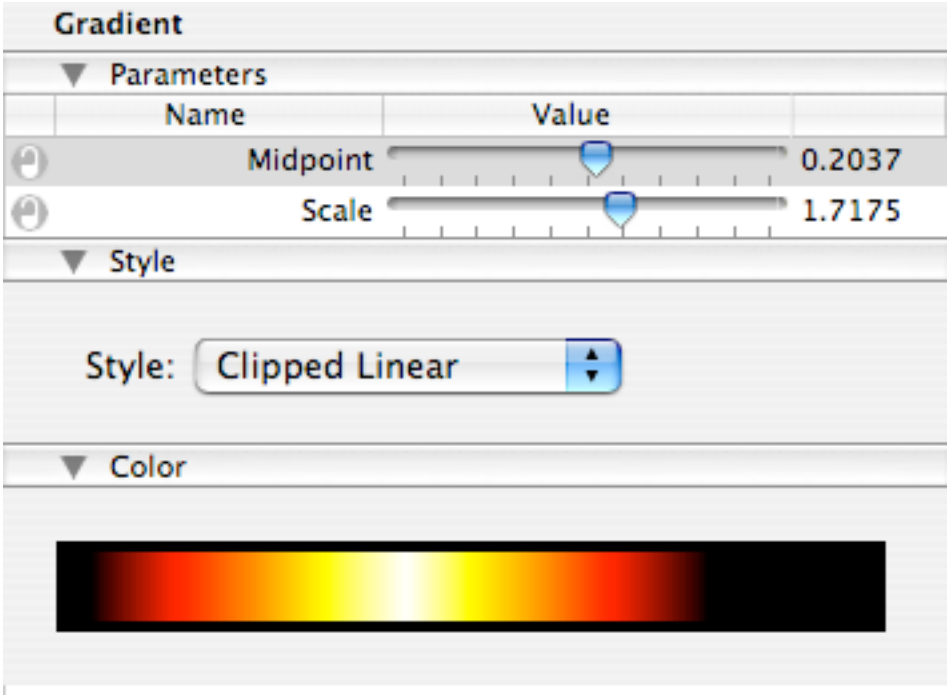
Lesson 6 - quadrium 2.1 Features

quadrium 2.1 includes some new features and a whole lot of enhancements. This lesson will touch on some of fractal enhancements. The final result of this lesson will be the following image:

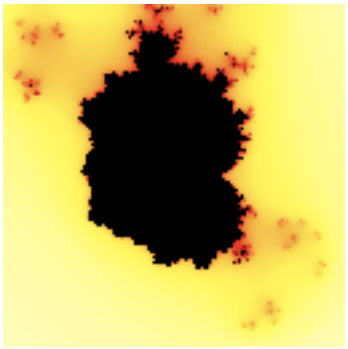
We'll start with a simple "Mandelbrot fractal with gradient, such as found in the starting points. The settings for the fractal should be set as follows:



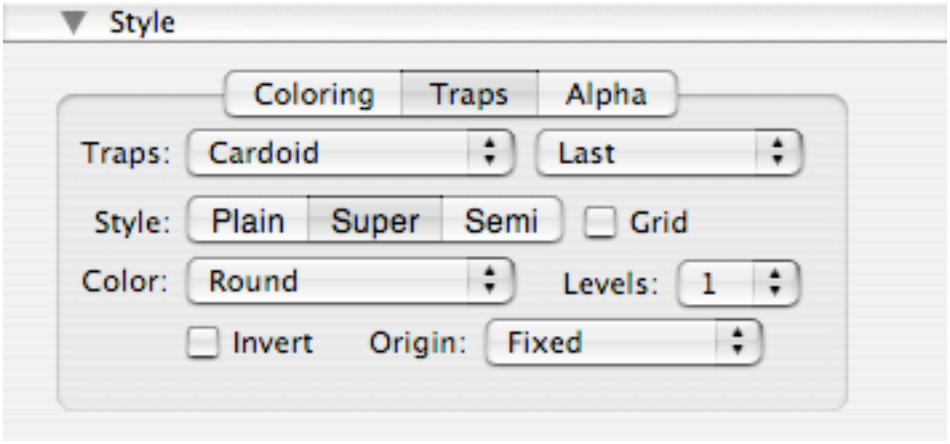
Set the internal coloring to “solid” and external to “continuous”. The gradient should be set up as follows:



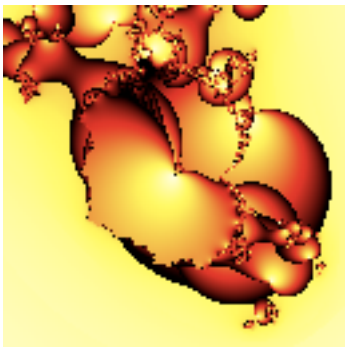
The result will be the following, not very intersesting, image:



To make this more interesting, we'll turn on an orbit trap (originally, the image was created by first turn on an orbit trap and then changing the fractal origin accordingly). So select the Mandelbrot node and switch to the traps tab in the style panel, and choose "Cardiod", "Last", "Super" and "Round" for the color

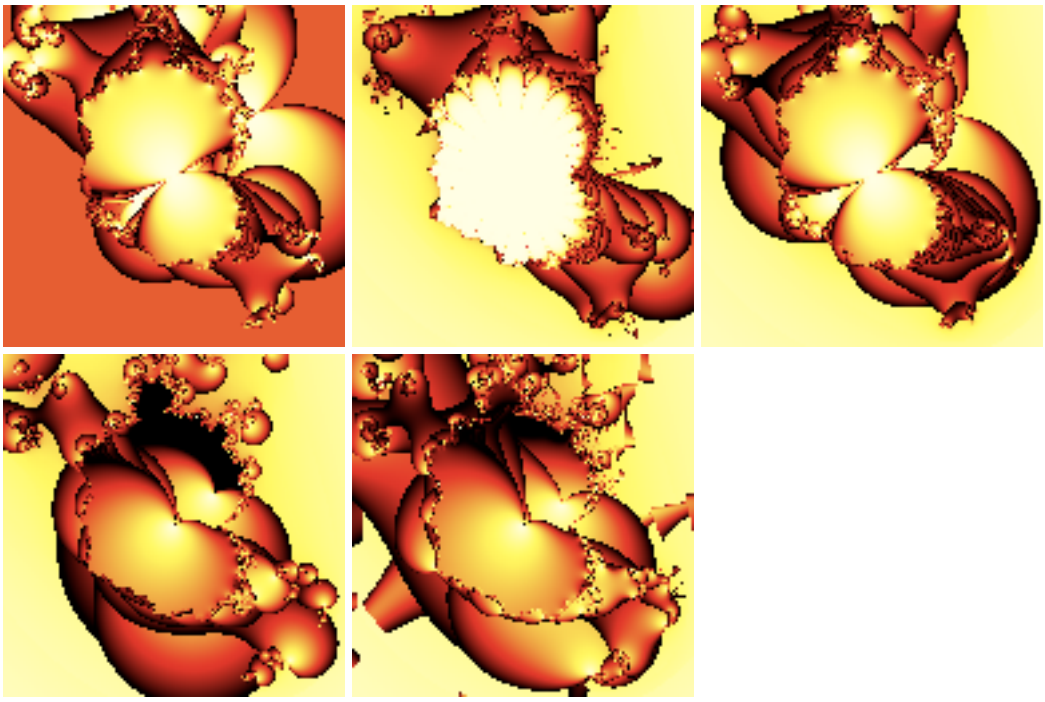


The result is this:

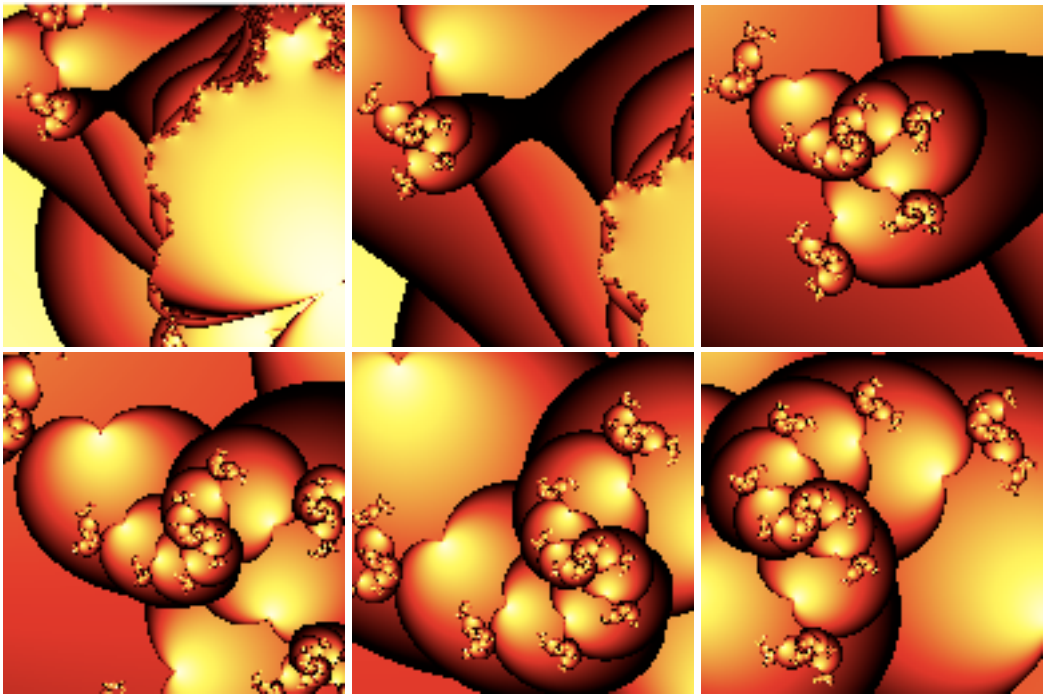


One of the new features in 2.1 is a series of new trap types - we're using one of them - the cardiod (which is a slightly bean shaped object, and works well for making "blobby" traps). More importantly, we can now alter where the trap is located. To make a change for the center of the trap is, use the "Trap X Origin" and "Trap Y Origin" sliders. We can also have the trap "move" from iteration to iteration - by default, the trap is fixed in space

(at whatever that origin value you set it). By changing the “Origin” popup, we can have it move with the fractal. Below is what you should see as you change this value:



Note that in some cases, truncated shapes appear - these will go away if you increase the trap “bailout” slider. We’re going to pick the “Product” setting (the third image above). We then want to zoom in that small appendix sticking on the side of the image near the top left corner:



The last step was to rotate it so that the rotation knob is approximately at “11 o’clock”. If you hover the mouse over the view, you should get a tool tip with the following coordinates:

Center = (-0.4257492, -0.2708803)
Scale = 44.93608: 1
Rotation = 156.976°

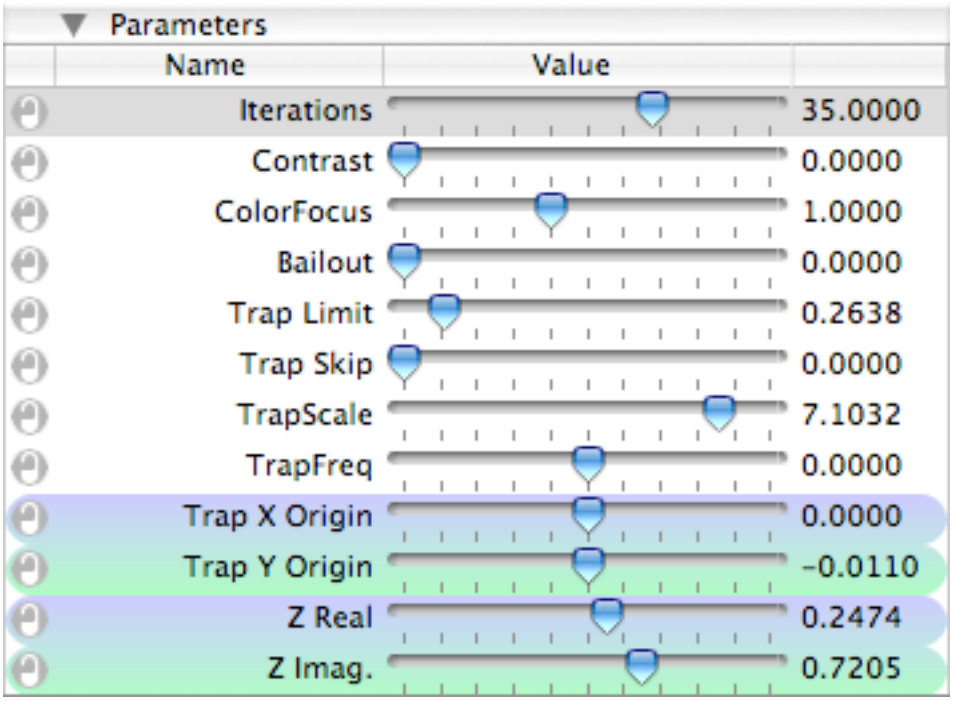
An even easier way to accomplish this is to select the text fragment below, and drag and drop it into the image view:

```
viewpoint:{ centerx = -0.4257492; centery = -0.2708803; magnification = 44.93608;
rotation = 0.9477275; }
```

This takes advantage of the new “textual parameters” feature of quadrium views - you can obtain this textual information using the contextual menu item “Copy View Point” that appears when you control/right click on the main image view. This is primarily useful for tutorials like this, but can also be used to find locations in common fractals (”Check out the bug shape in the Mandelbrot set at viewpoint:{centerx = -0.6033964; centery = -0.1580545; magnification = 510.9423;}”).

The current results are mildly interesting, but not quite there. So we’ll play with the trap settings and see what we can get.

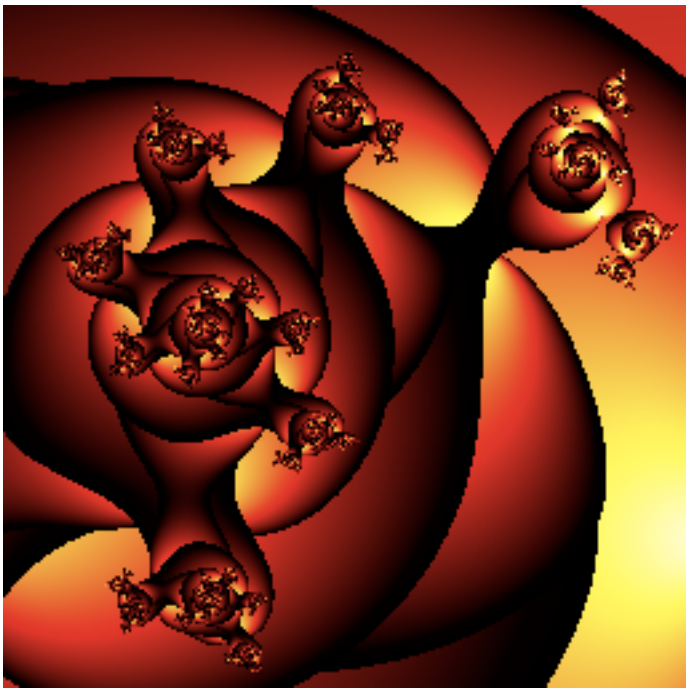
Change the sliders as follows:



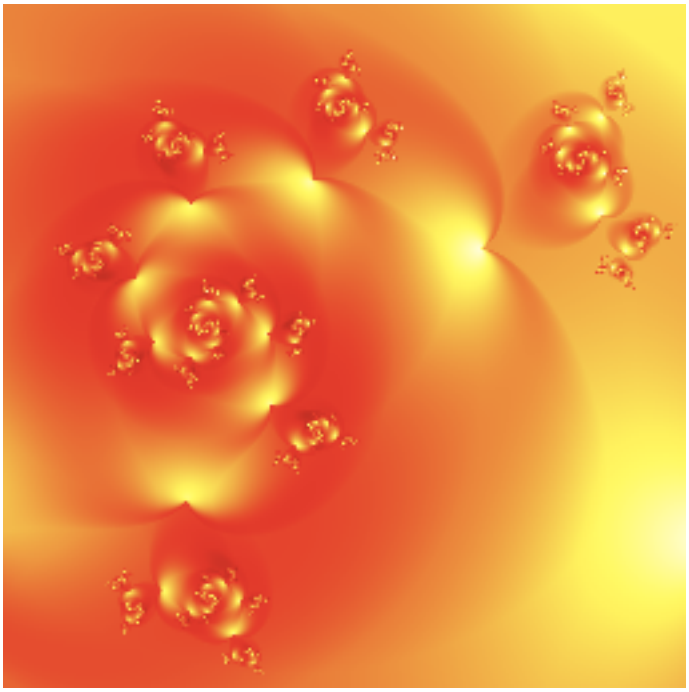
Just as the textual parameter for a view can be used, so to can the values of parameters of a node - to set all these values, select the text below and drag and drop it onto the node:

```
geneparameters:
{
    bailout = 0;
    colorfocus = 1;
    contrast = 0;
    iterations = 35;
    trapfreq = 0;
    traplimit = 0.2638263;
    trapscale = 7.103224;
    trapskip = 0;
    trapxorigin = 0;
    trapyorigin = -0.01095325;
    zimag = 0.7205006;
    zreal = 0.2474326;
}
```

This changes the origin of the trap, which causes all the blobs to move. Changing the trap scale actually causes this particular trap to spin, changing where the “dent” is. Between these two changes, we get this:



Obviously is similar to our “target” image, and pretty interesting in its own right. At this point, we can take a slight diversion to play with some of the new trap modes. If we change the popup from “Last” to “Blend In”, we get this:



Notice how the sharp edges of the traps are blended away, making for smoother images. “Multiply” also produces an interesting result, though both would need more work to add some value balance to the resulting image.

After switching back to “last” we want to adjust the image to include a scale texture. We’re going to use that texture to adjust the value of the image, imprinting the scales on the result. We can drag a value adjuster and wire it up downstream of the gradient, but since this is a fairly common task, we can take advantage of another new feature in quadrium2.1 - AppleScriptability.

Since quadrium2.1 adds the ability to build images using AppleScript, we could write a script to do common tasks like this. Fortunately, there already exists one for this specific task - under the Structure menu, there is a submenu labeled “Scripts”. That submenu contains “Presets” which contains “Value Adjust Fractal Gradient”. Selecting this item will add the node and wire everything up (and yes, this actually causes the whole visible image to turn black).

Looking at that script we see it is actually pretty straight forward:

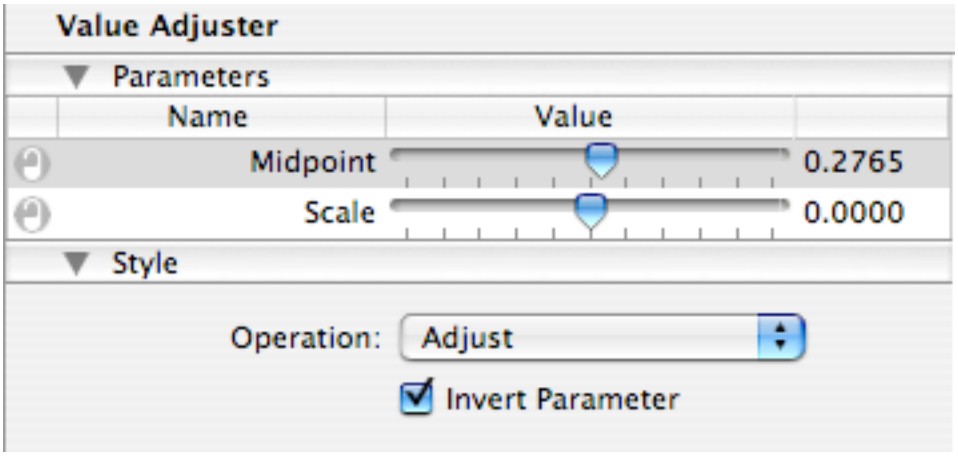
```
tell application "quadrium2"
	tell first document
		set fractalNode to first node whose kind contains "fractal"
		set gradientNode to last node whose name contains "gradient"
		try
			get name of gradientNode -- if not defined, we'll get an error
		on error
```

```
            set gradientNode to last node whose kind contains "ad-
                    juster"
        end try
        set adjNode to add node named "Value Adjuster" after gradient-
                Node
        connect fourth output of fractalNode to fourth connection of
                adjNode
        tell adjNode
            set value of (setting named "operation") to 2
        end tell
    end tell
end tell
```

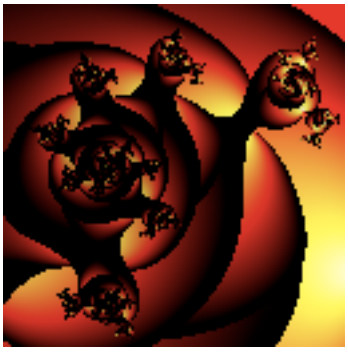
It basically just finds the fractal node (or and adjuster, if none found, it works with the last existing adjuster), and the last node named “gradient”. It creates a new adjuster, and wires the fractal alpha value to the adjuster parameter. The final step is that it set the operation to “Scale” (which isn’t what we’re going to use).

You can write scripts like these of your own, and they will appear the next time you launch quadrium2.1.

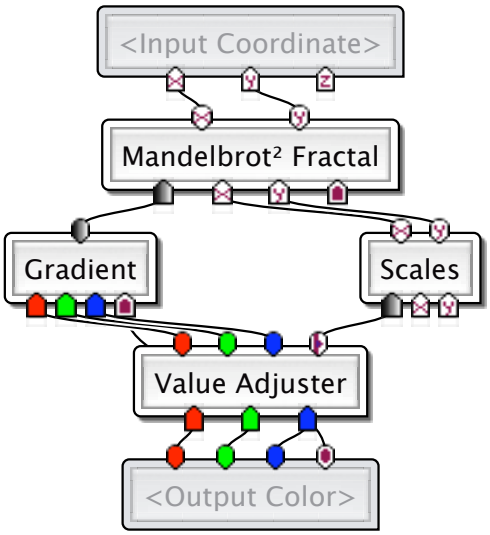
We need to make some minor changes to the adjuster node that was just created. So change the settings to reflect the following values:

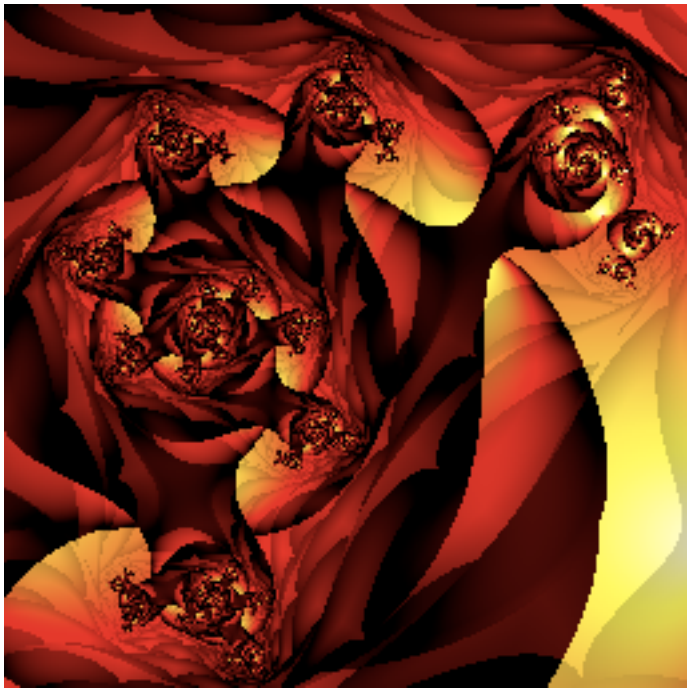


One of the useful minor additions to quadrium2.1 is that adjusters can automatically invert the parameter value (making it lighten where it otherwise would darken, without needing to insert an additional node to negative that value). Once we’ve done this, we are back to our last image (though slightly washed out). Normally, we’d change the alpha value coming out of the fractal to change this - for example, if we set the alpha value to have “traps: Fade First” it will result in this:

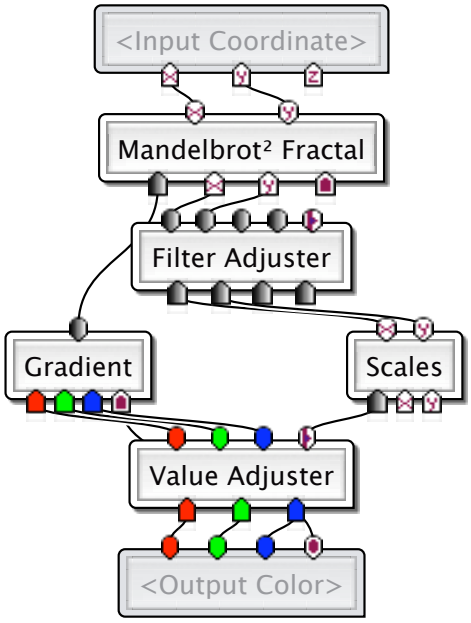


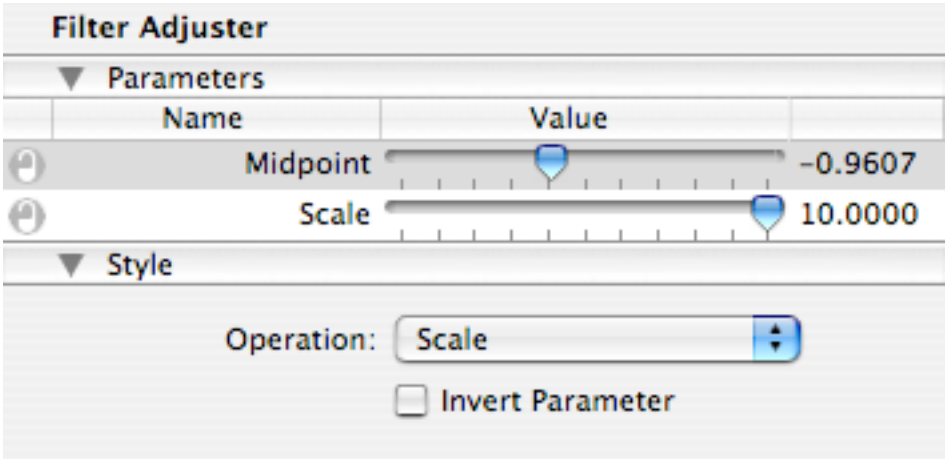
(where the “inside” is darker because it is a later trap, and our setting causes the first traps to be dark except that we’ve inverted it with the checkbox). Instead, we’re going to use a texture to drive this. So drag out the scale texture node, and hook it up to the X & Y “exit” values of the fractal and run the texture output to the adjuster parameter input, and make sure that the exit value for the fractal node is set to “Escape (Trap)”:

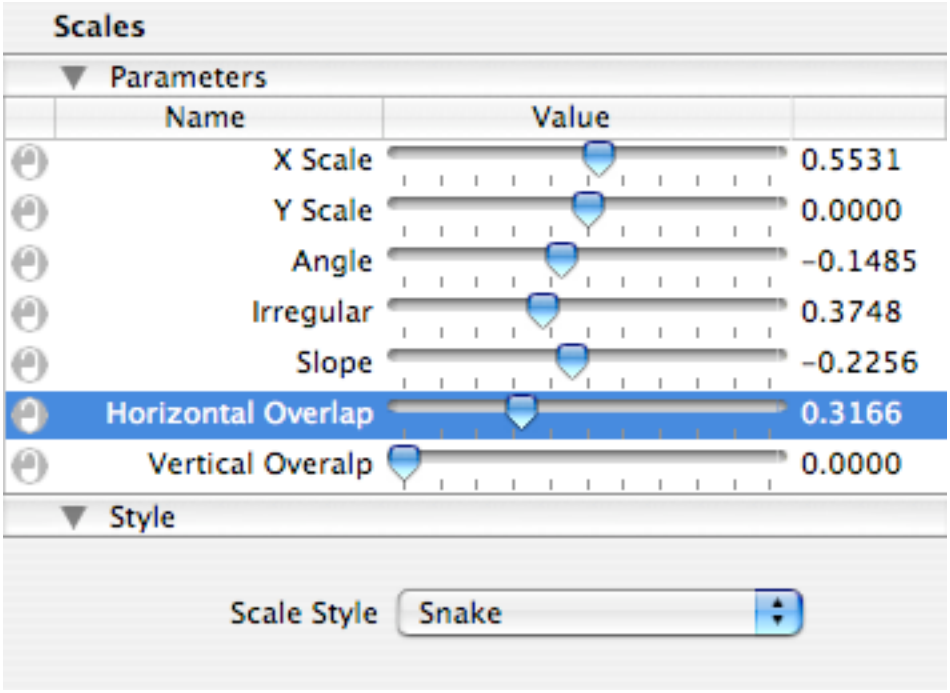




Obviously, the scales are the wrong size to be useful, so we can either just adjust the scales X & Y scaling factor, or, we can use a filter adjuster node to allow us to have more control over the result (a filter adjuster node works like a color adjuster, but adjusts each channel). Unlike just changing the scale factor on the scales node, we can effectively change the “origin” of the scales (since the “midpoint” slider value is added to our adjustment). So insert an adjuster filter node between the fractal and the scales node, and set them as follows:

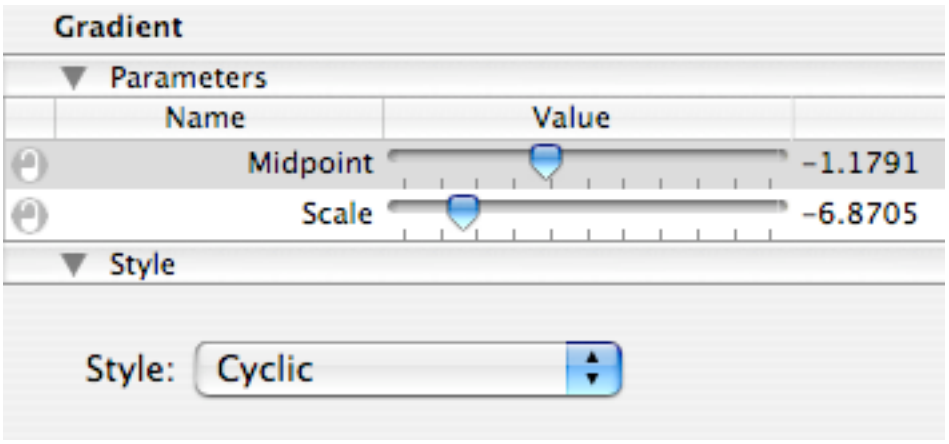

<Input Coordinate>
Mandelbrot² Fractal
Filter Adjuster
Gradient
Scales
Value Adjuster
<Output Color>


Filter Adjuster
Parameters
Name
Value
Midpoint
-0.9607
Scale
10.0000
Style
Operation:
Scale
Invert Parameter

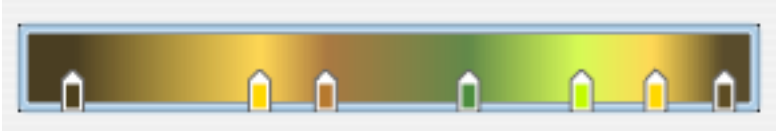


You can, of course, tweak the scales as desired - the irregular setting adds some natural variation in the scales, making them look a bit more like they are on a curved surface. At the very least, you'll probably want to switch to "snake" (which are pointier than fish scales), and tweak the overlap values.

Next, we're going to need to adjust the gradient to use less "flame like" colors. So change the settings as follows:



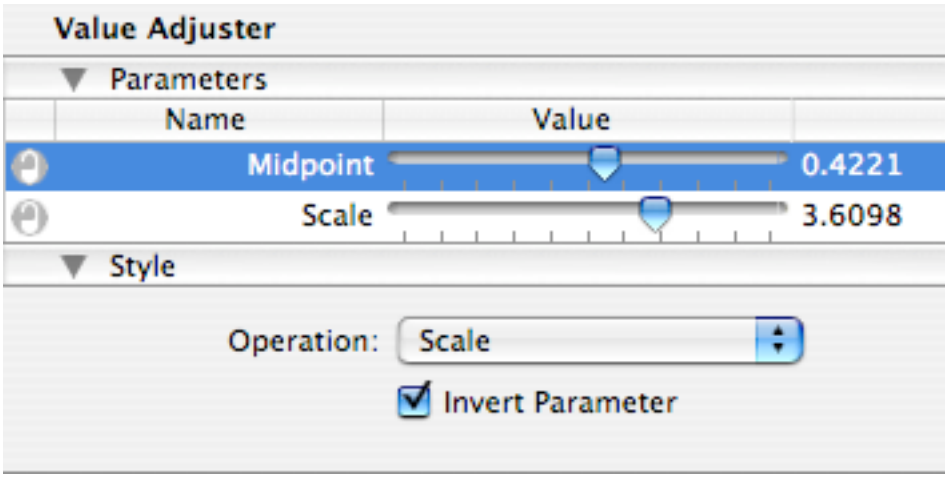
And change the gradient to look like this:



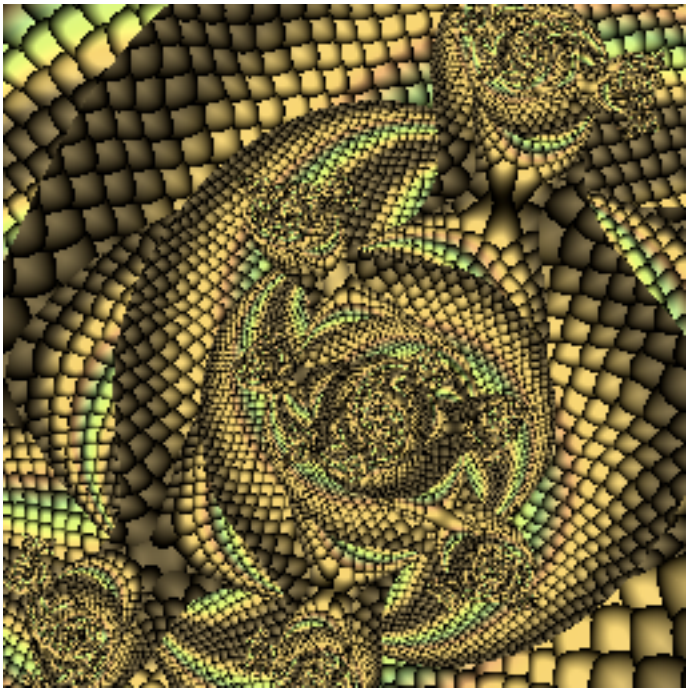
(The easiest way to replicate this is to use the color picker's "magnifier glass" mode to select a color from an existing place on the screen and use it to grab the color from this document - select from the color swatch inside the nub should give you a nice solid representation of the color).

One thing you'll notice is that with the scale set as it is, we end up with lots of stripes (or, roughly 6.8 stripes where there use to only be one stripe before). To fix this, we change the number of trap levels to 8 in the fractal nodes trap setting tab. The reason we did this is so that each level isn't exactly the same - the trap values are partitioned up into a series of 8 different value, and the gradients just under seven, so that mean each trap level will be slightly offset (in the gradient) from the last.

Now we just set up our value adjuster to adjust the gradient with our texture as follows:



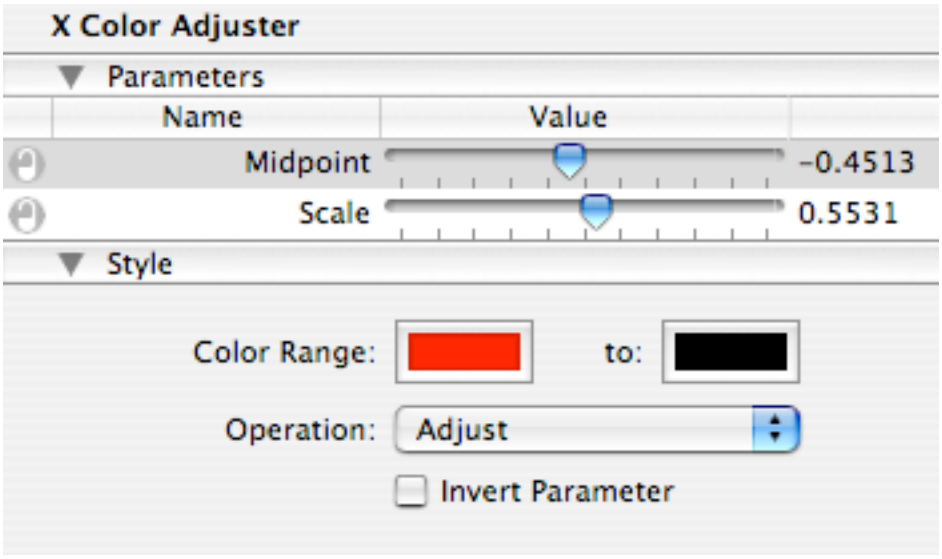
And the result should be something like this:



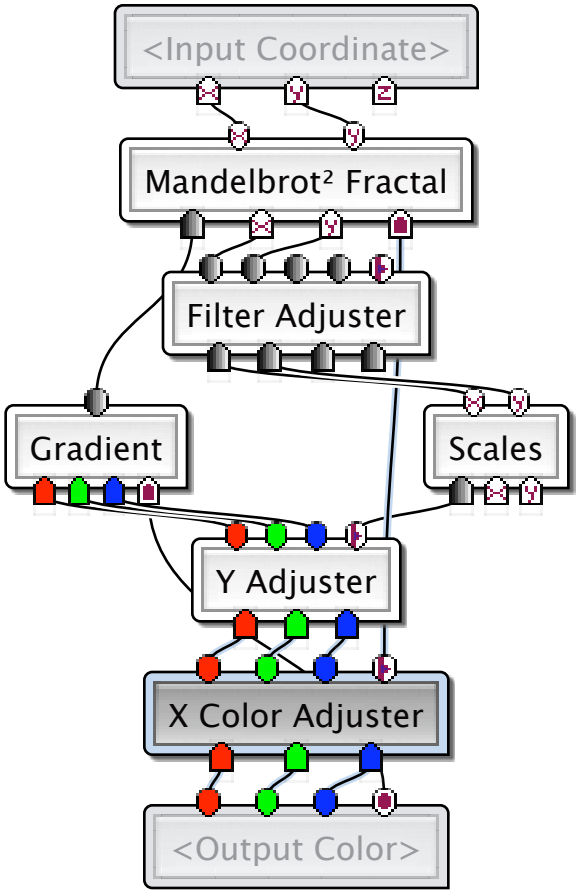
Really not so impressive. This is the point where we need to take an interesting image and fine tune it. There are lots of ways to accomplish this, we'll use a few here.

One of the first things we notice is that our colors are pretty washed out - we could change the gradient, or we can use a different adjuster. The Value Adjuster node we originally used does just that - it changes the "value" of the color, and doesn't impact the saturation of it (so while darks get darker, lights get brighter but not lighter). We'll switch instead to a "Y Adjuster" (which, to use a TV analogy, adjusts the brightness/contrasts as you'd see on a television, and, in fact, corresponds to the portion of a TV signal that gets received in a black & white television). So find the Y Adjuster and drag and drop it on top of the Value Adjuster - but hold down the command key while doing so. This will tell quadrium to attempt to copy over any similar parameters/settings from the old node to the new node (so if you look at the node settings, you'll see the identical slider and other settings.

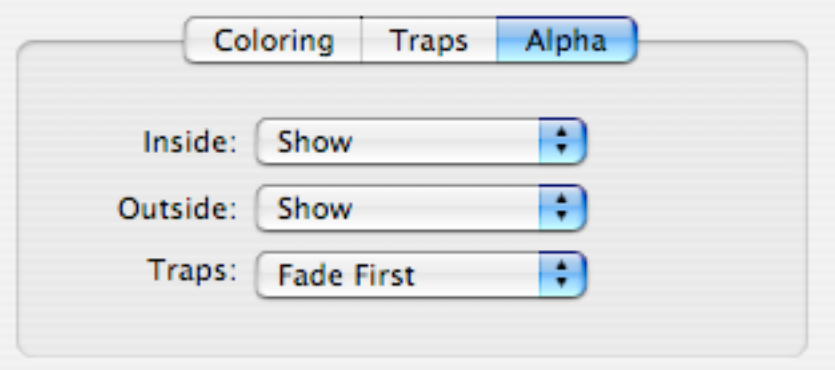
The next step is that we want the "lower" coils to be slightly different from the upper ones. So we'll add an "X Color Adjuster" (which adjusts between two arbitrary colors):



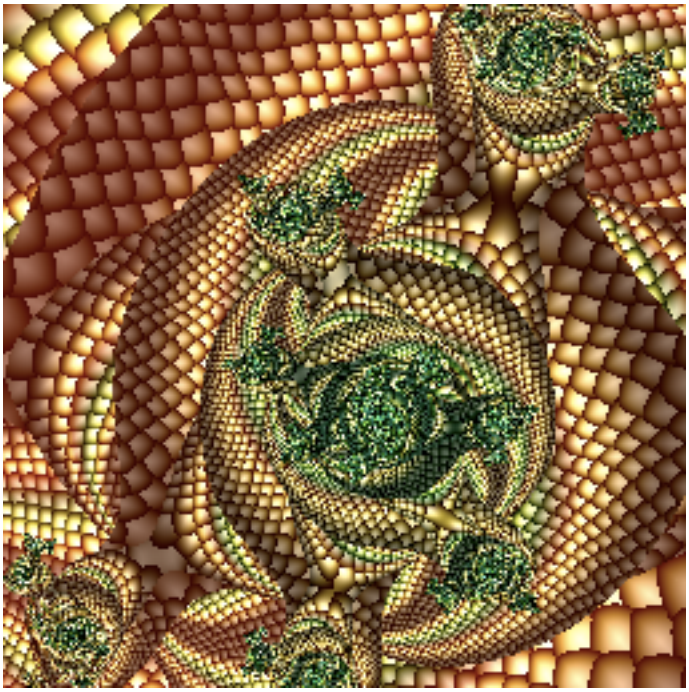
It will be downstream of the Y Adjuster and have it's parameter node connected to the alpha output of the fractal node:



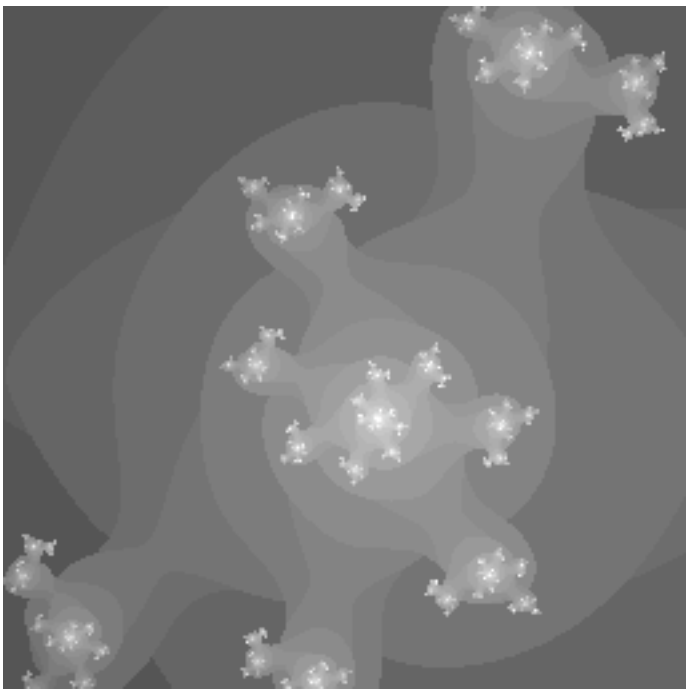
When you do this, the image will turn an ugly green color. The final key now is to set that alpha channel to be different for each different “coil” - to do this, double click the fractal node, and switch to the “Alpha” tab and set the alpha to have traps “Fade First”:



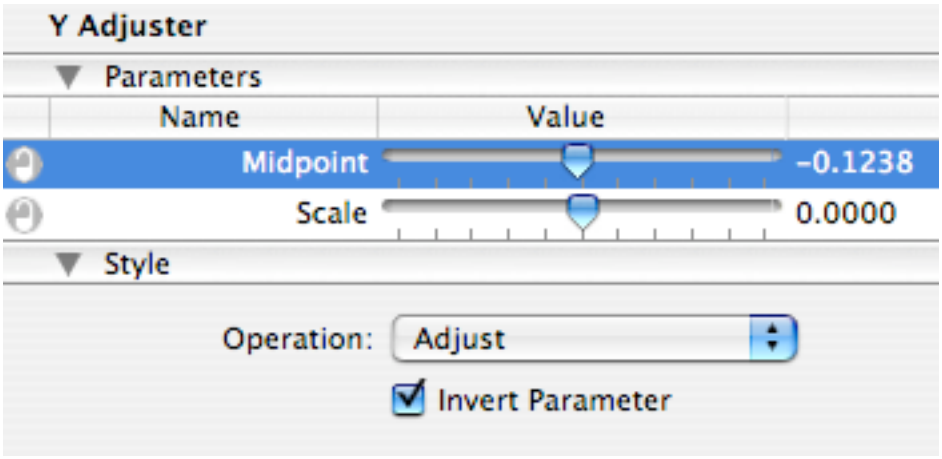
The resulting image is a bit more refined, but not quite there yet:



If you want to see exactly how that alpha channel coming out of the fractal is behaving, we can use another of the new features - the “output peek” ability. If you command-click on an output node of a fractal, it will show you the current values of that node’s output, as if you had connected it up to the RGB inputs of the output node. Doing this now on the alpha channel of the fractal node will show you this:

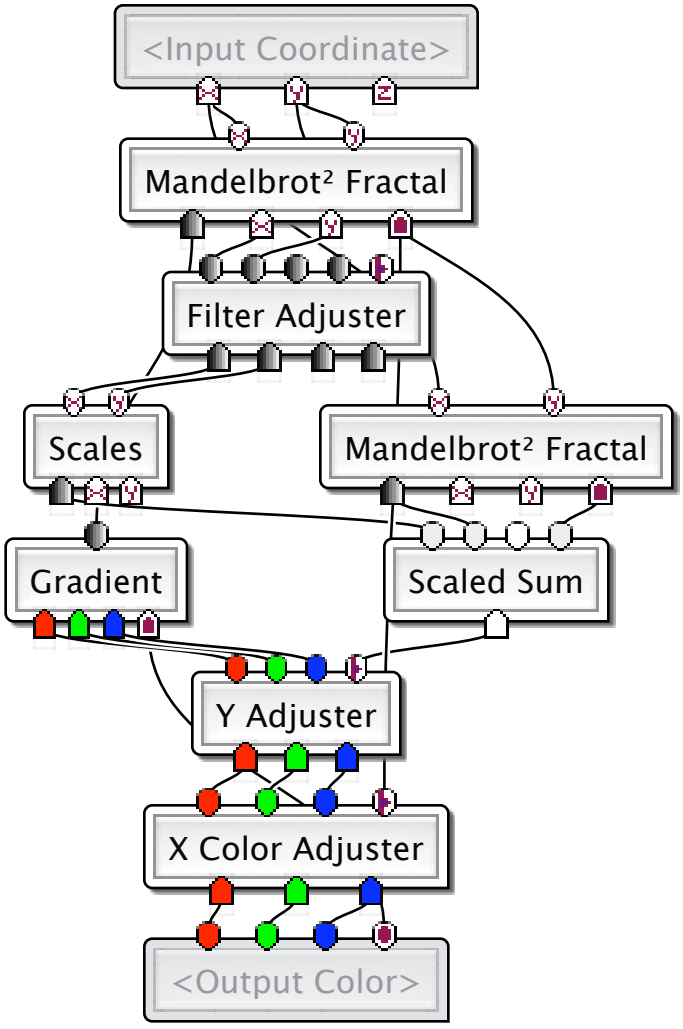


Next, adjust the values of the Y adjuster thusly:

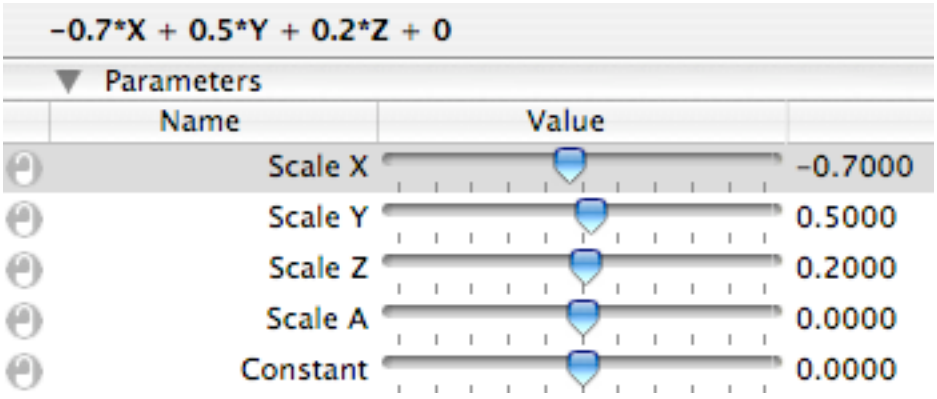


This provides a better range of values. It's still not perfect, but it's definitely better. It would be nice if we could figure out how to add some curvature into those coil - perhaps if we used some other sort of coloring algorithm of the fractal to "tint" the results, it would work. So select the fractal node, and then choose "Split Selected Node" from the "Structure" menu - this makes a second copy of the node, with identical inputs and settings.

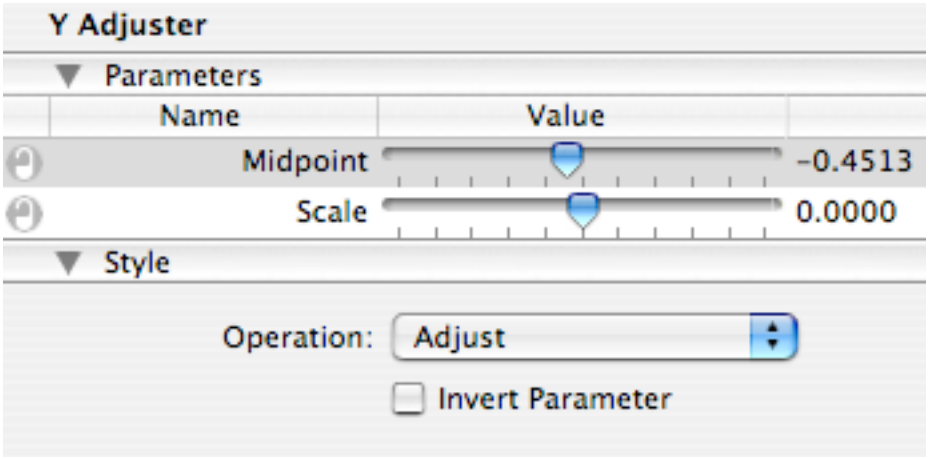
Now we'll want to combine whatever coloring we pick with the snake scale texture, so we'll use the "Scaled Sum" combiner. Since it might be useful, we'll connect its inputs with the output of the texture, the main output of the fractal, and the alpha of the fractal as well:



So now all that's left is to adjust the various settings. Our scaled sum settings looks like this:



and since this actually makes the scale texture negative, we need to change the Y Adjuster as well:



The changes we'll make to the second fractal node are to change the trap levels to 1 (instead of eight, which we did earlier to get the stripes to be slightly offset). If we didn't need to make that, we wouldn't ultimately have needed to split the fractal node. For fun, however, you can play with some of the parameters of that second node and see how the image gets "ghosts".

Otherwise, change the aspect ratio of the image (using the size settings of the info tab) to 6" x 8" and we're done: