quadrium2
Reference

All Nodes

Below is a comprehensive alphabetic list of all nodes available in quadrium. After this, we have it broken down into categories (the same that are used in the browser).

2D Affine	1
2D P-Rotator	1
2D P-Skew	1
2D Rotator	1
2D Skew	1
Abs	2
Abs Max Reducer	2
Abs Min Reducer	2
Abs Sine	2
AbsMax Blend	2
AbsMin Blend	3
Add Max Blend	3
Alpha Blend	3
Angled RGB Lights	3
Angled RGB Planes	3
Apollonian Packing	4
Averager	4
Barnsley Fractal	4
Bit And	4
Bit Or	4
Blackness Adjuster	5
Blorp Lens	5
Boolean Blender	5
Bump Map	5
Bump Normal	5
CMYK to RGB	6
Cartesian→Polar	6
Celtic Texture	6
Center Warp	6
Charm Fractal	6
Checker Warp	7
Chladni Pattern	7
Chunky Pixelizer	7
Clipper	8
Clustered Shapes	8

Filters

Filters are nodes that operate on multiple channels independently of each other. You can use one, two, three or all four of the input channels, and the respective output.

Combiners

Combiners are nodes that operate on multiple channels and combine them to a single output channel. For the most part, they are order independent (you can connect the input to the first or last channel and get the same output), though there are a couple of exceptions. Most also have per-channel parameters that can be adjust, as well as overall parameters (such as a scale or offset).

Complex

Complex nodes are complex in the sense that they use complex math, but are otherwise actually usually fairly simple. Complex math involves using a pair of numbers - a 'real' part (representing ordinary numbers) and a 'imaginary' part (representing a multiple of *i*, which the imaginary value representing the square root of -1). If this meaningless to you, don't despair - think of these nodes as nodes that operate on pairs of channels (such as a 2D (x,y) coordinate pair) and produce the same thing. Of particular usefulness is the *Complex Inverter* node which distorts things such that what was infinitely far away is moved to the center (and vice versa).

Adjusters

Adjusters are nodes which adjust RGB colors to produce another RGB color. These are useful to add subtle shading to an image, or otherwise manipulating the image to add details (such as making certain areas darker to represent distance).

Blenders

Blender are nodes that combine a pair of RGBA channels to produce a third RGBA set of channels. This can be often done to superimpose two different images on each other. Blender nodes can also often be used as series of 4 parallel mini-reducer nodes, since the channels are usually handled independently.

Warpers

Warpers are used to convert a coordinate space into another space, often distorting things in the process in a wide variety of ways. In general, however, the result is "continuous" - meaning that the result is similar to taking the image on a sheet of rubber and stretching & twisting it (and not cutting it apart into different pieces, which *Tilers* do).

Tilers

Tilers are similar to *Warpers* but instead of producing a smooth, continuous result, a tiler will break the original coordinate space into smaller pieces and reassemble them in a different way, often repeating these pieces. They sometimes are seamless (like you'd find on the pattern on desktop of your computer monitor), other times there are visual edges between them.

Resamplers

Resamplers are similar to *Tilers* but take very small parts of the original coordinate space and convert them to larger areas in the result. If one were to take a small bitmap in a painting program and stretch it out large, resulting in a chunky coarse image, that's the sort of thing a resampler does.

Textures

Textures are general purpose nodes that take a 2D (or sometimes 3D) coordinate and convert it to a grayscale value that looks like something (sometimes a real world object, other times not). These are often the building blocks of an image - space is altered using a warper or tiler, it runs through a texture to produce the basic image, and then some colorizer such as a gradient to produce the result.

Vectures

Vectures are like textures but instead of producing a grayscale value, it produces a 2D value. This usually represents something like a direction. This result can be used to add subtle shading or highlighting to an image.

Colorizers

Colorizers convert something into a full color RGBA result. This could be as simple as a a single solid color, or the frequently used Gradient node (which converts grayscale data from a texture into color). There are also some colored versions of texture nodes which directly produce colors instead of grayscale values.

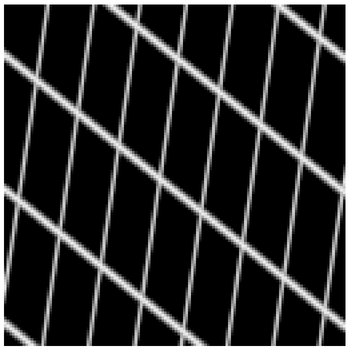
Fractals

Fractals, for the most part, refer to nodes that produce a self-similar image - one that looks similar at different magnification levels. Most fractal nodes are what are known as "escape function plots" or "limit functions". Basically, a starting value is taken from the input channel (an (x,y) coordinate pair). That function is evaluated with that input, which produces a new coordinate pair.   This is repeated as many times as the *iteration limit* is set for (so a high value on this parameter will take longer). If that result ends up being close to (0,0) it is assumed that at that point the function is stable (and on the 'inside') - if the value ends up a long ways from (0,0) then it is assumed that the function value has "escaped" (and on the 'outside'). The number of iterations it takes to determine if that starting point results on the inside or outside is used to determine the final grayscale value. This result is often adjusted using a wide variety of techniques to add additional shading and structure to the image.

For more details on how to work with escape fractal, see Appendix B.

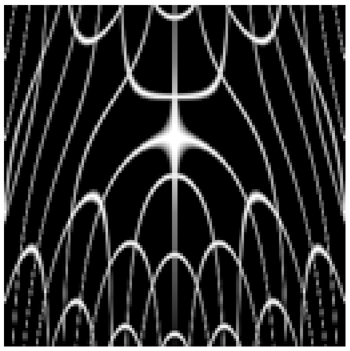
Noise

Noise nodes are nodes that are based on random noise, and so give unpredictable results. Most noise nodes include a "phase" parameter, which can be adjusted to give a completely different set of random results. Note that these results are actually consistent between documents and sessions (and even across machines), so if an image is created using a noise node, it will produce the same image everywhere.



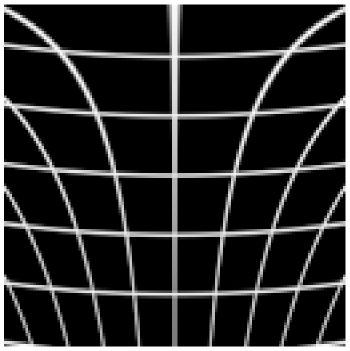
2D Affine

Provide generic 2D affine transformation - any sort of combination of scaling, transformation, rotation, and skewing is possible.



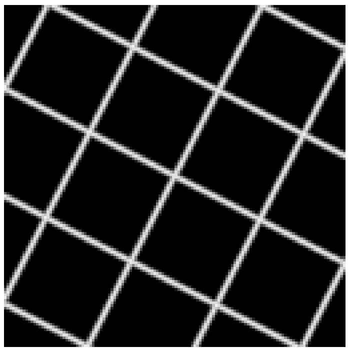
2D P-Rotator

Rotates the 2D coordinated controlled by a third, parametric, input channel.



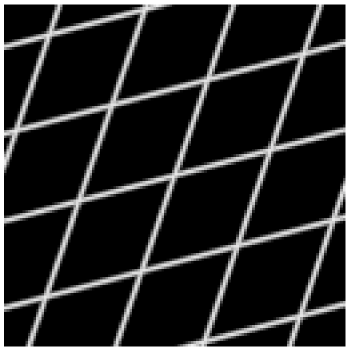
2D P-Skew

Skews the 2D coordinated controlled by a third, parametric, input channel.



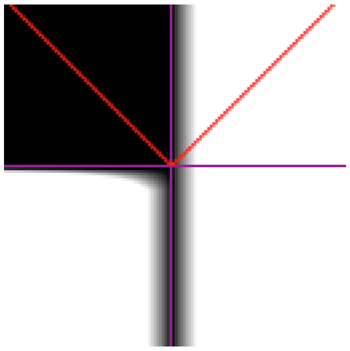
2D Rotator

Rotates the 2D coordinates clockwise or counterclockwise.



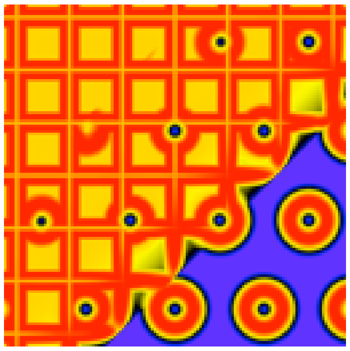
2D Skew

Skews the coordinate system, skewing each axis independantly.



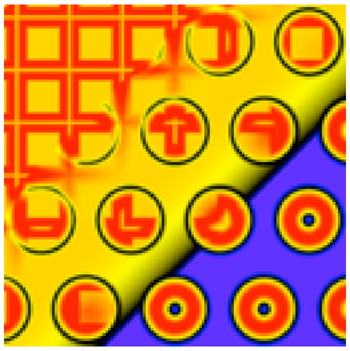
Abs

Find the absolute value of each channel.



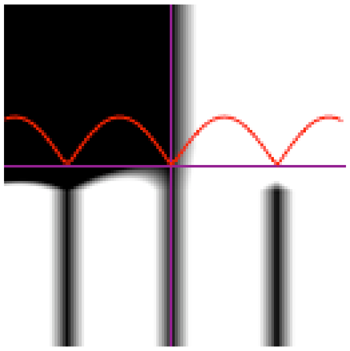
Abs Max Reducer

Takes the maximum of the absolute values of all four input channels as the output channel.



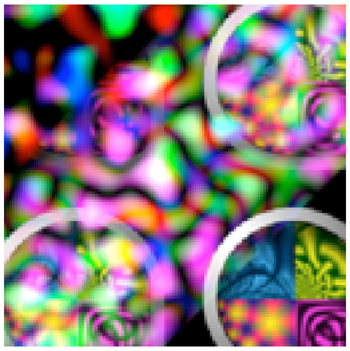
Abs Min Reducer

Takes the minimum of the absolute values of all four input channels as the output channel.



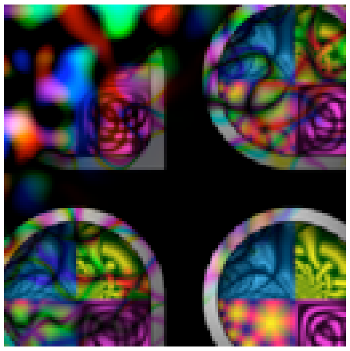
Abs Sine

Calculate the absolute value of the sine of each channel, scaling both the frequency and amplitude, and adjusting the phase of the result. Note that this produces a series of 'humps'.



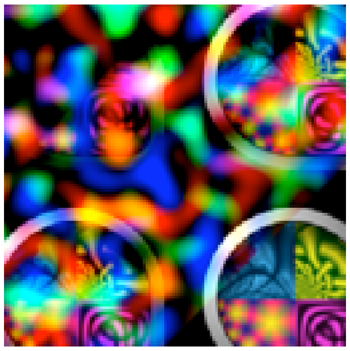
AbsMax Blend

Blends each output channel as the maximum of the two corresponding input channels.



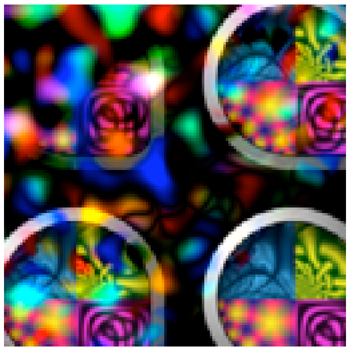
AbsMin Blend

Blends each output channel as the minimum of the two corresponding input channels.



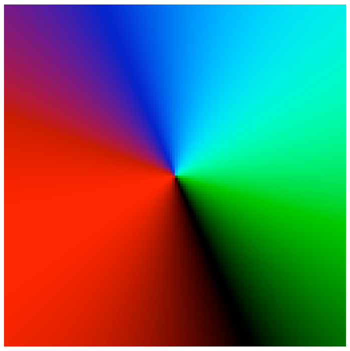
Add Max Blend

Blends each output channel as the maximum of the left corresponding input channels and the weighted sum of the right and left channels.



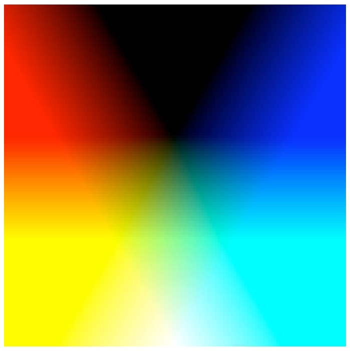
Alpha Blend

Blends each output channel as the a weighted balance between the two corresponding inputs, utilizing the alpha channel of each as well.



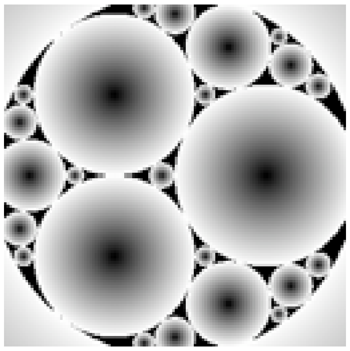
Angled RGB Lights

Creates colors based on overlapping cones of pure primary colors from the origin.



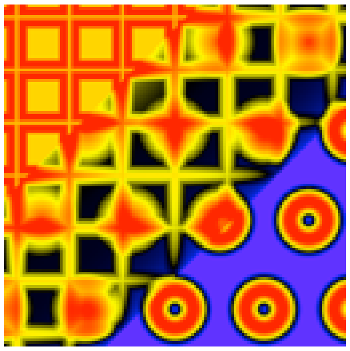
Angled RGB Planes

Creates colors based on overlapping planes of pure primary colors.



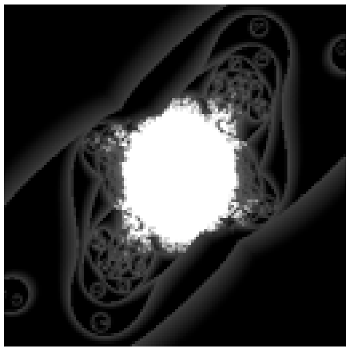
Apollonian Packing

Packs a series of spheres inside other spheres so they just touch the other spheres (and then the gaps are filled in with more spheres).



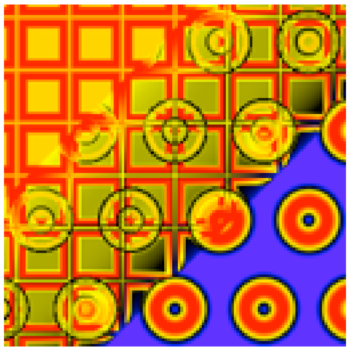
Averager

Combines the four input channels to produce a weighted average of them.



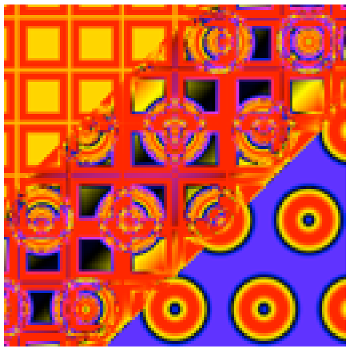
Barnsley Fractal

Creates an iterated escape fractal based on the Barnsley function.



Bit And

Combines all four input channels using a bitwise boolean AND, which gives a 'digital' look to the results.

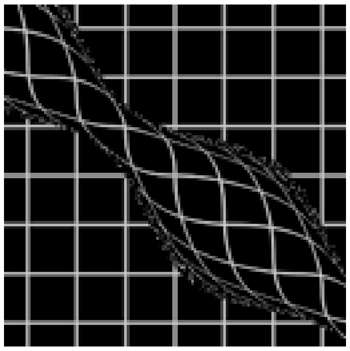


Bit Or

Combines all four input channels using a bitwise boolean OR, which gives a 'digital' look to the results.

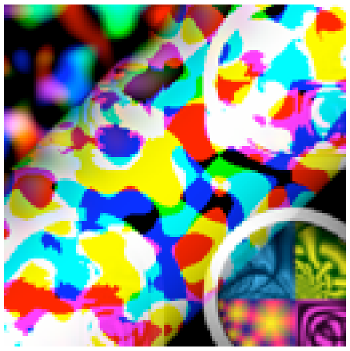
Blackness Adjuster

Adjusts the black component of the input color (by converting RGB to CMYK, adjusting the K, and converting back to RGB).



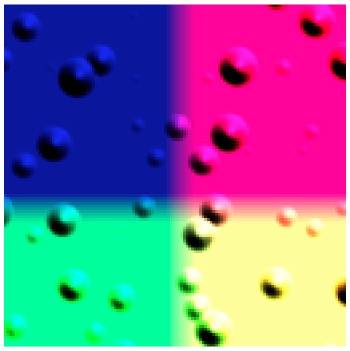
Blorp Lens

Warps the coordinate space as if viewed through a rod with sine-wave 'blorps' on it.



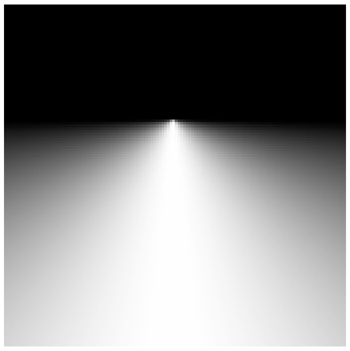
Boolean Blender

Compares the each pair of input channels to produce either 1.0 or 0.0 depending on the comparison result.



Bump Map

Takes a 3D vector and color and applies bump-mapped lighting for 3D style textures

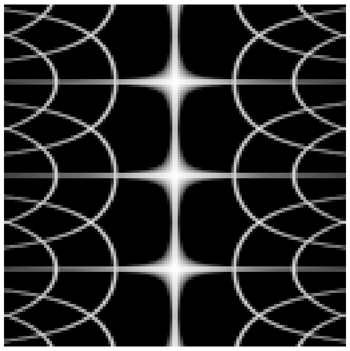


Bump Normal

Calculates the a value to simulate a light shown at an angle against the surface whose normal is the input (such as the output of a vecture and a height map), which can be used in an adjuster or blender for lighting effects.

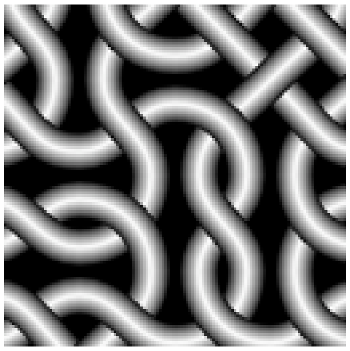
CMYK to RGB

Convert CMYK color space to RGB.



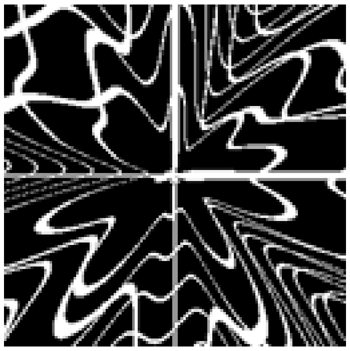
Cartesian→Polar

Converts cartesian coordinates (X,Y) into polar coordinates (angle, distance).



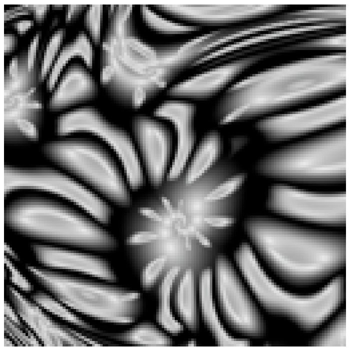
Celtic Texture

Creates a random 'celtic knot' pattern.



Center Warp

Warps the image radially, randomly inward/outward toward/away from the center.

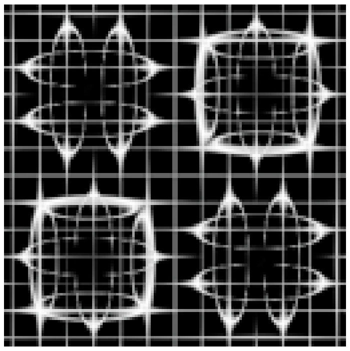


Charm Fractal

A charm fractal is based on a strange attractor formula, but rather than plotting the orbit of the attractor, we record the minimum magnitude of the orbit.   What makes this particular node different is that besides the slider parameters, the coefficients of the quadratic used are presented as editable text.

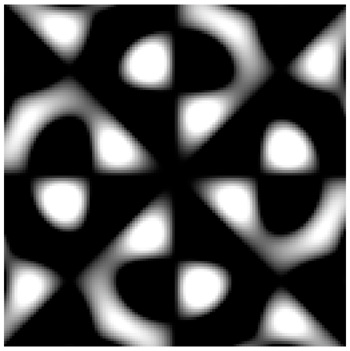
Each coefficient is represented by the letter 'A' through 'Y', and there are 12 of them all told, so you can put in any random series of letters (in either uppercase or lowercase). If you enter something that isn't between 'A' and 'Y' it will ignore it. If you don't enter enough letters, it will add an 'L' and then repeat the letters entered (so if you don't enter anything, you get a series of 'L's). As a result, any sort of mashing of the keyboard can be converted to a valid coefficient specifier (or you can choose from the pop-up list of various valid and "interesting" coefficients).

Note that there are two outputs - one is the minimum orbit point (using the specified shape as a measurement criteria), and the other contains the index of when that point was hit. Connecting one to the input of a gradient and the other to and adjuster of that gradient's output can produce rich and subtlety shaded images.



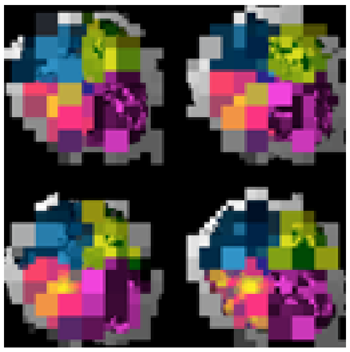
Checker Warp

Warps the coordinates to produce something that often resembles a checkerboard.



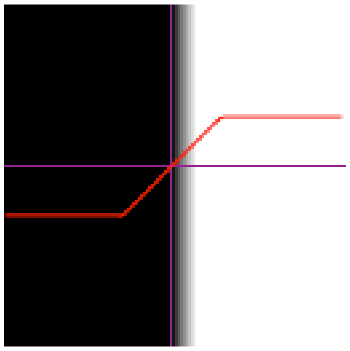
Chladni Pattern

Creates patterns based on drawing a bowstring against a salt covered metal plate, as demonstrated by Ernst Chladni in the 1700s.



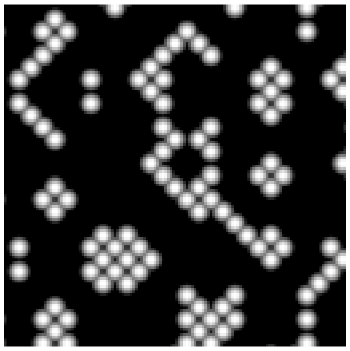
Chunky Pixelizer

Pixelizes the coordinate space into varying sizes pixels.



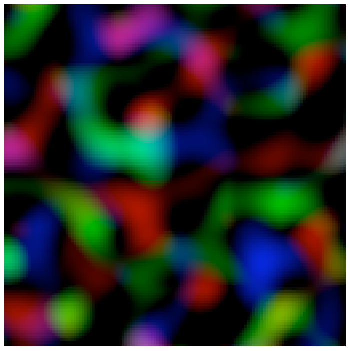
Clipper

Limit the upper and lower value of each channel.



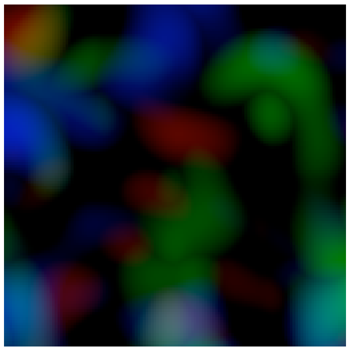
Clustered Shapes

Produces random clusters of the specified shape.



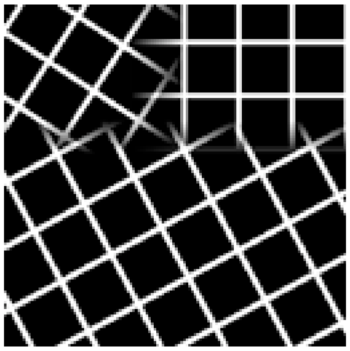
Color Noise

Produces colored noise based on Perlin noise functions.



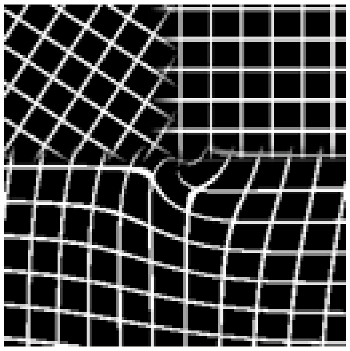
Color Turbulence

Produces colored noise based on turbulence noise functions.



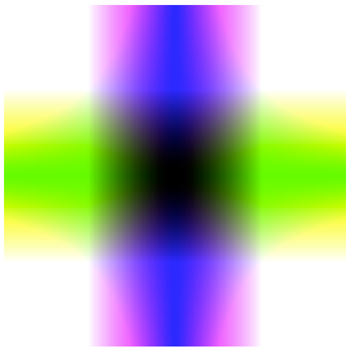
Complex Add

Treats the inputs as complex numbers and adds them.



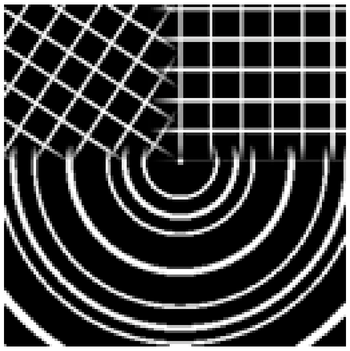
Complex Add Norm

Treats the inputs as complex numbers, adds them, and then normalizes the result.



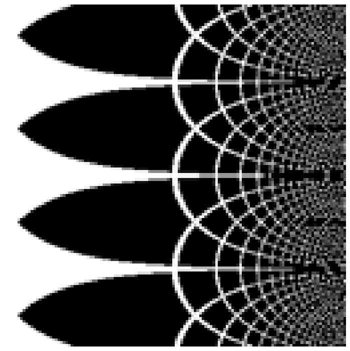
Complex Colorer

Converts a complex number into a color, based on two user specified colors.



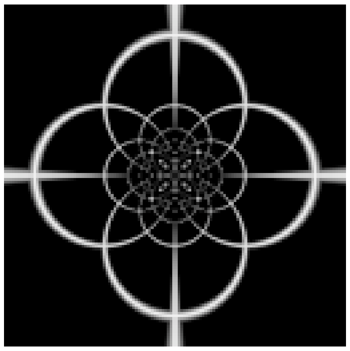
Complex Div

Treats the inputs as complex numbers and divides the left by the right.



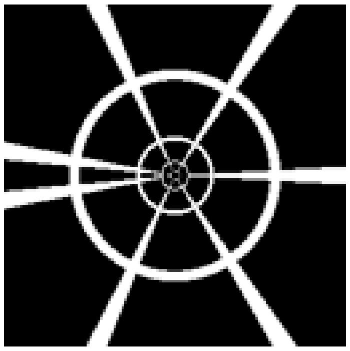
Complex Exp

Treats the input as a complex number and performs a complex exponential on it



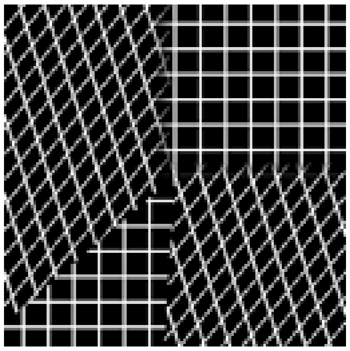
Complex Inverse

Treats the input as a complex number and calculates the inverse of it. This warps space such that infinity is move to the origin, and the origin is stretched to infinity.



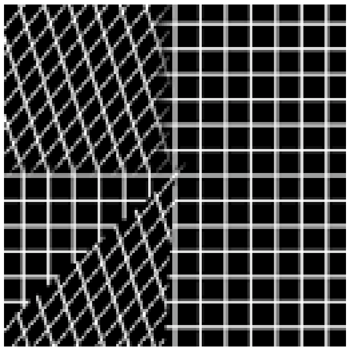
Complex Log

Treats the input as a complex number and performs a complex logarithm on it



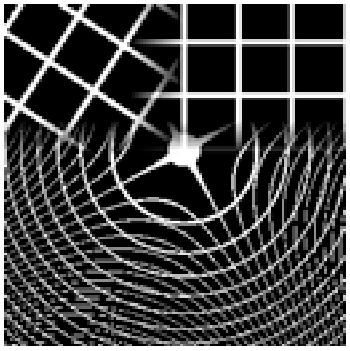
Complex Max

Treats the input as complex numbers and find the larger of the two.



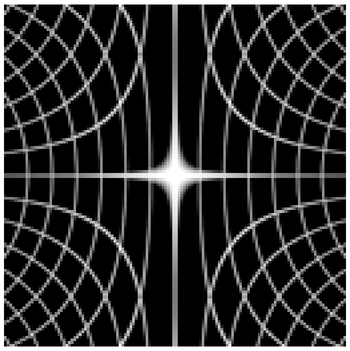
Complex Min

Treats the input as complex numbers and find the smaller of the two.



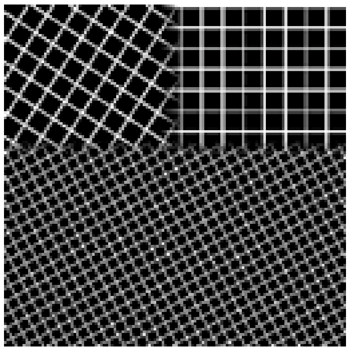
Complex Mult

Treats the inputs as complex numbers and multiplies them,



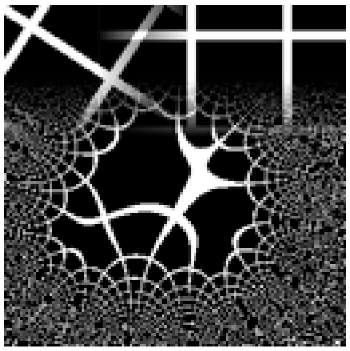
Complex Polar Map

Transforms the coordinate space by treating it as a complex number and applying a polar map to it.



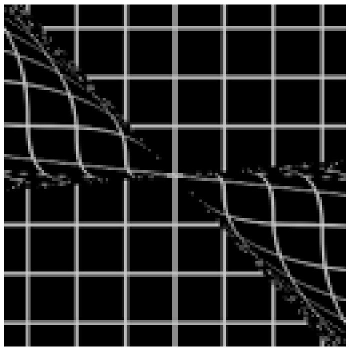
Complex Sub

Treats the input as complex numbers and subtracts one from the other.



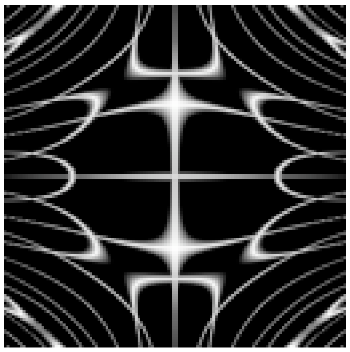
Complex Warptal

Applies a complex number function to the input coordinates to warp it into a new, distorted space.



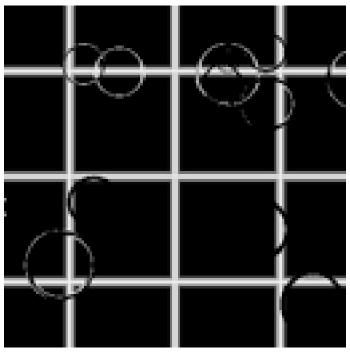
Cone Lens

Warps the coordinate space as if viewed through a pair of cones joined at the tips.



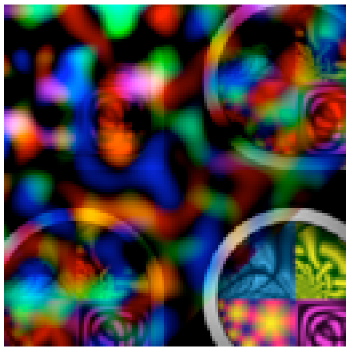
Corniside

Warps the coordinate space, pushing things into the quadrant corners near the origin, leaving the area near the axis relatively untouched.



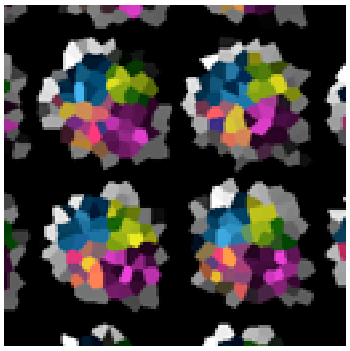
Craters

Creates randomly scattered distortions that resemble a crater-marked surface.



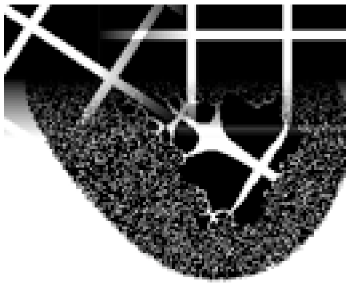
Cross Blend

Blends the two input colors into a color that is a uniform blend of the two (controlled by the slider).



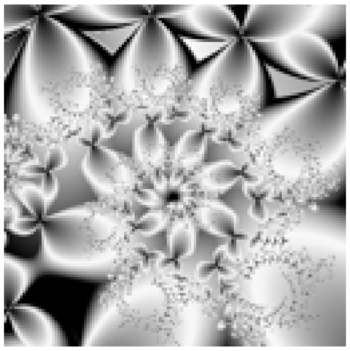
Crystalizer

Resamples the coordinate space to break it into varying sized crystal facets instead of squares.



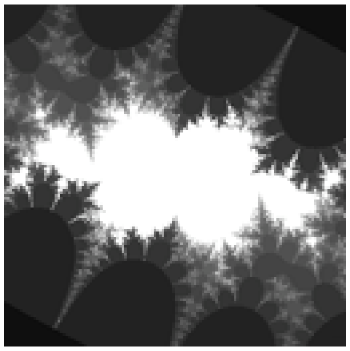
Cubic Warptal

Warps space by treating the coordinates as a complex number and cubing the result multiple times.



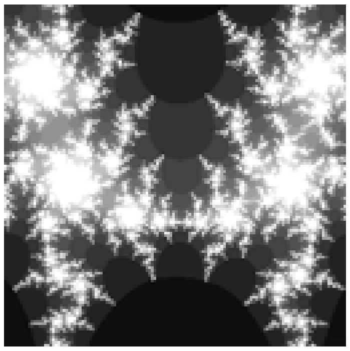
Custom Charm Fractal

Similar to a charm fractal (finding a minimal orbit point using a customizably shaped distance function) but allows a custom formula to be specified.



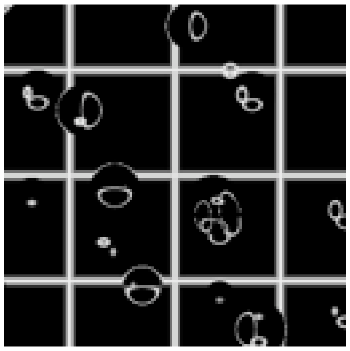
Custom Fractal

Creates a Mandelbroit style escape iterative fractal with a user specified expression.



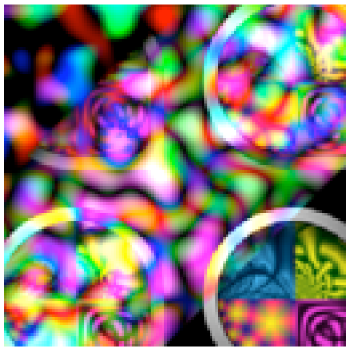
Custom Julia

Creates a Julia style escape iterative fractal with a user specified expression.



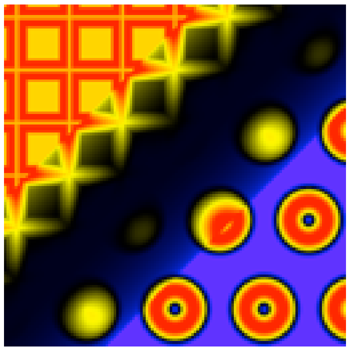
Dew Drops

Creates randomly scattered distortions that resemble drops of water on the surface.



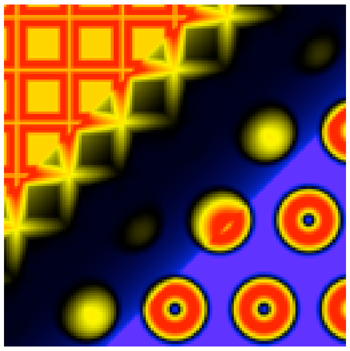
Difference Blend

Blends two colors together by taking the absolute value of the difference between the two.



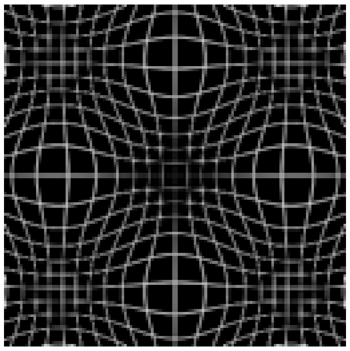
Distance

Calculates the distance from the origin that the input coordinate is.



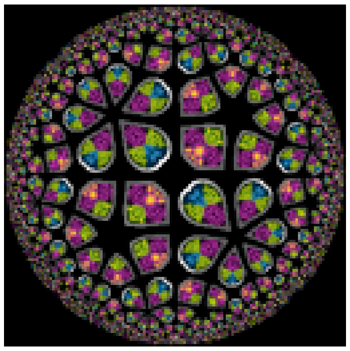
Distance^2

Calculates the square of the distance from the origin that the input coordinate is.



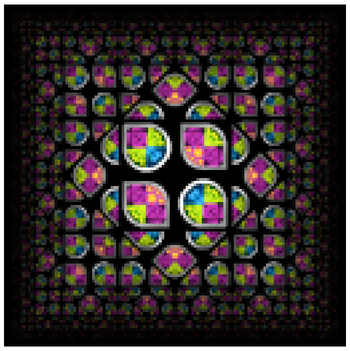
EggCrate

Warps space to resemble an egg crate.



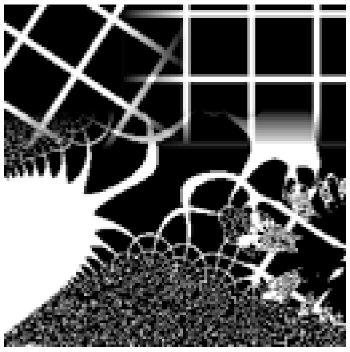
Escher Disk

Tiles the coordinates to produce a hyperbolic tessellation similar to the 'Circle Limit' works of M.C. Escher.



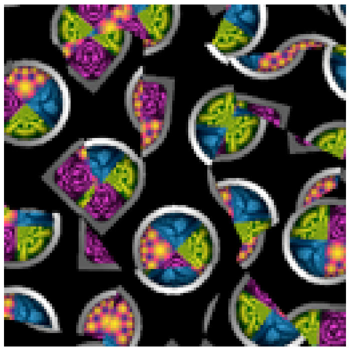
Escher Square

Tiles the coordinates to produce a tessellation similar to the 'Square Limit' work of M.C. Escher.



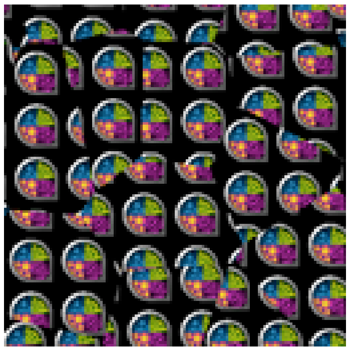
Exp Warptal

Warps space by treating the input coordinates as a complex number and using the exponent function on it.



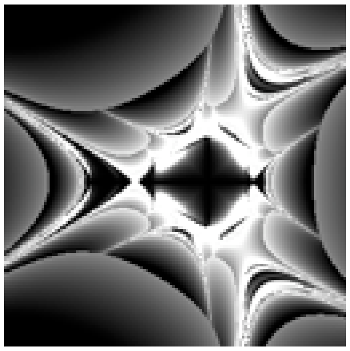
Facet Spinner

Breaks the coordinate space into multiple irregular tiled facets which are rotated around the origin.



Facet Tiler

Breaks the coordinate space into multiple irregular tiled facets.

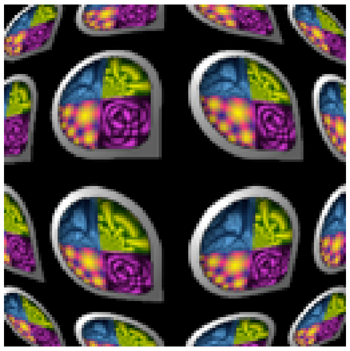


Fearful Icon Fractal

Iterative fractal based on Fearful Icon formula

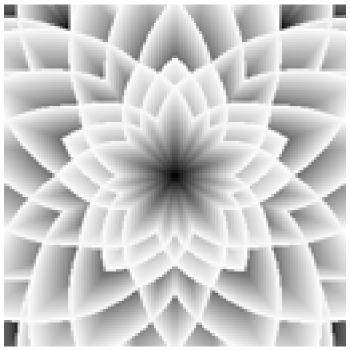
Filter Adjuster

Adjust up to four indepedent channels simultaneously.



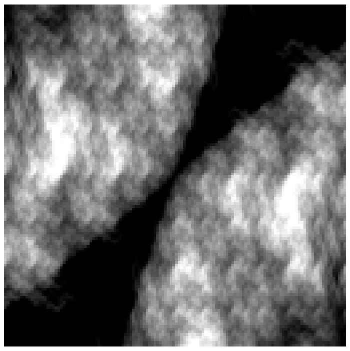
Fish Eye

Warps the image as if viewed through a fish-eye lens.



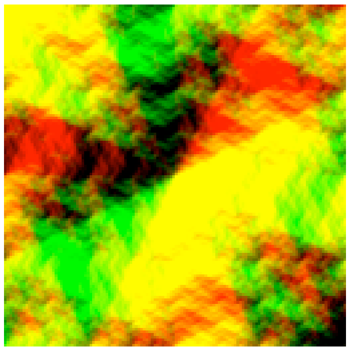
Flower Petals

Creates a texture that looks like various combinations of overlapping flower petals.



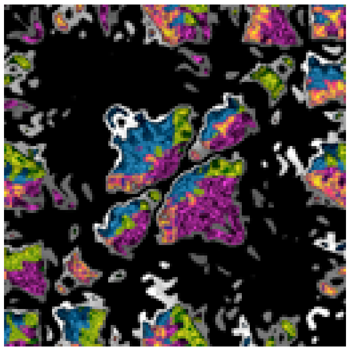
Fourier Texture

Using pseudo-Fourier transformations, creates a texture based on the source image frequencies (details vs flat spaces). Will work if only one input is connected (or connect same source to both inputs).



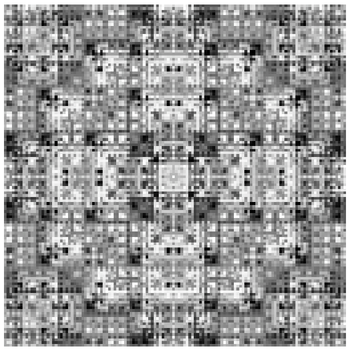
Fourier Vecture

Using pseudo-Fourier transformations, creates a vecture based on the source image frequencies (details vs flat spaces). Will work if only one input is connected (or connect same source to both inputs).



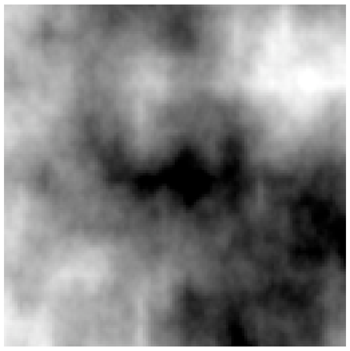
Fourier Warp

Using pseudo-Fourier transformations, warps either details or gentle curves of the source space (or just use it like any old warper).



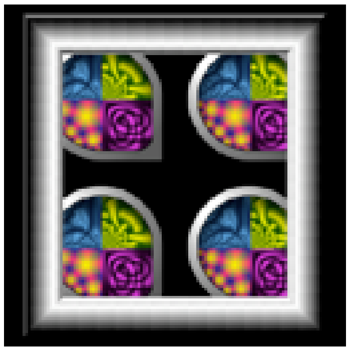
Fractal Carpet

Creates a fractally generated carpet, where the corners determine the middle of the square, which is subdivided and the process repeated.



Fractal Landscape

Creates a fractal landscape via the offset midpoint technique.

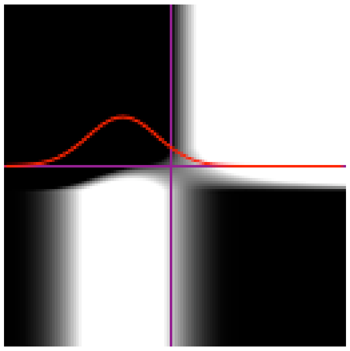


Frame Mat

Creates a frame for an image, producing the coordinates for the frame material, its alpha (allowing the inside to show through), and a Y-Adjustment value to create a lighting effect

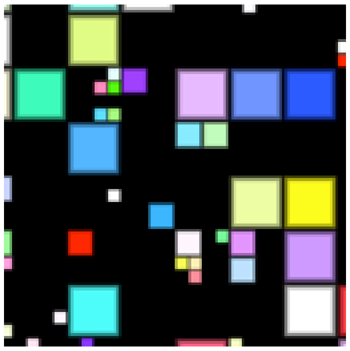
Frieze Tiler

Tiles the coordinate space in one direction in any of the 7 classical 'frieze' patterns.



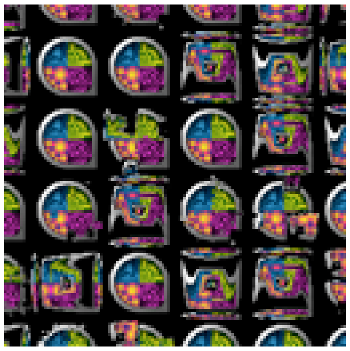
Gausian

Apply a Gaussian transform to each channel, resulting bell curve shape.



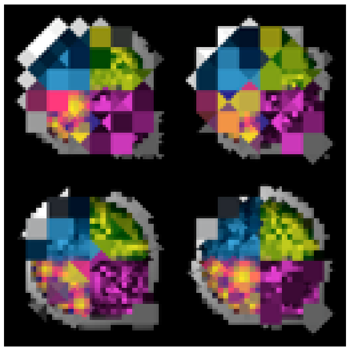
Giza Colorizer

Creates a colored texture composed of randomly sized pyramid structures.



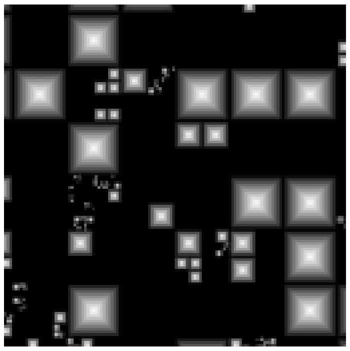
Giza Displacer

Warps space to wrap it around randomly sized scattered pyramid structures.



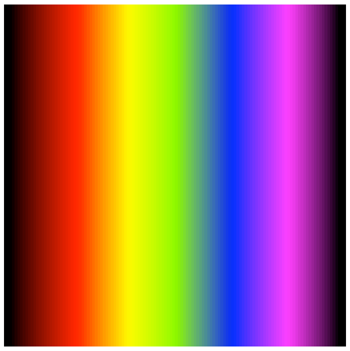
Giza Pixelizer

Resamples the coordinate space, converting it to a series of randomly sized squares & triangles instead of square pixels.



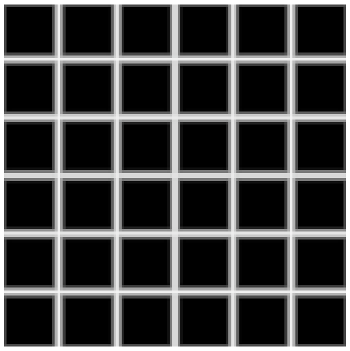
Giza Texture

Creates a texture composed of randomly sized pyramid structures.



Gradient

Converts the input coordinate into a color based on an specified gradient.



Grid Lines

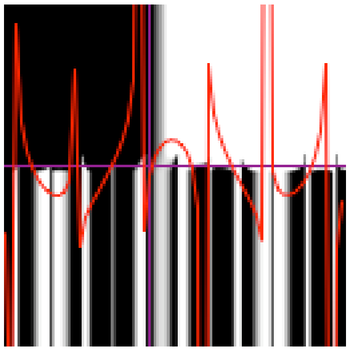
Generates a periodic grid of lines.

HSB to RGB

Convert HSL color space to RGB.

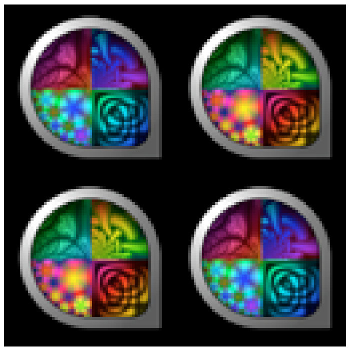
HSV to RGB

Convert HSV color space to RGB.



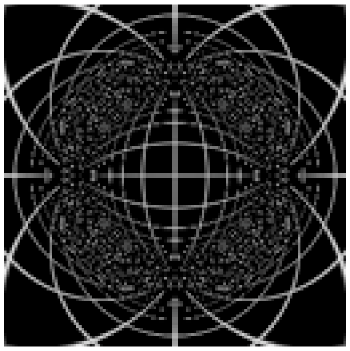
Harmonic Sine

Creates a harmonic version of a sine wave.



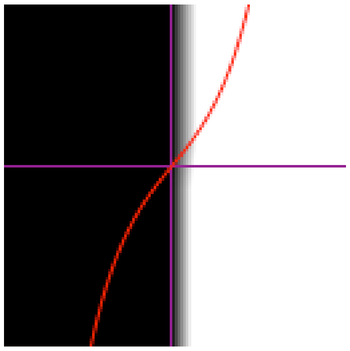
Hue Adjuster

Uses the last input channel to control the hue of the incoming color.



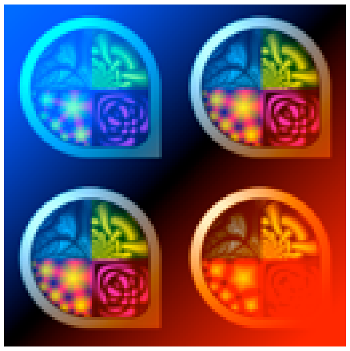
HyperTransform

Warps the coodinate space through a hyperbolic transformation, which maps the entire space into a disk, with infinity at the edge of the disk, and then flips it again ouside that disk such that the origin is at infinity.



Hyperbolic Sine

Calculate the hyperbolic sine of each channel, scaling both the frequency and amplitude, and adjusting the phase of the result.

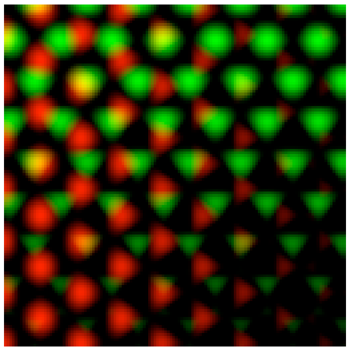


I Adjuster

Adjusts the input color using last parameter to change the 'I' chroma.

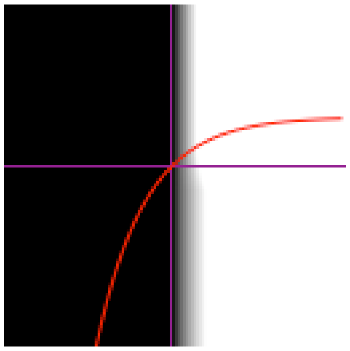
Image

Converts the input coordinate to the pixel at that location in the specified image.



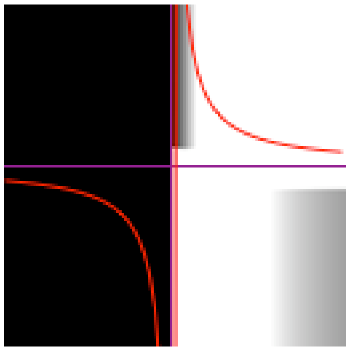
Interference Sine

Warp space based on the interference between two different sine waves.



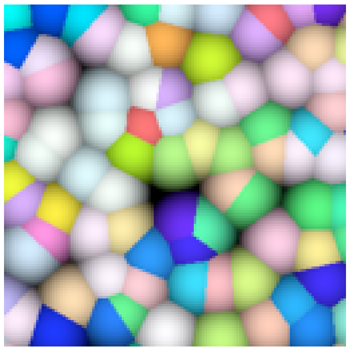
Inverted Gausian

Apply an inverted Gaussian transform to each channel, which smoothly limits values that approach infinity.



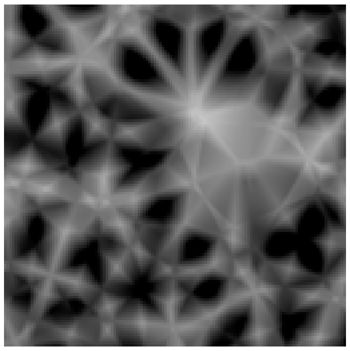
Inverter

Calculate the reciprocal of each channel.



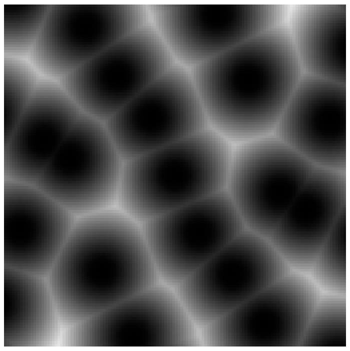
Irregular Colored Shapes

Creates an irregularly spaced set of colored shapes, creating a wide variety of effects, from cell membranes to barnacles.



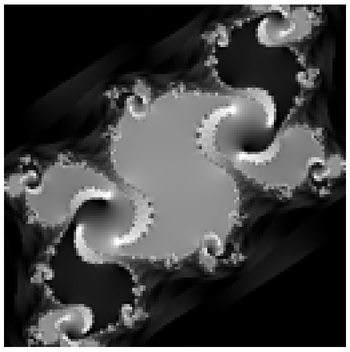
Irregular MetaShapes

Super imposes a network of irregular shapes, allow for a much more sophisticated texture.



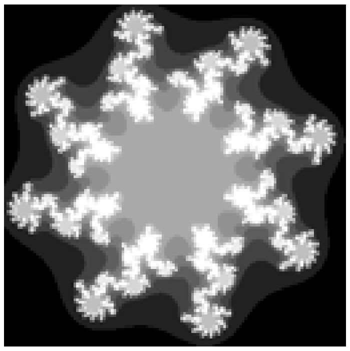
Irregular Shapes

Creates an irregularly spaced set of shapes, creating a wide variety of effects, from cell membranes to barnacles.



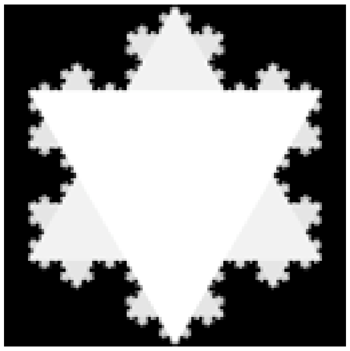
Julia Fractal

Calculates an iterative Julia fractal based on a quadratic.



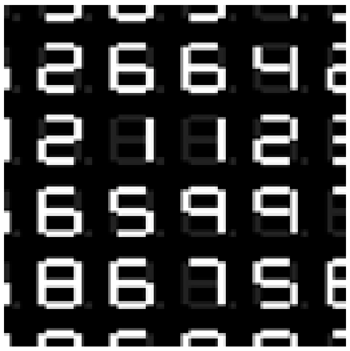
Julia^{x} Fractal

Calculates an iterative Julia fractal based on a power function.



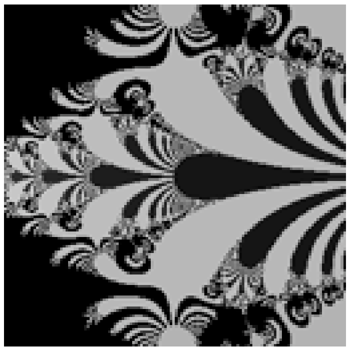
Koch Snowflake

Fractally tiles space into the shape of a Koch Snowflake.



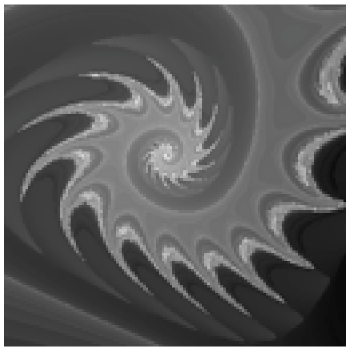
LED Segments

Create a texture composed of random LED displays.



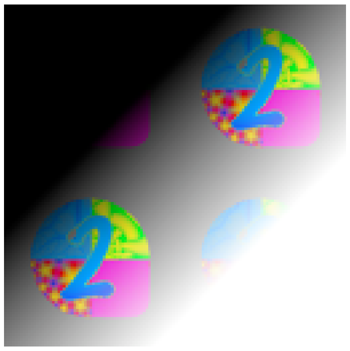
Lambda Exp Fractal

Calculates an iterative Julia fractal based on the exp function.



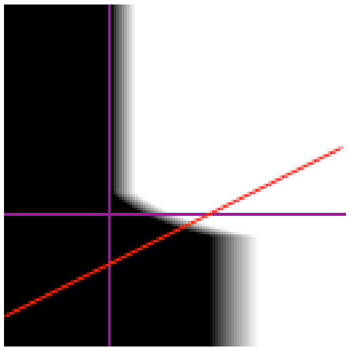
Lambda Fractal

Calculates an iterative fractal based on the lambda function.



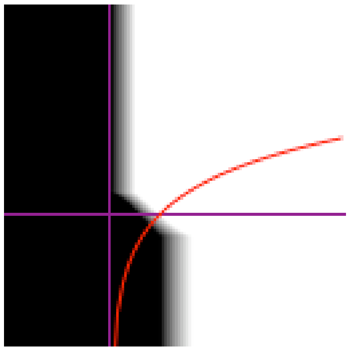
Lightness Adjuster

Adjust the 'Lightness' color component.



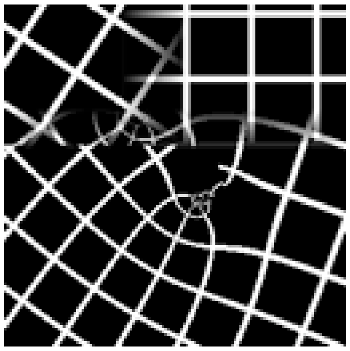
Linear

Apply a linear transformation to each channel independently. Note that a single scaling and offset value are used for all four channels.



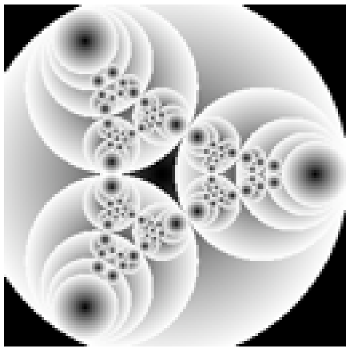
Log

Apply the log function to each channel.



Log Warptal

Warps space by treating the coordinates as a complex number and take the log the result.



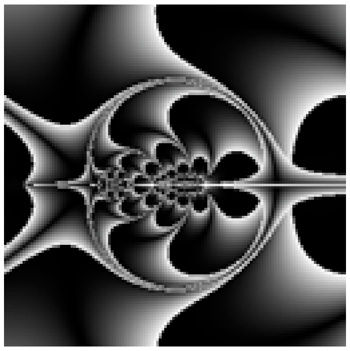
Lucas Circle

Take a circle, with three equally spaced points along the circumference of it. Now imaging three circles nested inside the first circle such that they touch each other and the other circle at those three points. These three circles circles are called *Lucas Circles* (and, combined with the outer circle and another inner circle that would lay between them and just touch the three circles are called *Soddy Circles*).

If you then take each of those circles, and the point there it touches the outer circle and the other two circles, and use those three points to generate the next generation of circles, and repeat, you create this delicate lacework of circles.

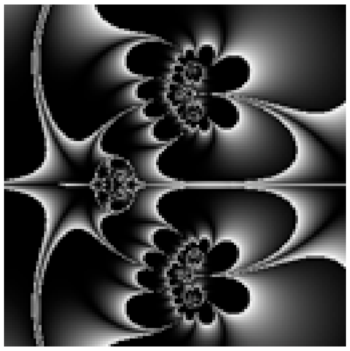
Lyapunov

Creates a texture based on the Lyapunov function.



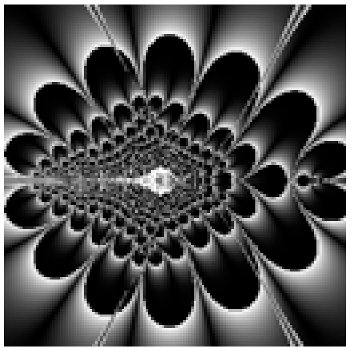
MagnetJ^2 Fractal

A Julia style fractal based on second order magnet field equations (great for traps).



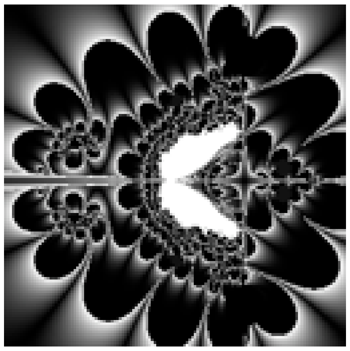
MagnetJ^3 Fractal

A Julia style fractal based on third order magnet field equations (great for traps).



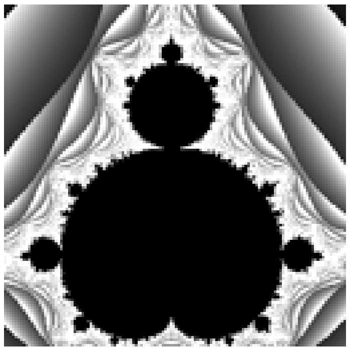
MagnetM^2 Fractal

A Mandelbort style fractal based on second order magnet field equations (great for traps).



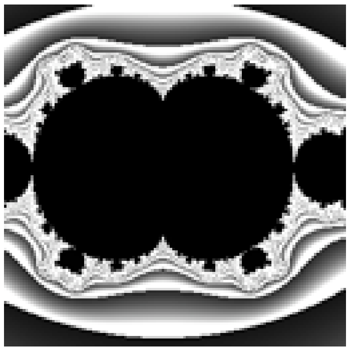
MagnetM^3 Fractal

A Mandelbort style fractal based on third order magnet field equations (great for traps).



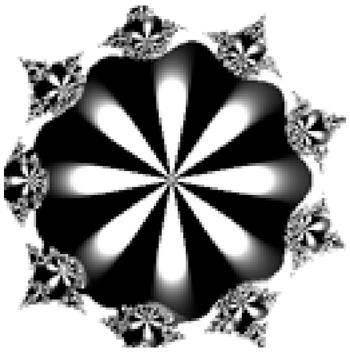
Mandelbrot^2 Fractal

Calculates an iterative fractal based on a quadratic.



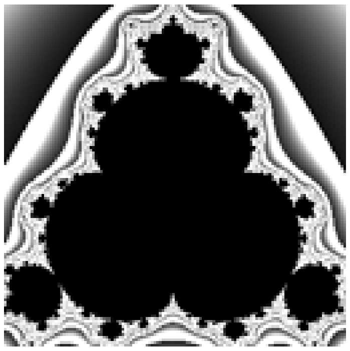
Mandelbrot^3 Fractal

Calculates an iterative fractal based on a cubic.



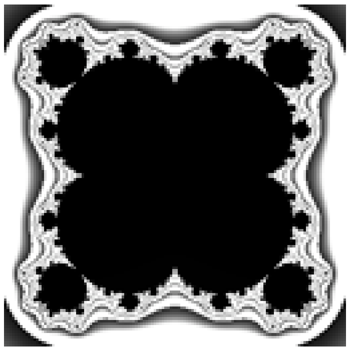
Mandelbrot^x Fractal

iterative fractal based on an arbitrary power equation



Mandelbrot^4 Fractal

Calculates an iterative fractal based on a fourth power equation.

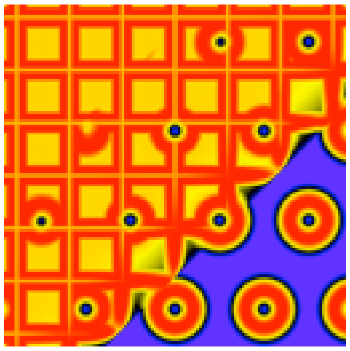


Mandelbrot^5 Fractal

Calculates an iterative fractal based on a fifth power equation.

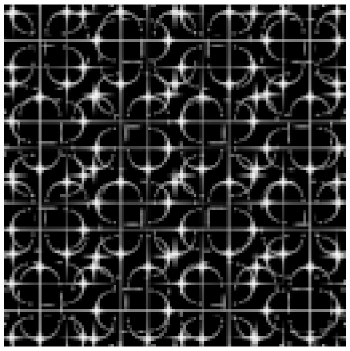
Max Blend

Blends two colors by taking the maximum of each of the respective channels.



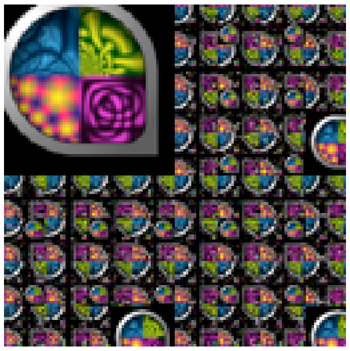
Max Reducer

Combines all the input channels and takes the maximum of them.



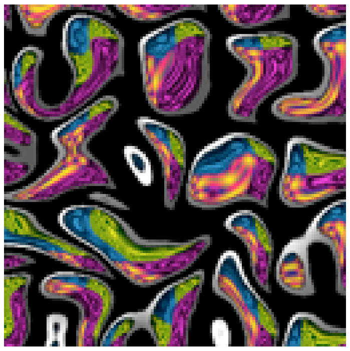
Mayan Warp

Warps space to produce something that often resembles Mayan style heiroglyphics.



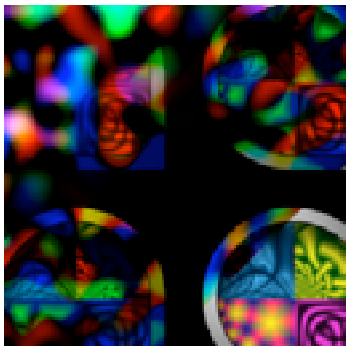
Menger Tiling

Similar to Sierpinksi Tiler, breaks space into 3x3 grids, repeating this process for 8 of the 9 grids.



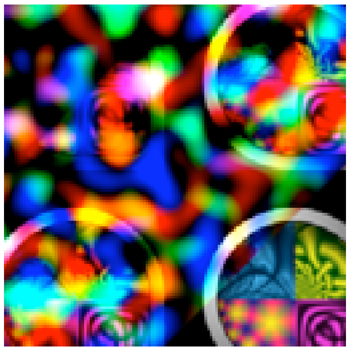
Mercury Warp

Warps space by distorting it with random noise;



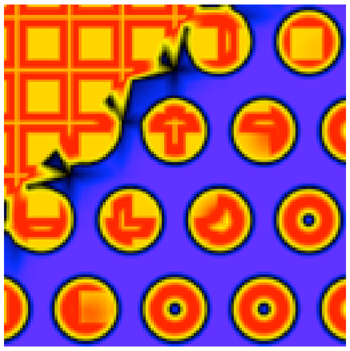
Min Blend

Blends two colors by taking the minimum of each of the respective channels.



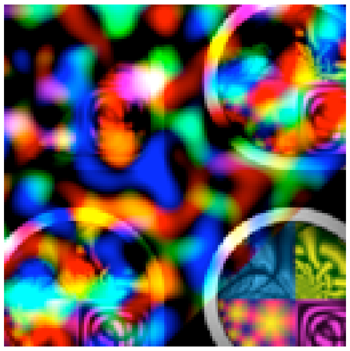
Min Max Blend

Blends two colors by taking the blend of maximum and minimum of each of the respective channels.



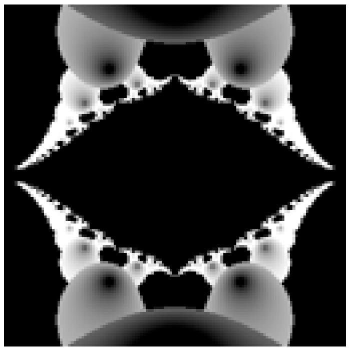
Min Reducer

Combines all the input channels and takes the minimum of them.



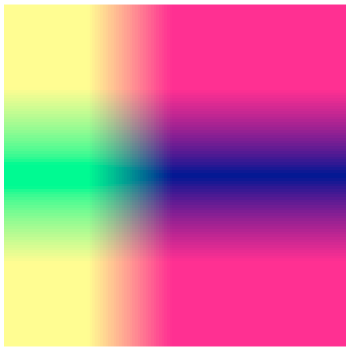
Min+Max Blend

Blends two colors by taking the sum of maximum and minimum of each of the respective channels.



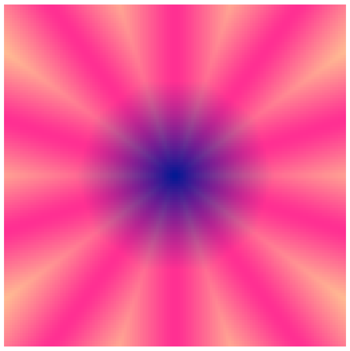
Mira Fractal

Calculates an iterative fractal based on the Mira function.



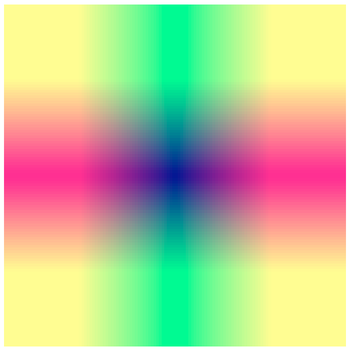
Mirror

Splits space as if a mirror was placed in it, reflecting it in half.



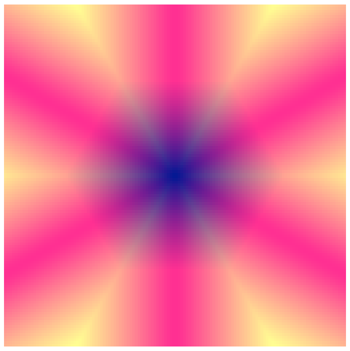
Mirror 10

Splits space as if angled mirrors were placed in it, reflecting it in tenths.



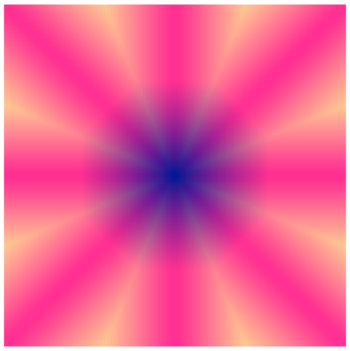
Mirror 4

Splits space as if two mirrors were placed in it at right angles, reflecting it in quarters.



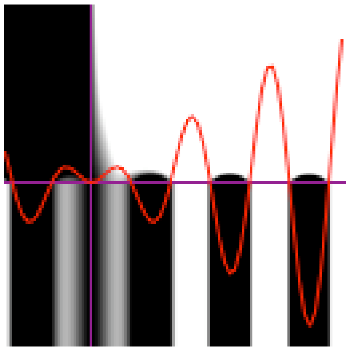
Mirror 6

Splits space as if angled mirrors were placed in it, reflecting it in sixths.



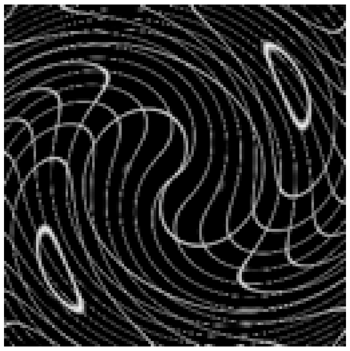
Mirror 8

Splits space as if angled mirrors were placed in it, reflecting it in eights.



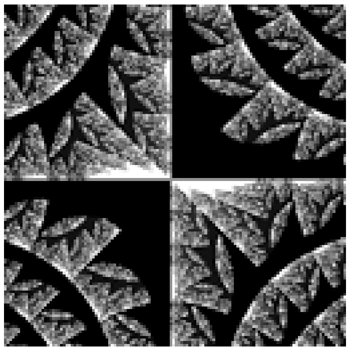
Modulated Sine

Calculate the sine of each channel multiplied by the original value, scaling both the frequency and amplitude, and adjusting the phase of the result.



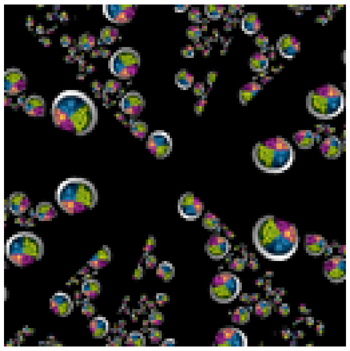
Modulated Spiral

Warps the coordinates into a spiral of ripples.



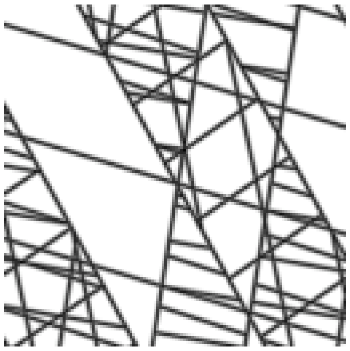
Moire Fractal

Calculates an iterative fractal based on a Moire function.



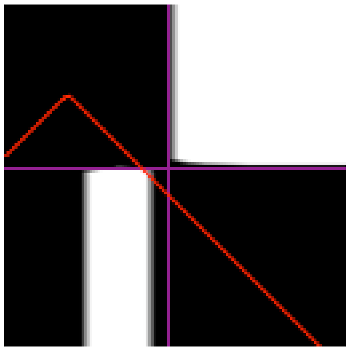
Moire Warptal

Warps space using a Moire function.



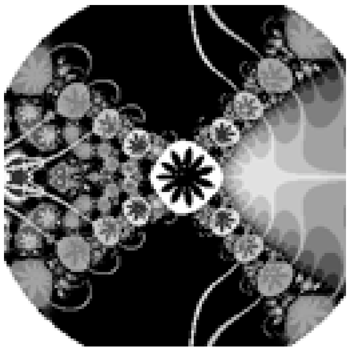
Nazca Lines

Generates random straight lines that criss-cross (or stop) no unlike the lines on the Nazca plain in Peru.



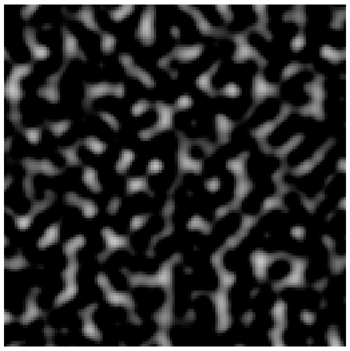
Negative Absolute

Subtract the absolute value of each channel from a constant.



Newton Fractal

Calculates an iterative fractal based on the Newton method of root finding.

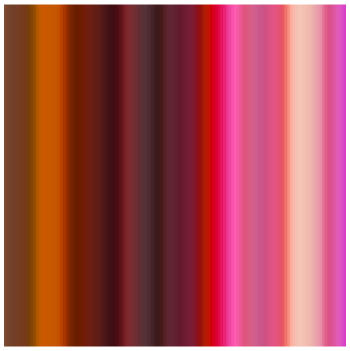


Noise

Random noise generated using a white noise function.

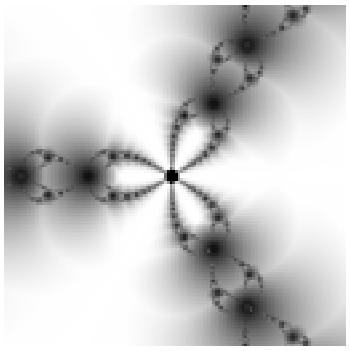
Noise Warp

Distorts space using random noise.



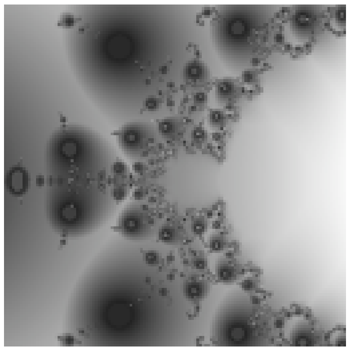
Noisy HLS Gradient

Creates a gradient based on a single color using noise to generate different hues or saturations.



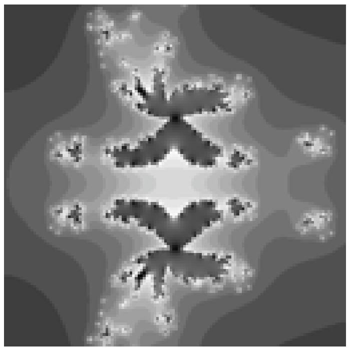
NovaJ Fractal

A Julia-set style fractal based on the Nova function.



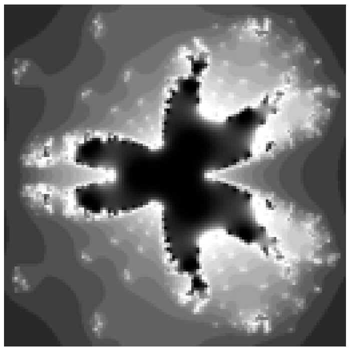
NovaM Fractal

A Mandelbrot-set style fractal based on the Nova function.



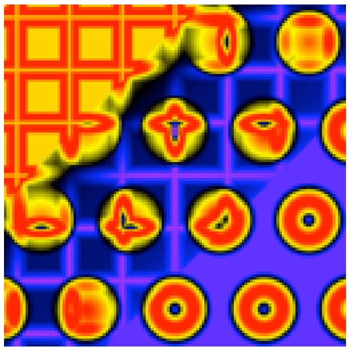
NuPhoenix Fractal

Calculates an iterative Mandelbrot-set fractal based on an enhanced Phoenix function.



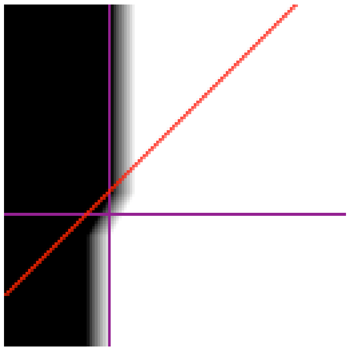
NuPhoenixJ Fractal

Calculates an iterative Julia-set fractal based on an enhanced Phoenix function.



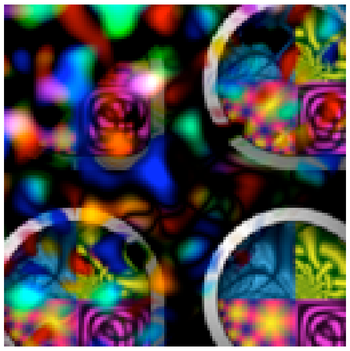
Offset Product

Reduces the input parameters by multiplying them all together.



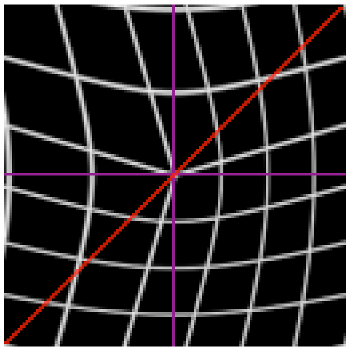
Offseter

Offset each channel independently by four independent offsets.



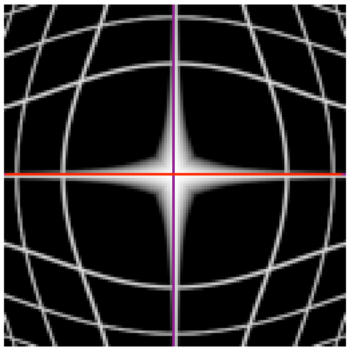
Overlay Blend

Overlays one image on top of another, using both a transparency and the alpha change of the top image.



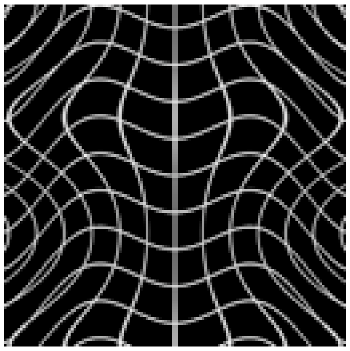
P-Displace

Displaces all four input channels by a fifth, parametric, channel.



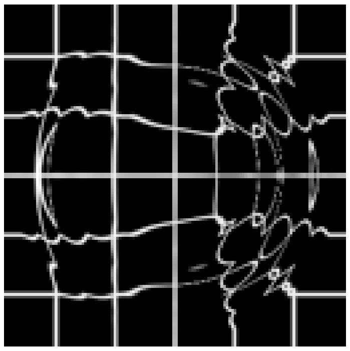
P-Scaler

Scales all four input channels by a fifth, parametric, channel.



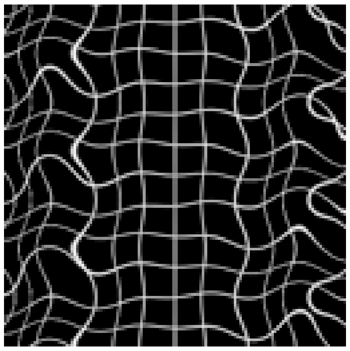
P-Sine Distort

Distorts space using a sine function modulated by a parametric channel.



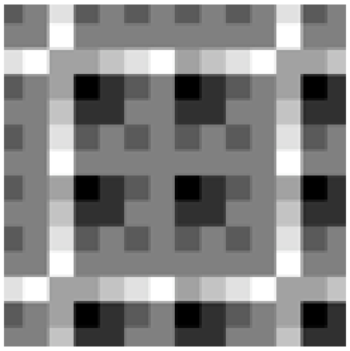
P-Splash

Distorts space like splash, modulated by a parametric channel.



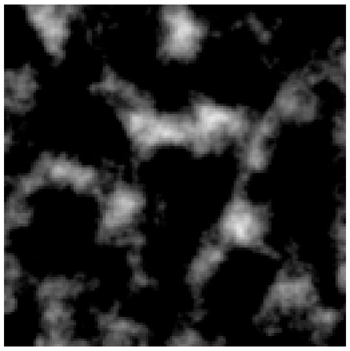
P-Wave

Distorts space like a wave, modulated by a parametric channel.



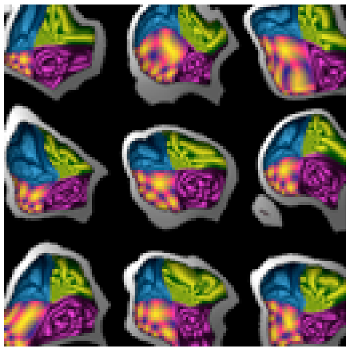
Pattern Texture

Generates repeating grayscale square patterns.



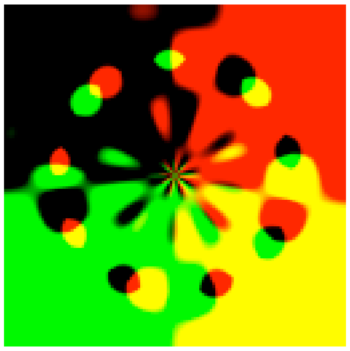
Perlin Noise

Random noise generated using a Perlin noise function.



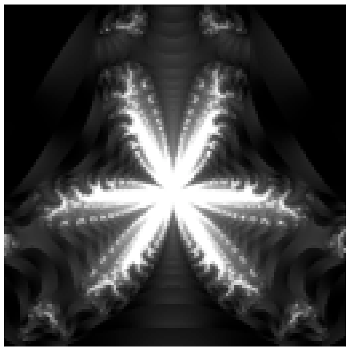
Perlin Warp

Distorts space using a Perlin noise function.



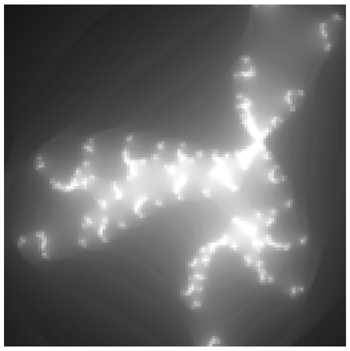
Petalizer Sine

Warps space to squeeze and pinch it into various flower petal like shapes.



Phoenix Fractal

Calculates an iterative Mandelbrot-set fractal based on the Phoenix function.

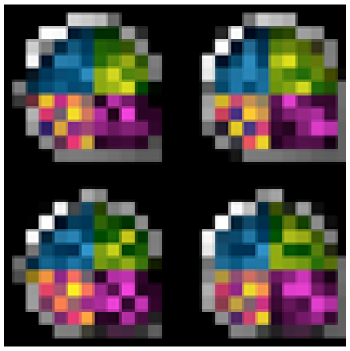


PhoenixJ Fractal

Calculates an iterative Julia-set fractal based on the Phoenix function.

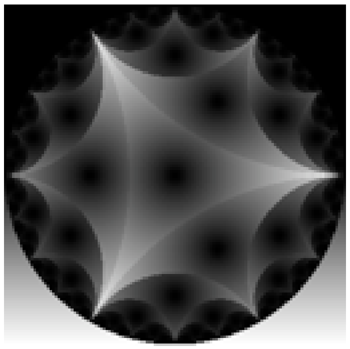
Pinwheel

Slices and spins space around like a pinwheel, repeating a pie-shaped slice to form a complete circle.



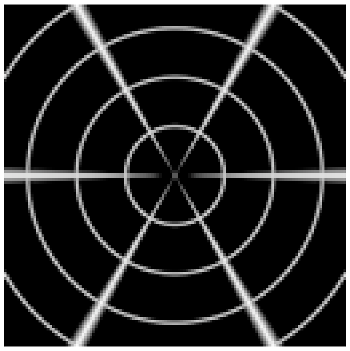
Pixelizer

Resamples space to pixelize a square grid with all the same values within the grid.



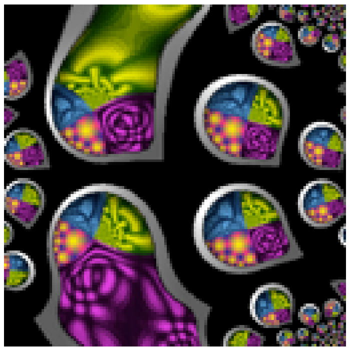
Poincare Disk

Generates a Poincare disk with hyperbolic parallel lines repeatedly bisecting it.



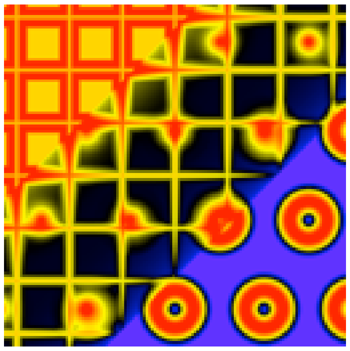
Polar→Cartesian

Converts from Polar coordinates (angle, distance) to Cartesian (X,Y).



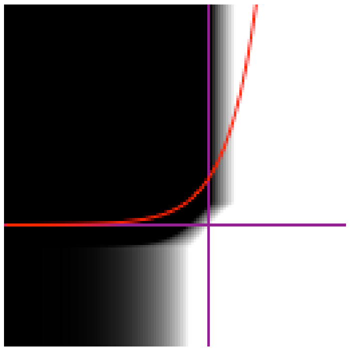
Poly Warp

Warps the image using a complex polynomial, whose coefficients are the parameters, yeilding a vaguely fractalish distortion.



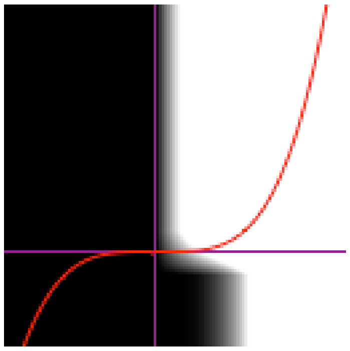
Polynomial

Reduces the four inputs using a fourth order polynomial (the first channel is raised to the fourth, the second channel is cubed, etc...)



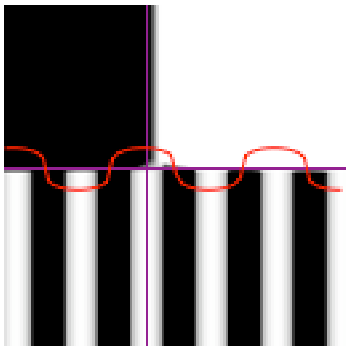
Power

Apply a power function to each channel.



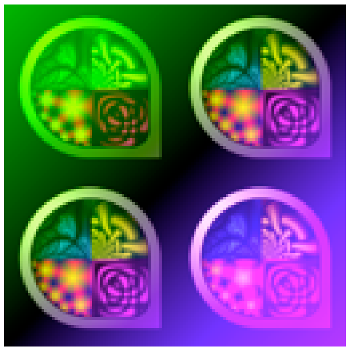
Power Curve

Apply a power function to each channel, maintaining the sign of the original value.



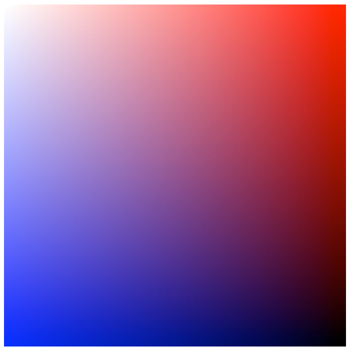
Power Sine

Similar to a sine function, but then passes the output through a power function.



Q Adjuster

Adjust the 'Q' chroma color component.



Quad Colorer

Blends together four colors, one from each quadrant, to produce the final color. This is simplified 2 dimensional gradient.

RGB to CMYK

Convert RGB color space to CMYK.

RGB to HSB

Convert RGB color space to HSL.

RGB to HSV

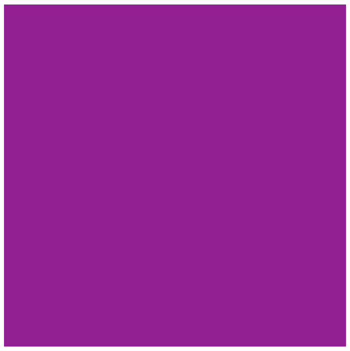
Convert RGB color space to HSV.

RGB to XYZ

Convert RGB color space to XYZ.

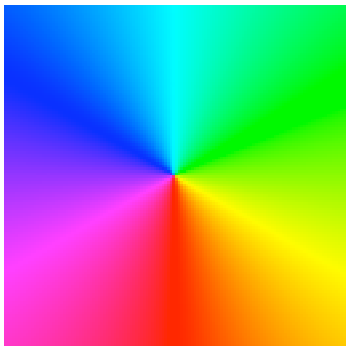
RGB to YIQ

Convert RGB color space to YIQ.



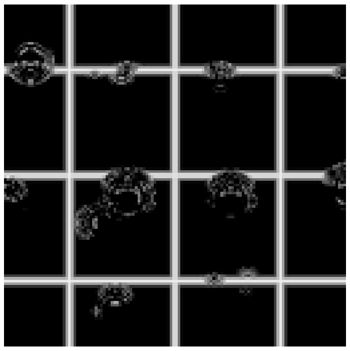
RGBA Color

Creates a single solid fixed color - also useful for providing up to four constant values.



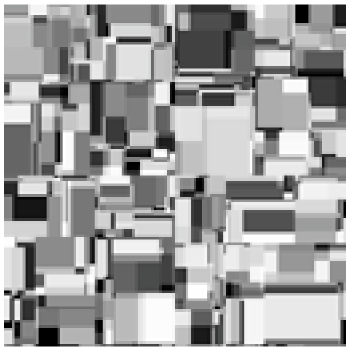
Radial Hues

Generates a color wheel around the origin.



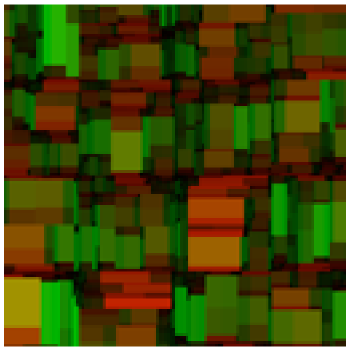
Rain Drops

Creates randomly scattered distortions that resemble a pond in the rain.



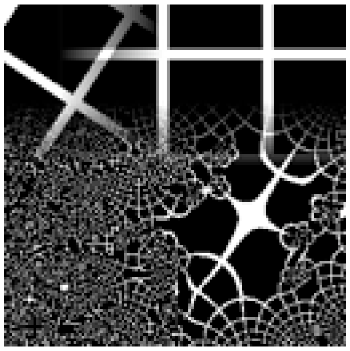
Random Rectangles

Generates random shaded rectangles.



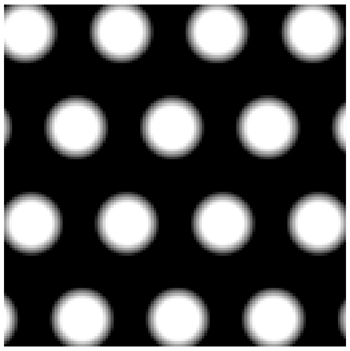
Random Vectangles

Generates a random vector texture of rectangles, with the output being the width and height of the rectangles.



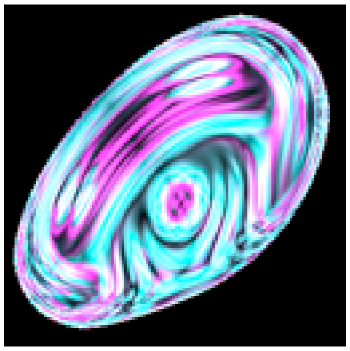
Recip Warptal

Warps space by treating the coordinates as a complex number and taking the reciprocal and offsetting multiple times.



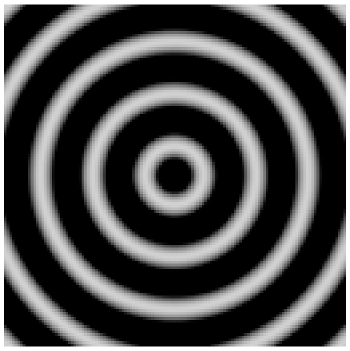
Regular Shapes

Creates a texture composed of evenly spaced dots (or squares, diamonds, etc...).



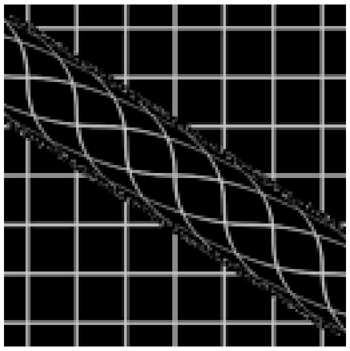
Ripple Attractor

Creates a strange attractor style fractal based on transformed polynomials.



Rippled Texture

Distorts space by introducing periodic circular ripples.

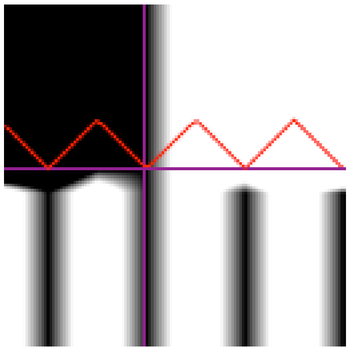


Rod Lens

Warps the coordinate space as if viewed through a cylindrical rod.

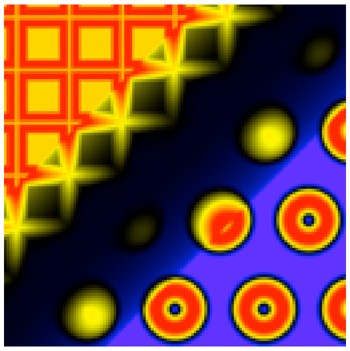
Saturation Adjuster

Adjust the saturation of the input color.



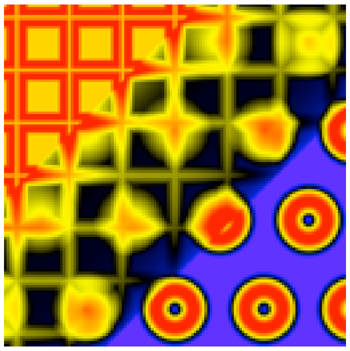
Saw

Converts each channel to a triangle wave, with the given frequency, amplitude, and phase.



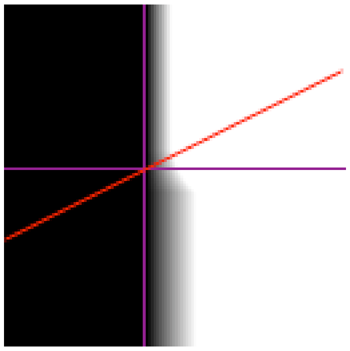
Scaled Product

Combines all the input channels by multiplying them all together to form the output.



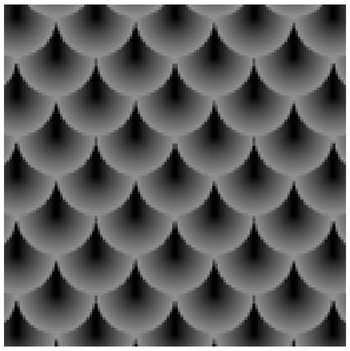
Scaled Sum

Combines all the input channels by multiplying each by a different constant and then summing the results.



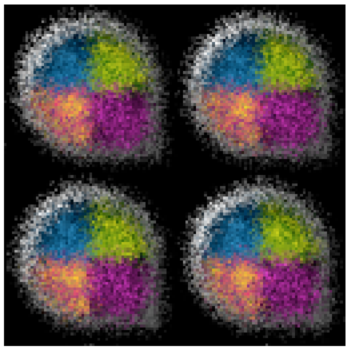
Scaler

Multiply each channel independently by four independent scales.



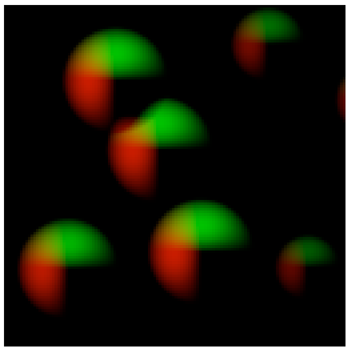
Scales

Generates a texture similar to overlapping snake (or other creature) scales.



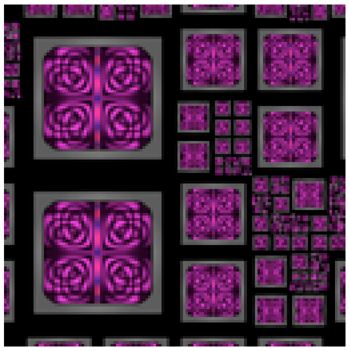
Scatter Displacer

Disperses space by a random amount, making the result appear 'fuzzy'.



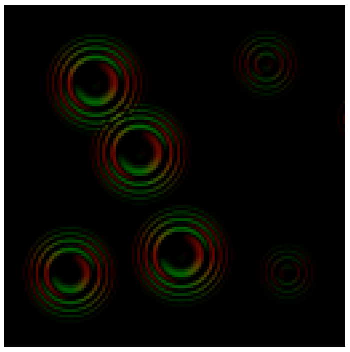
Scattered Bumps

Creates a vector texture composed of 2D-bumps scattered randomly about.



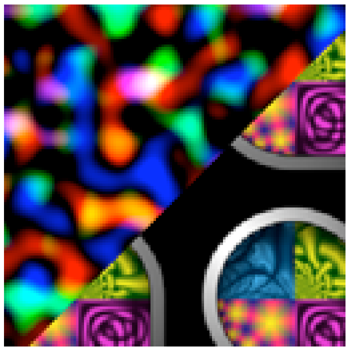
Scattered Sierpinski Texture

Remaps space similar to a Sierpinski Tiling, but with randomness determining the output instead of simple rules.



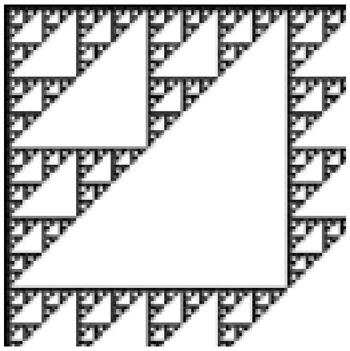
Scattered Splash

Creates a vector texture composed of 2D-splashes scattered randomly about.



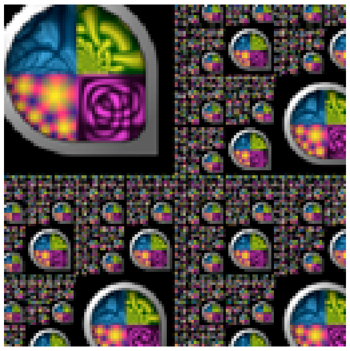
Selector Blend

Uses a parametric input channel to determine which of the two sets of inputs are used for the output.



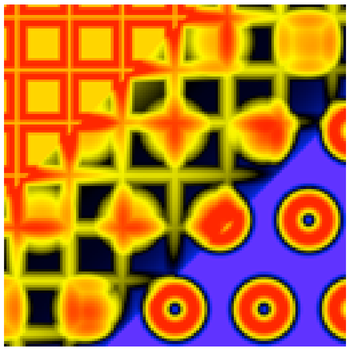
Sierpinski

Creates a recursive structure known as Sierpinski's Triangle.



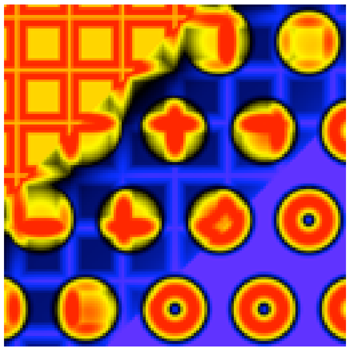
Sierpinski Tiling

Tiles space to make it appear similar to Sierpinski's Triangle, but with the underlying coordinates instead of solid colors.



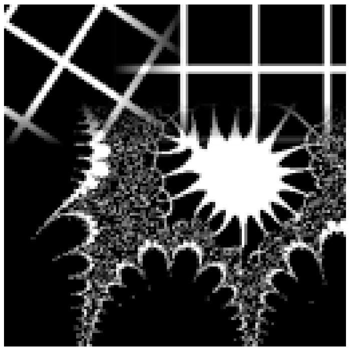
Sin Add Reducer

Combines the input channels by applying a sine function to each and then totaling them.



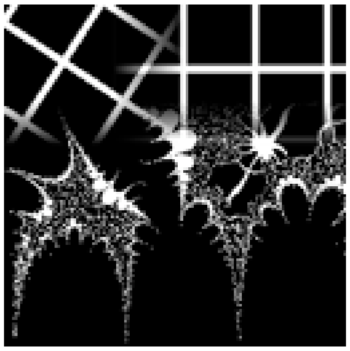
Sin Mult Reducer

Combines the input channels by applying a sine function to each and then multiplying them.



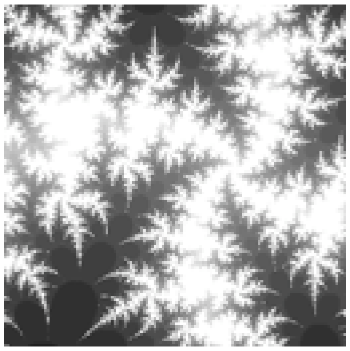
Sin Warptal

Warps space by treating the coordinates as a complex number and taking the sine of it multiple times.



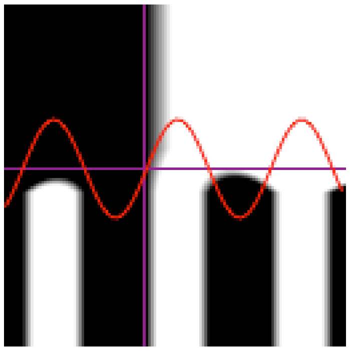
SinCos Warptal

Warps space by treating the coordinates as a complex number and taking the produce of sine and cosine of it multiple times.



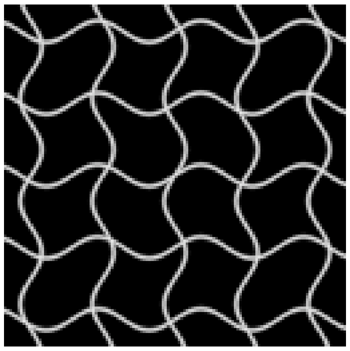
SinH Fractal

Iterative escape fractal based on the hyperbolic sine function.



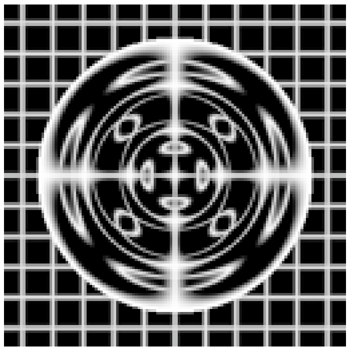
Sine

Calculate the sine of each channel, scaling both the frequency and amplitude, and adjusting the phase of the result.



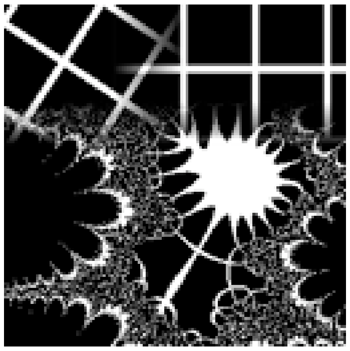
Sine Distort

Distorts space by offsetting it with a sine wave.



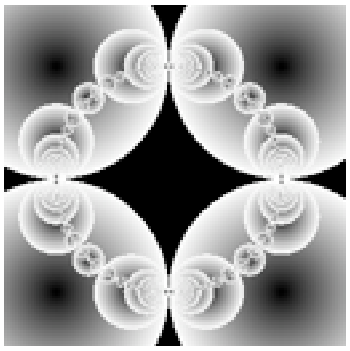
Sine Lens

Warps the coordinate space as if viewed through a circular lens with a strange sine wave cross section.



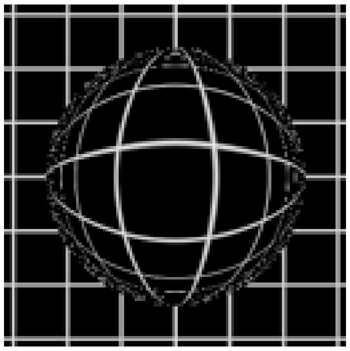
Sinh Warptal

Warps space by treating the coordinates as a complex number and taking the hyperbolic sine of it multiple times.



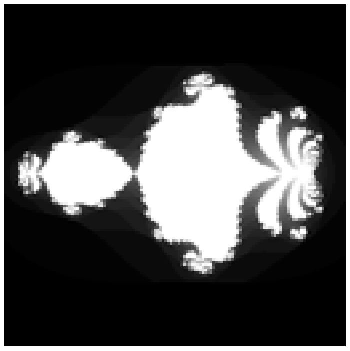
Sphere Chain

Creates an image that simulates a chain of reflecting silvered spheres.



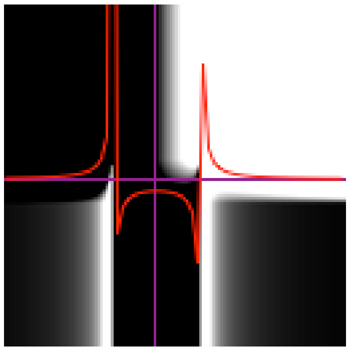
Sphere Lens

Warps the coordinate space as if viewed through a spherical lens with a strange sine wave cross section.



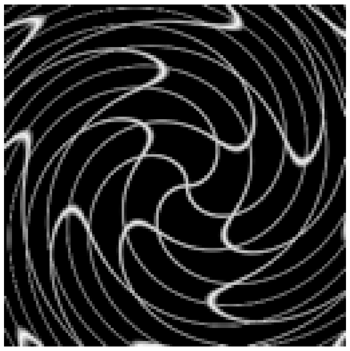
Spider Fractal

Calculates an iterative fractal based on a Spider function.



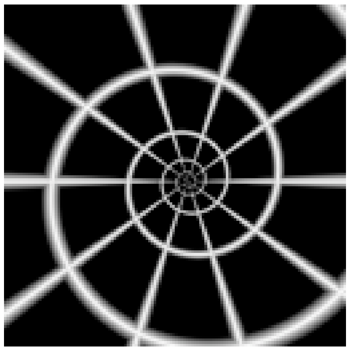
Spike Curve

Produce a sharp spike from each channel.



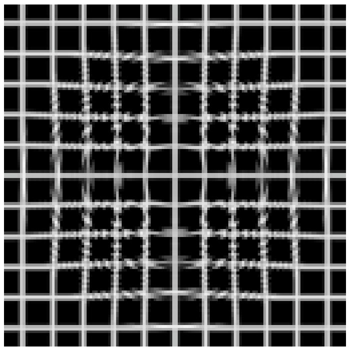
Spiral

Warps space by wrapping it in a spiral around the origin.



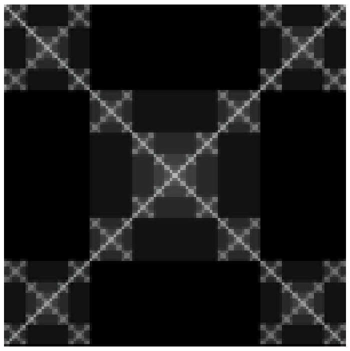
Spiral Tiler

Tiles space into a series of strips that are then spiraled around the center.



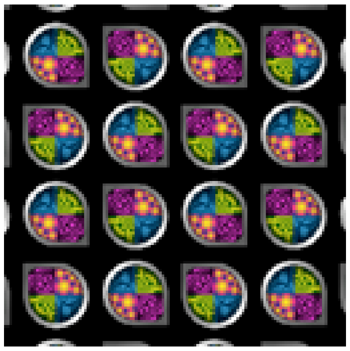
Splash

Warps space as if the surface a pond was splashed with a rock.



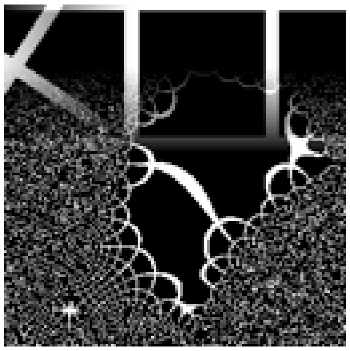
Square Flake

Fractally tiles space in a recusive square subdivision (creating a Peano style curve).



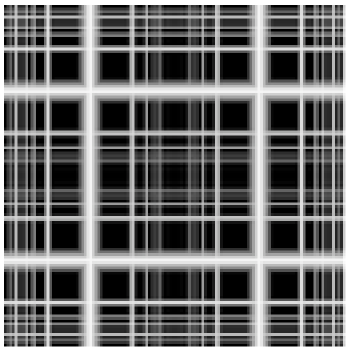
Square Tiler

Tiles space using a series of simple flips, rotations, or mirror images.



Square Warptal

Warps space by treating the coordinates as a complex number and taking the square of it multiple times.

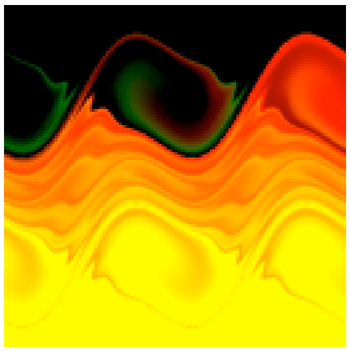


Squarify Warp

Warps space so the entire plane fits in a unit square (mapping infinity to 1.0), which is then repeated.

Stair Step

The moral equivalent of a pixelizer for 1D values, converts a smooth curve into a series of fixed steps.



Standard Map

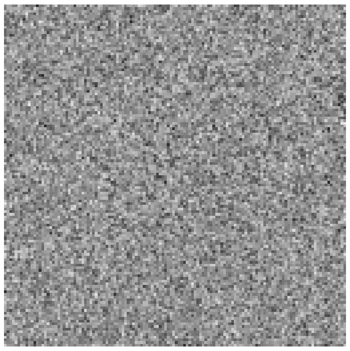
Warps space by applying a Standard Map function to it multiple times.

Starburst Sine

Warps space into a starburst like shape.

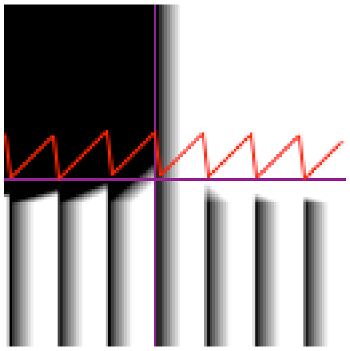
Starfield

Creates a random series of tiny dots scattered on a field of black.



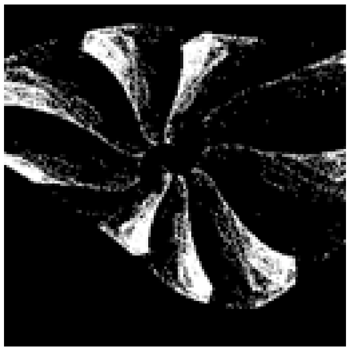
Static

Creates a texture composed of pure random static values.



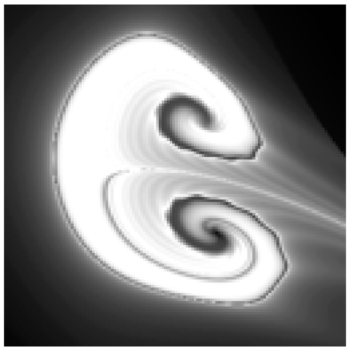
Step Saw

Converts each channel to a sawtooth wave, with the given frequency, amplitude, and phase.



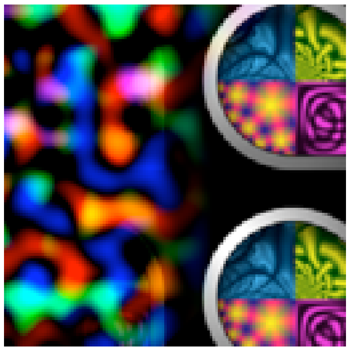
Strange Attractor

Creates a plot of a quadratic strange attractor, with an optional non-linear space.



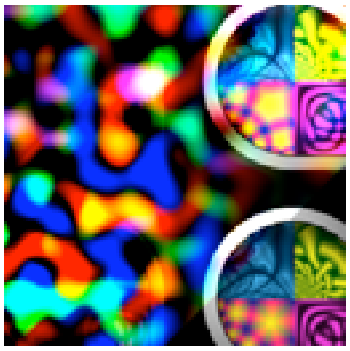
Strange Fractal

Calculates an iterative fractal based on the escape values of a quadratic strange attractor.



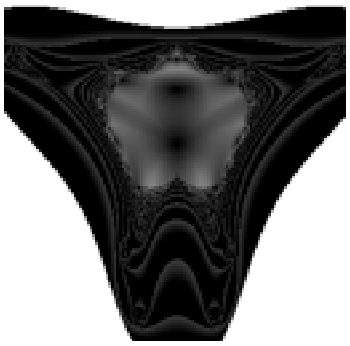
Super Balancer

Blends together two RGBA colors using a third set of channels to control balancing each channel individually.



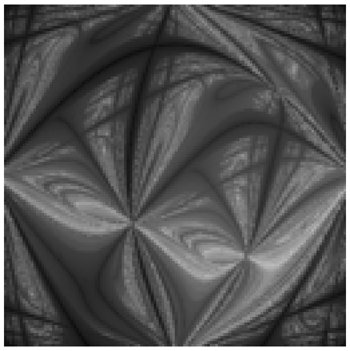
Super Blender

Blends together two RGBA colors using a third set of channels to control blending each channel individually.



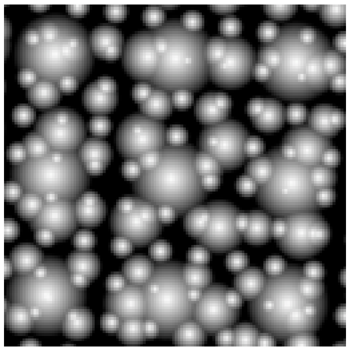
SuperJulia Fractal

Calculates an iterative Julia fractal based on a quadratic function that uses an extra set of inputs to determine the other complex parameter (which can then vary through the image).



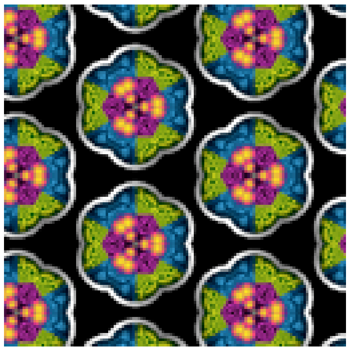
Thorn Fractal

Calculates an iterative Thorn fractal, also known as the Secant Sea.



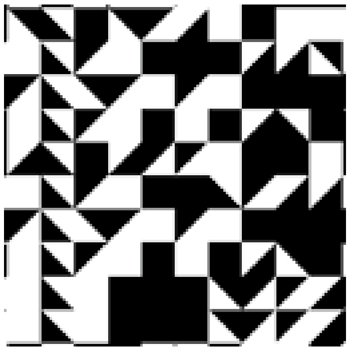
Transformed Grid

Layers of gridded objects, each layer transformed by rotation, scaling, and displacement.



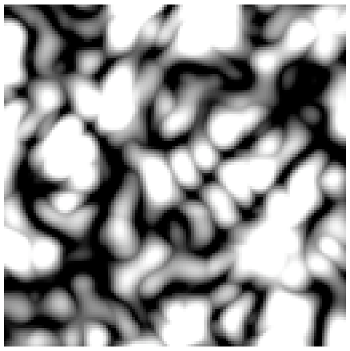
Triangle Tiler

Tiles space using a triangular cell in the classic wallpaper tiling symmetries.



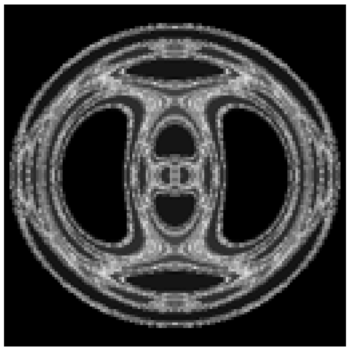
Truchet Texture

Generates a random texture using a Truchet texture - randomly selected 'cells' that connect to form a complex pattern.



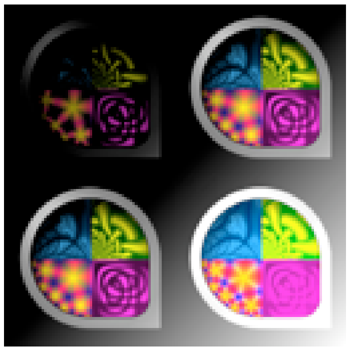
Turbulence

Generates random turbulence based on the input channels.



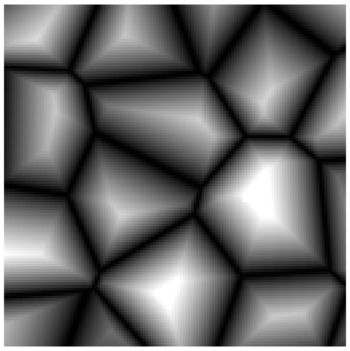
Unity Fractal

Calculates an iterative fractal based on a Unity function.



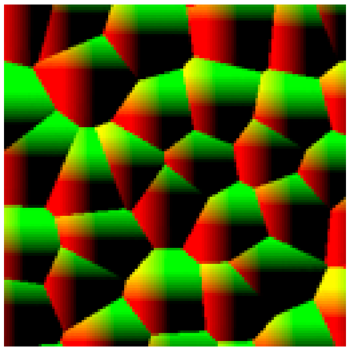
Value Adjuster

Adjust the value of the input color. Similar to, but not exactly the same as, adjusting the intensity or lightness.



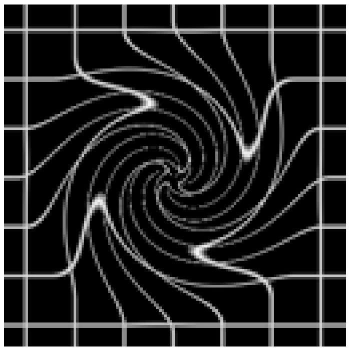
Voronoi Texture

Creates a texture representing a Voronoi Set, which resembles dried crack mud, or certain reptile scales.



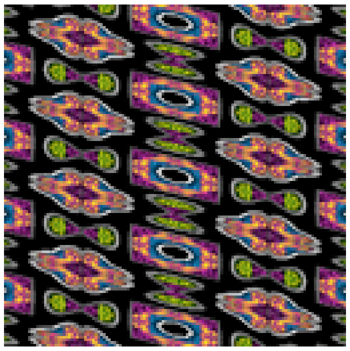
Voronoi Vecture

Creates a vecture representing a Voronoi Set, which resembles dried crack mud, or certain reptile scales.



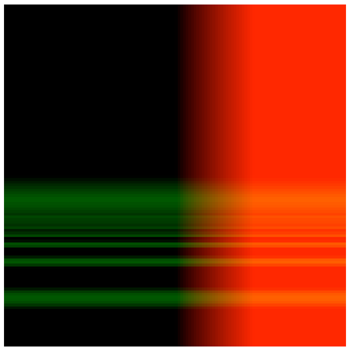
Vortex

Warps space by having a single large spinning vortex in the center of it.



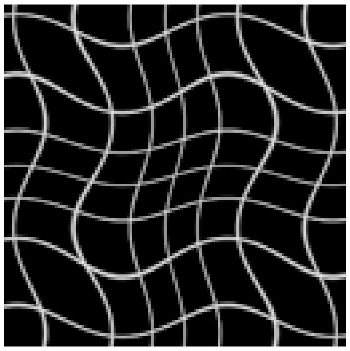
Wallpaper Tiler

Tiles space using a rectangular/rhombic cell in the classic wallpaper tiling symmetries.



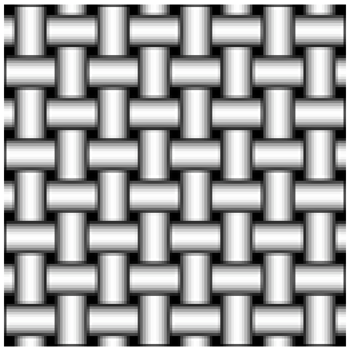
Waterline

Creates a rippled reflection that makes the image appear as if reflected over a lake's surface.



Wave

Distorts space with a series of waves in both the X and Y axis.



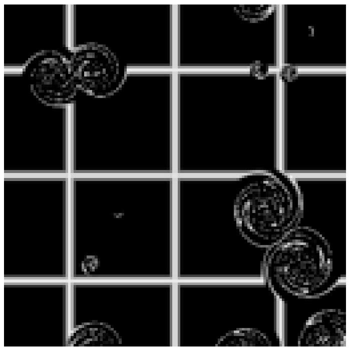
Weaver

Weaver models an eight shaft, eight treadle loom. If that doesn't mean anything to you, just play around with the various presets. The left output provides the shape of the woven material - if you hook only this output up to a gradient, it will provide a nice texture. The right output is used to determine the "color" of the thread, and can be used either as an input for a gradient, or in a blender to determine which of multiple images are used (which can produce "woven images"). If this is done, you'll want to run the left output to a Value Adjuster to shade the result (to give it the texture of woven thread).

If "an eight shaft, eight treadle loom" does mean something to you, you can click the "Edit" button to bring up window that allows editing the warp and weft, as well as what colors the threads are, and the tie up used. On the left are the warp and weft descriptions - a series of digits 1-8 to represent which shafts/treadles are used in the warp/weft. The right two fields let you enter the thread color in a similar manner, but it uses the letters A-Z (uppercase) to allow for up to 26 colors. The checkboxes on the right control the tie up. Changes made in this dialog will immediately be shown in the image, so you can see your updates "live".

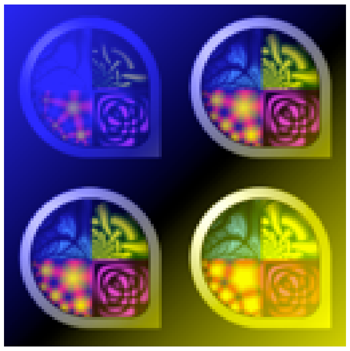
You can also import ".wif" (Weaving Information File) formatted files, though you are limited to 8 shafts & treadles (it can remap thread color if more than 24 are present).

As a short cut, you can use the "-" (minus) character to represent a range of either letters or digits. For example, "1-5-2" is equivalent to "12345432".



Whirls

Creates a series of small vortices scattered throughout the image.

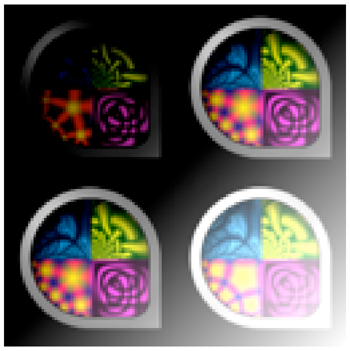


X Color Adjuster

Adjusts the input color as balanced between to user specified colors.

XYZ to RGB

Convert XYZ color space to RGB.



Y Adjuster

Adjust the 'Y' chroma of the input color.

YIQ to RGB

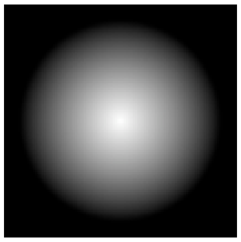
Convert YIQ color space to RGB.

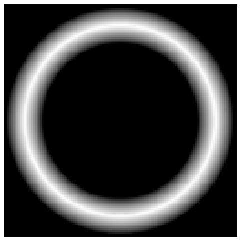
Appendix A - Shapes

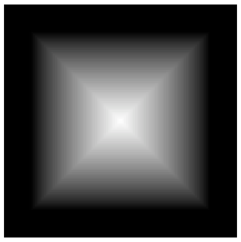
quadrium supports a number of underlying shapes in a variety of nodes. Rather than having a node that represents a series of spheres, and another representing a series of pyramids, etc... it's more convenient to have a single node and be able to specify the underlying shape used. Shapes are used in Regular Textures, Irregular Textures, traps in various escape fractals, and as a distance function in the Charm Fractal. In all instances, this causes a repetition of the underlying shape, often distorted, stretched, twisted, warped, or otherwise made "more interesting".

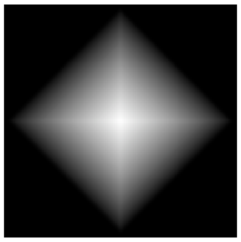
Shapes provide not only the outline of the underlying object, but are actually continuous functions unto themselves. This means that the result of a circular shape function (which is based on the square root of the sum of the squares of the X and Y coordinate - i.e., traditional Euclidian distance) actually produces values that range from 0 at the edge of the shape, to 1 at the center, which produces nice smooth contoured shape. As a result, if viewed as grayscale data, it results in shaded "three dimensional" shapes (a circle actually looks like sphere).

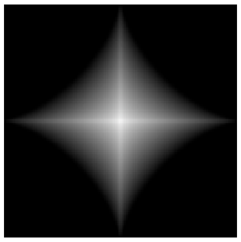
Below is the list of available shapes:

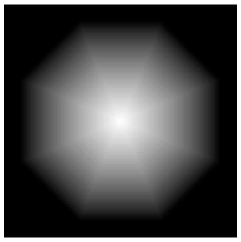

Circle/Sphere

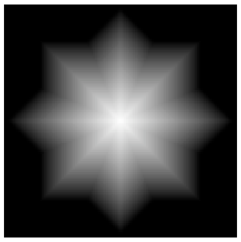

Ring

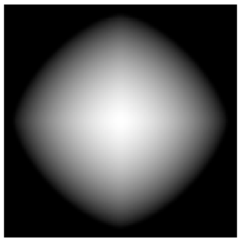

Squares/Pyramid

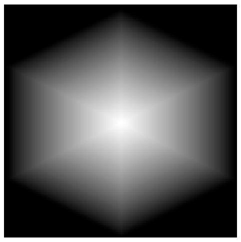

Diamond

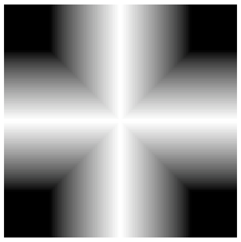

Astroid

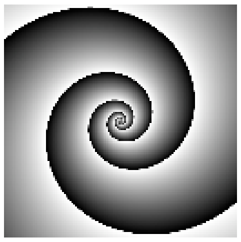

Gem

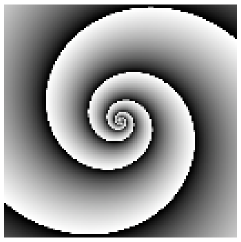

Star

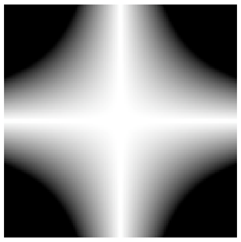

Pillow

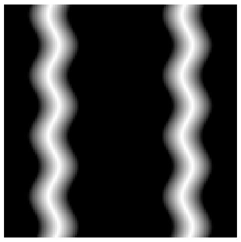

Hexagon

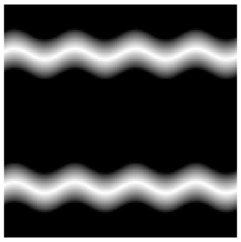

Cross

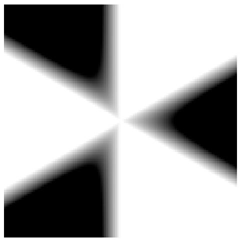

Spiral +

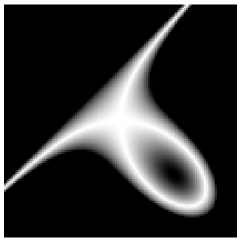

Spiral -

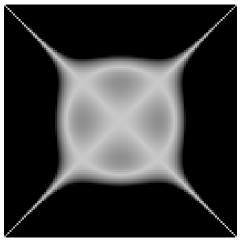

Hyperbola

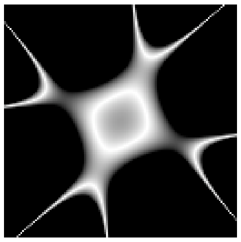

Waves X

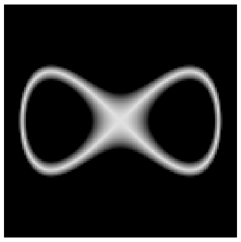

Waves Y

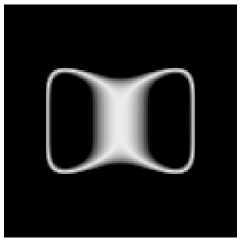

Petals

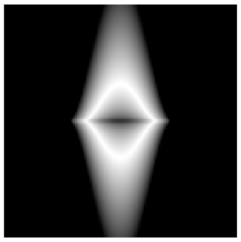

Decarte Folium

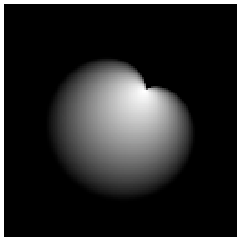

Devil's Curve


Maltese Cross


Eight Curve


Butterfly

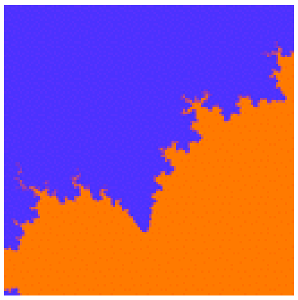

Lips

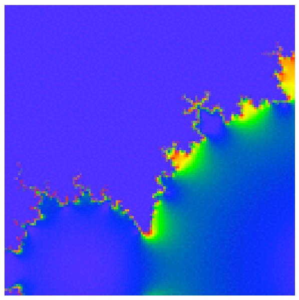

Cardoid

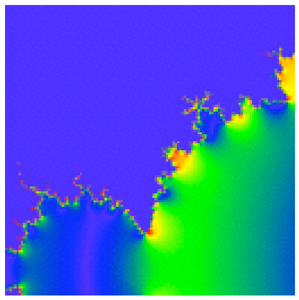
Appendix B - Escape Fractals

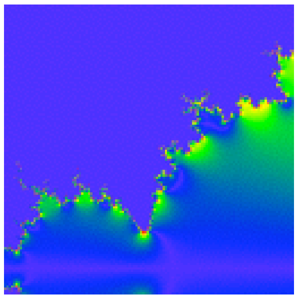
Escape fractals are one of the more complex nodes available in quadrium. These include the various versions of the Mandelbrot fractal, as well as Julia, Dragon, Spider, Thorn and others. Despite their varying appearance, they all produce the same kinds of results, which can be interpreted in a wide variety of ways, producing wildly different images. Escape fractals work by partitioning the image into areas that are "outside" and ones that are "inside". This corresponds to points that "escape" (become increasingly large when evaluated using the underlying formula) and ones that don't. Besides that underlying "inside" vs "outside", quadrium can take additional information (such as the number of iterations that were required before reaching the point of "escape", or the maximum value found while iteration) and use that to produce something more detailed than just the outline of the shape.

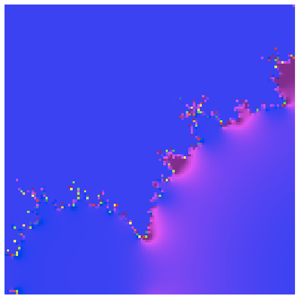
These techniques are the various coloring variations available to all escape fractals. There are a relatively small number of ways to color (and by color, we mean "add more possible values" so that a gradient can produce more than just two colors) the inside, while there are a much larger number of ways to color the outside. Below is a list of the currently available internal colorings (external is set to zero, all other parameters and nodes are unchanged):

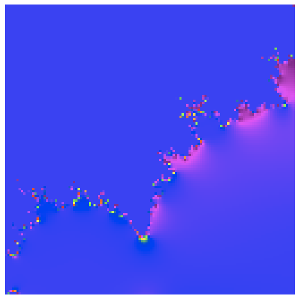

Solid

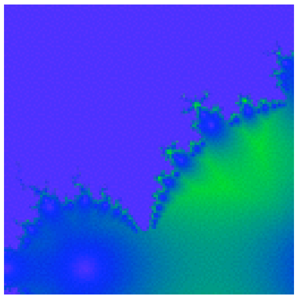

Tension 1

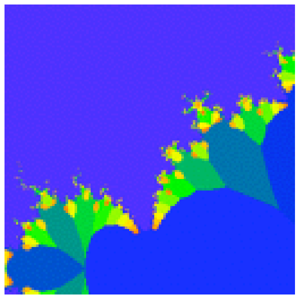

Tension 2

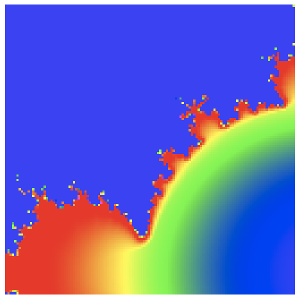

Tension 3

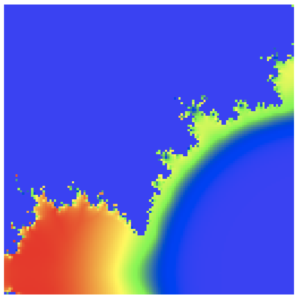

Tension 4

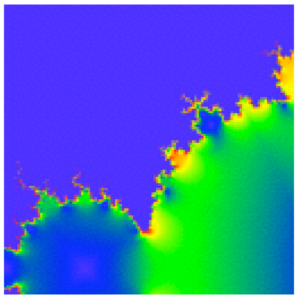

Tension 5

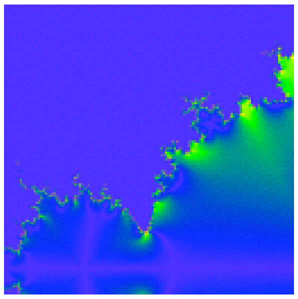

Min Distance

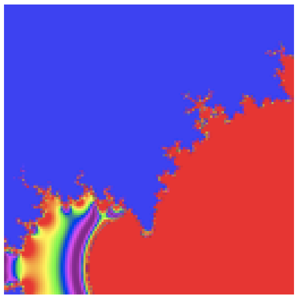

Beauty Index

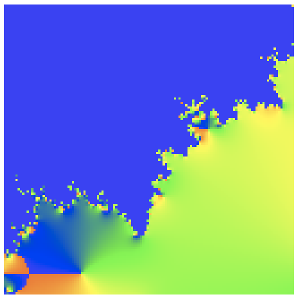

Max Delta


Min Delta

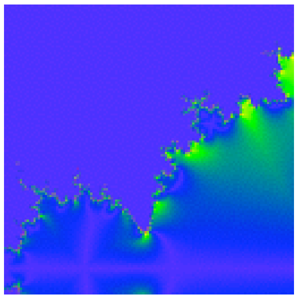

Inner Max

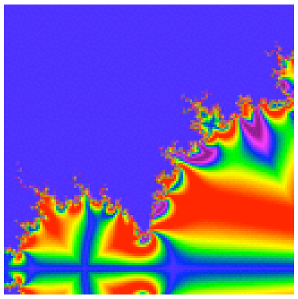

Inner Min


Curvature

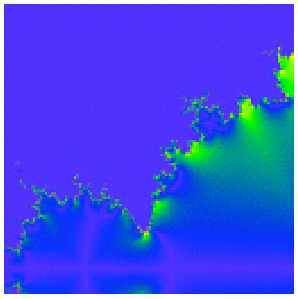

ATan

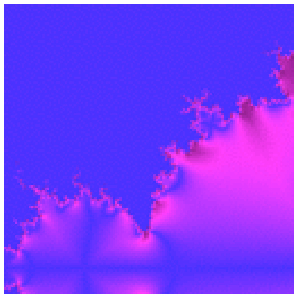
Besides different coloring styles, we can also adjust the "contrast" parameter, which is used to scale the result as show below (since the output goes to a cyclical clut, the result is more bands of color):


Contrast = 0.0

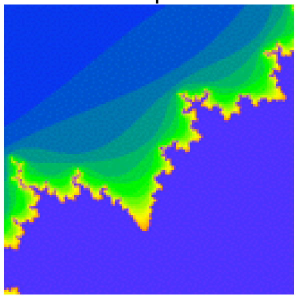

Contrast = 10.0

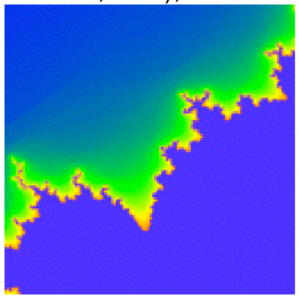
Finally, internal coloring can be regular or inverted. Instead of producing a value that ranges up from zero, if inverted, it goes down from zero (and since the gradient is cyclic, this causes the colors at the end of the cycle to be used instead of the start - this is a way to make it easier to use different sets of colors for the inside and outside):

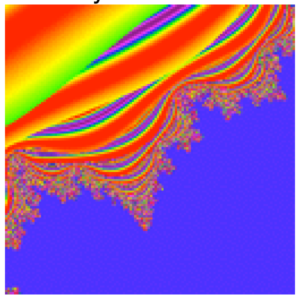

Regular

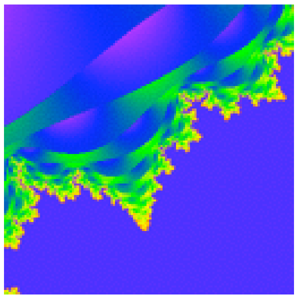

Inverted

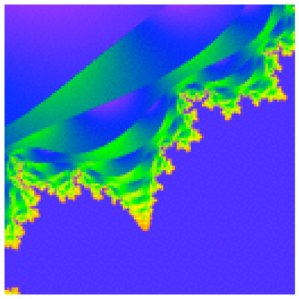
The external color schemes are more varied. Again, all parameters are left set to the same values (with exception of bailout, which is increased for the last set of images, since low bailout values obscure the patterns for Pinecone, etc..), with the internal style set to zero:

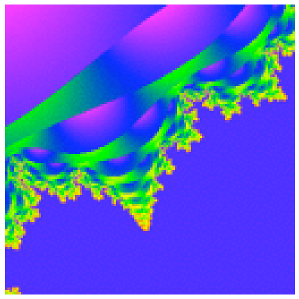

Level

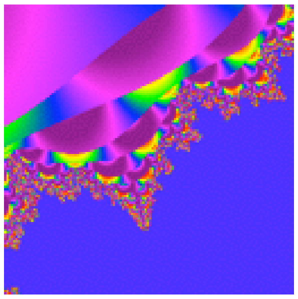

Continuous

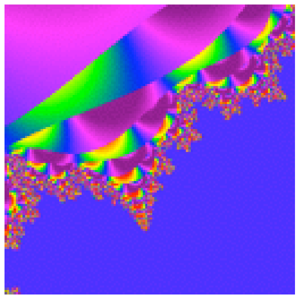

Shaded

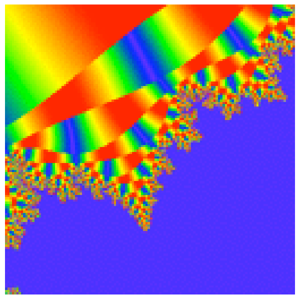

Twisted 1

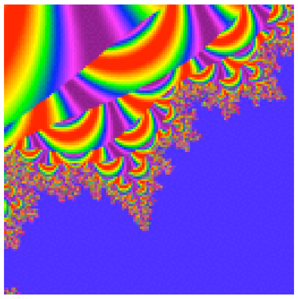

Twisted 2

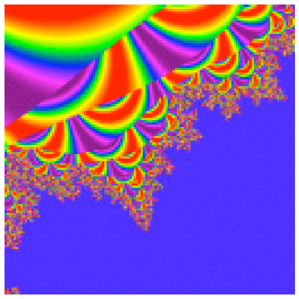

Twisted 3

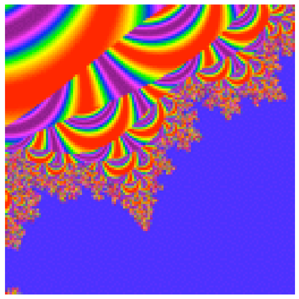

Harsh 1

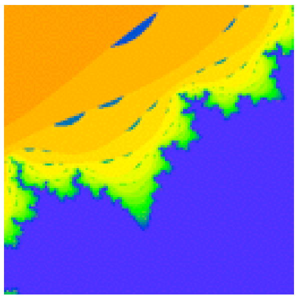

Harsh 2

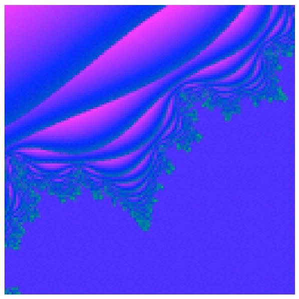

Shiny

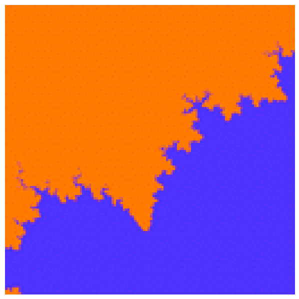

Shattered 1

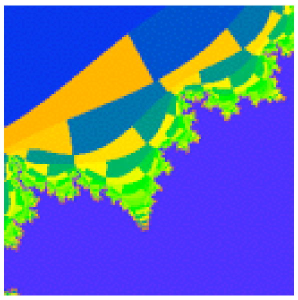

Shattered 2

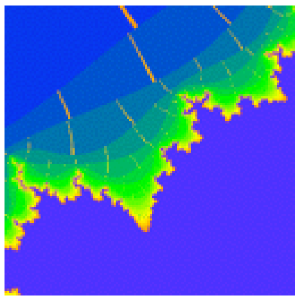

Shattered 3

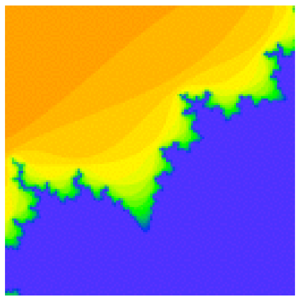

Scalloped

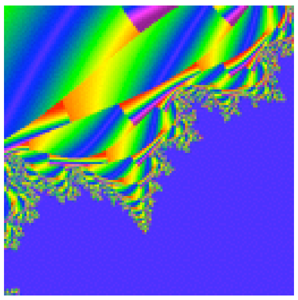

Curved

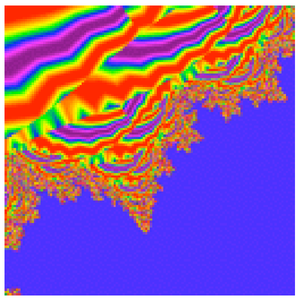

One

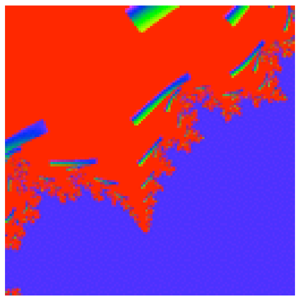

Checkered

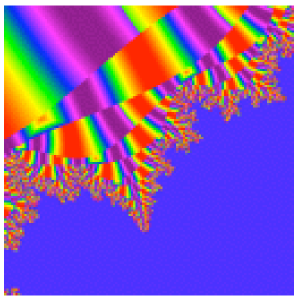

Cellular

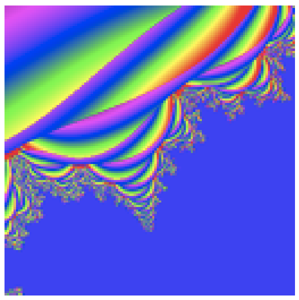

Triangular

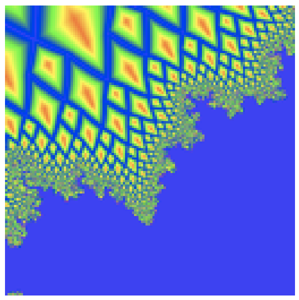

Crossed

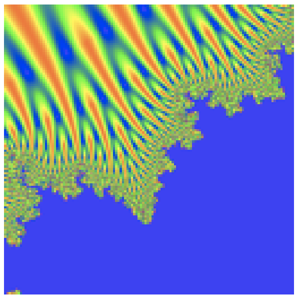

Krinkled

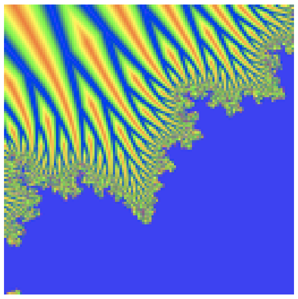

Squares

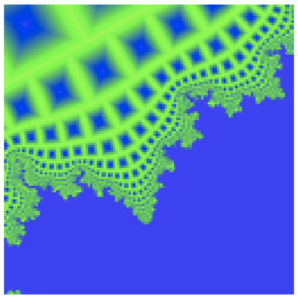

Bricks

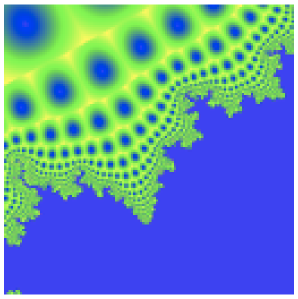

Curvature

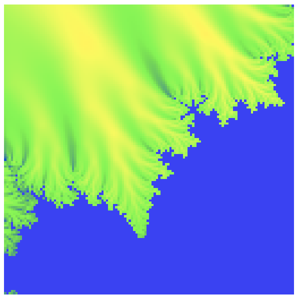

Pinecone

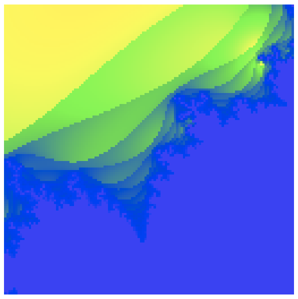

Rivulet


Diamond Tree

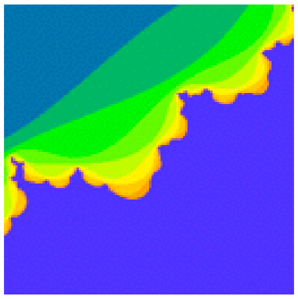

Mosaic

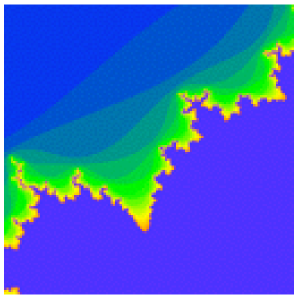

Hobnail

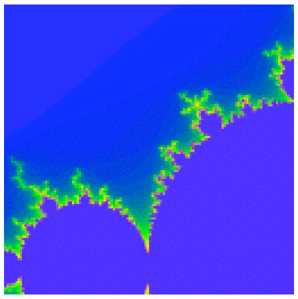

Satin


Distance

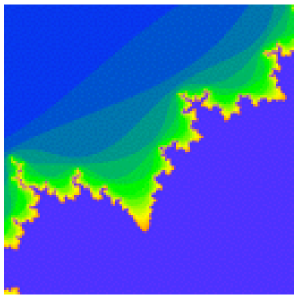
Some coloring styles work better on some fractals than others - you'll need to experiment to see which works the best in any given case. For example, Pinecone, Rivulet, Diamond Tree, Mosaic Hobnail and Satin are designed for simple quadratics such as Mandelbroit and Julia fractals - the 'color focus' slider can be used to help adjust these for optimal results on other fractals. Similarly, the various parameters such as iterations and bailout can change the appearances:

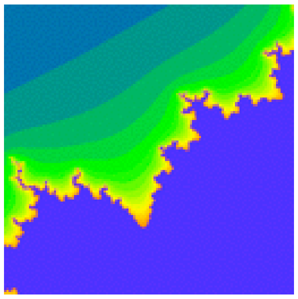

Iterations = 10

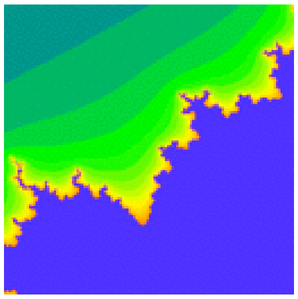

Iterations = 20


Iterations = 50

In general, increasing the iterations adds more detail, and has the side effect of "narrowing" the results as well (not to mention taking more time to display).

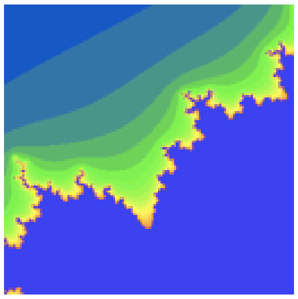

Bailout = 0.0

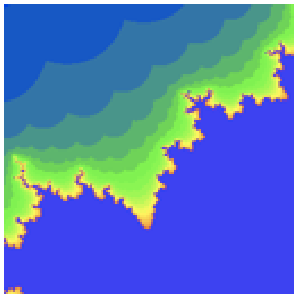

Bailout = 4.0

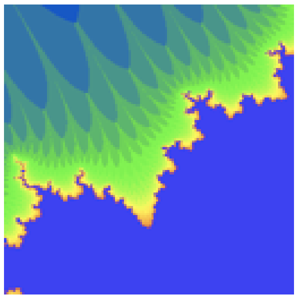

Bailout = 10.0

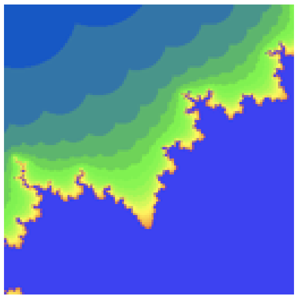
Bailout, however, tends to "widen" the outside area (this varies from fractal to fractal, however).

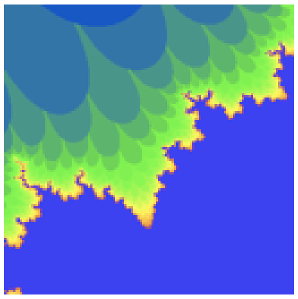
As of version 1.1, there is a new menu that controls how the escape value is calculated, which causes changes in the shape of the "bands" on the outside of the fractal. Previously, the escape value was only calculated by circular distance - there are now several variations available:

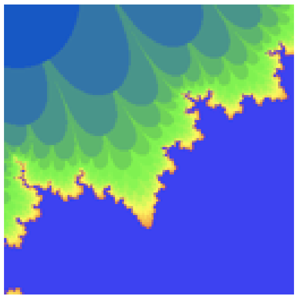

Circle

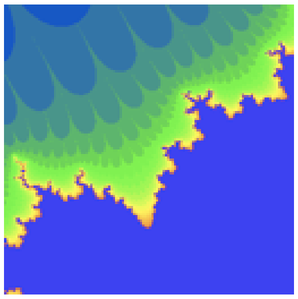

Diamond


Cross

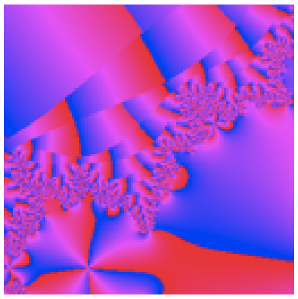

Square

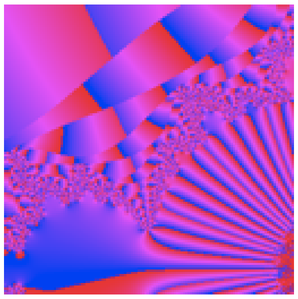

Real

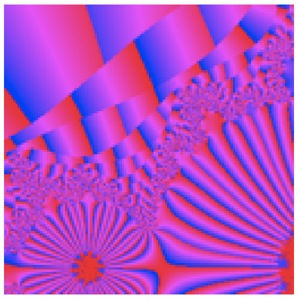

Imag

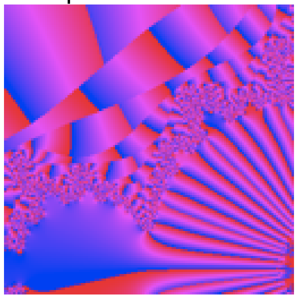

Hyperbola

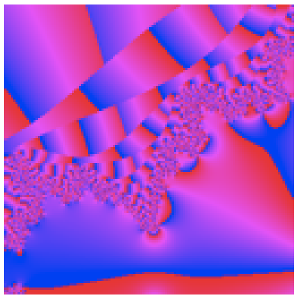
Escape fractals now have three additional outputs - X,Y and Alpha. The X and Y output represent the last state of the calculation - the value when it escapes from the formula. These values can be fed into a "complex colorer" node (which was specifically designed to convert these values into RGB colors) or can be used to adjust gradient colors, or even directly set the final color. There are actually several possible values that can be calculated - either the final value, the change between that value and the previous iteration (or the iteration before that, etc...), the total change between the final value and the starting value, the average value for the most recent iterations, or the total average value. Below are examples of different effects possible with these settings:

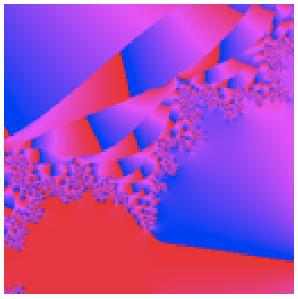

Escape Value

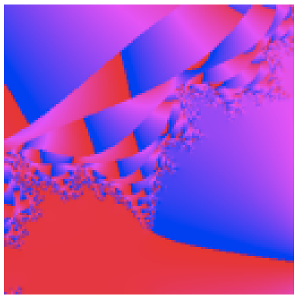

Last Delta


Delta + 1


Delta + 2

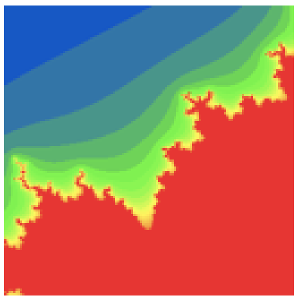

Total Delta

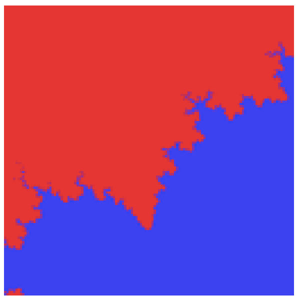

Recent Average

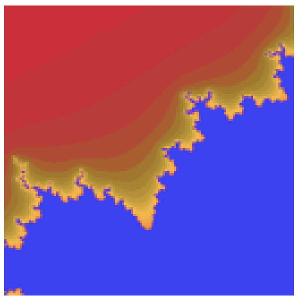

Total Average

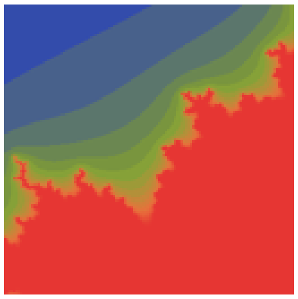
The exact effect of these options will vary greatly with the fractal chosen, as well as whatever method is used to convert this X & Y value into colors (these connections are great to feed into a bump map filter to create "embossed" style fractals)

The other output in quadrium is an alpha channel. This can be used to blend or superimpose the fractal on another image, or the alpha channel can be used to adjust the color value generated from a gradient. The alpha can be set to show or hide any specific feature (inside, outside, or traps - see below), as well as fading them "in" or "out". If we connect the alpha channel to the alpha of the output color, and render with alpha over a solid red color, we can get the following effects:

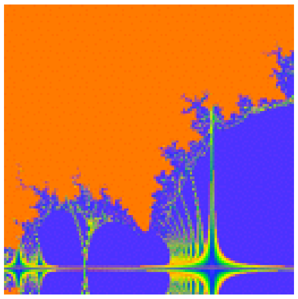

Hide Inside

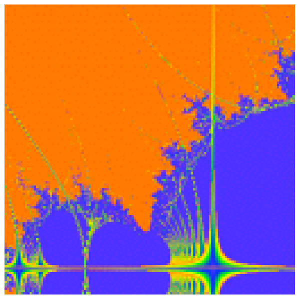

Hide Outside

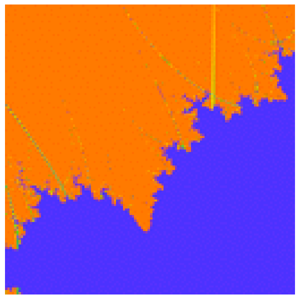

Fade Exterior


Fade Interior, Hide Inside

Traps are ways to enhance an escape fractal. The way they work is that while calculating the each iteration of the fractal, quadrium examines the value and see if it falls within a given area, which can "trap" that value (and later be used to produce different coloring). This results in adding "structures" to the fractal (and that structure is shaped similar to the resulting fractal). This is very flexible, because it can be applied to both the inside and outside (or just one or the other):

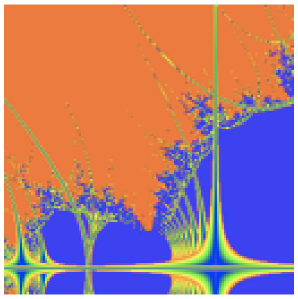

Plain

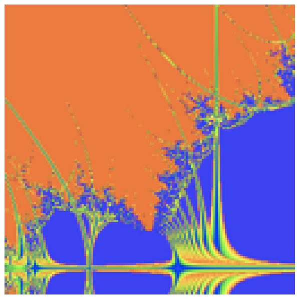

Super

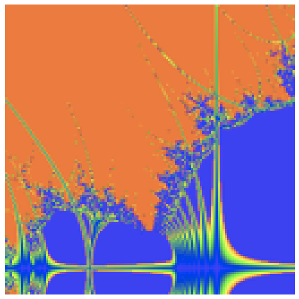

Semi

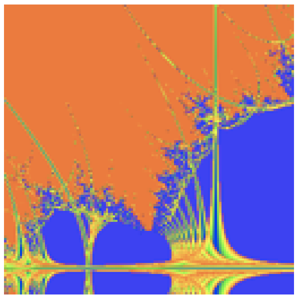
Plain is applied only to the inside, super is both inside and outside, while semi is outside only. Note that large bailout values tends to cause traps in the external area to often become overwhelmed in "noise".

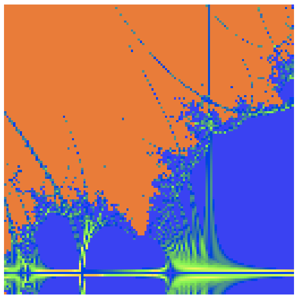
Furthermore, quadrium can use either the first or last time the value crosses into the trap, or which trap is "closest" or "furthest", or any wide variety of blending

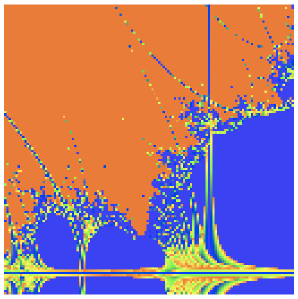

First

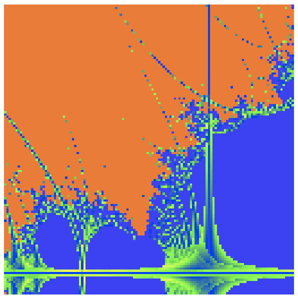

Last

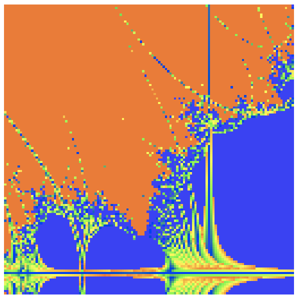

Closest

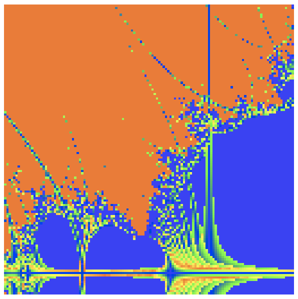

Furthest

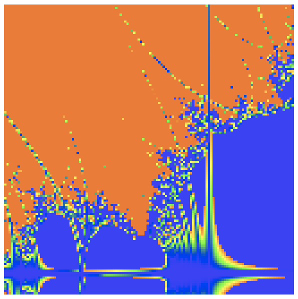

Blend In

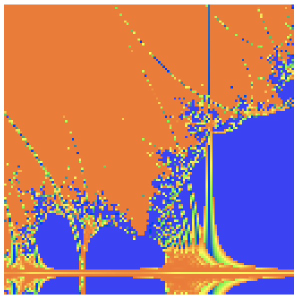

Blend Out

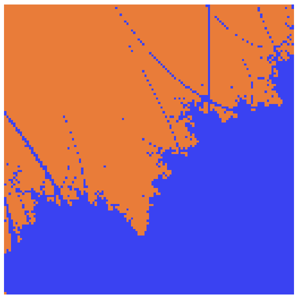

Blend Over

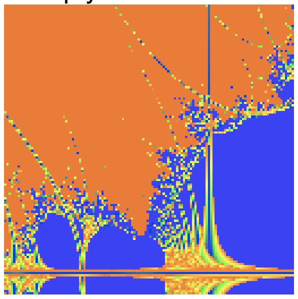

Blend First


Blend Last

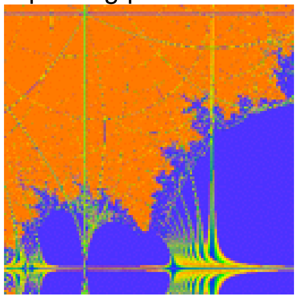

Multiply


Screen


Min

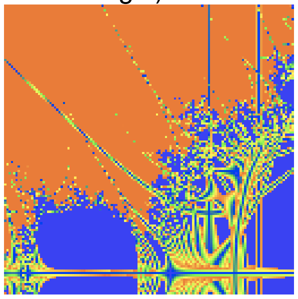

Max

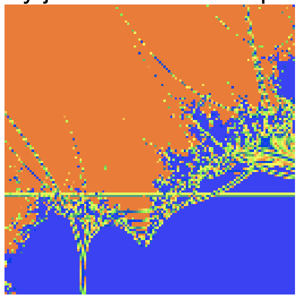
Besides just a single trap, we can have a large grid of traps, which will produce a lace like mesh, or repeating pattern of the underlying trap shape:

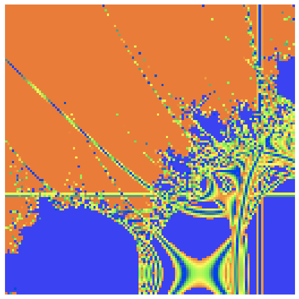


Grid on

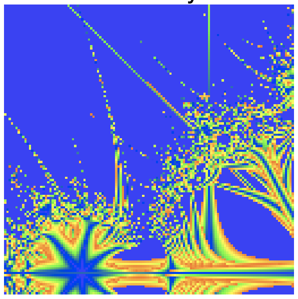
quadrium2.1 introduces the ability to move a trap. There are two different techniques - the first is the ability to change where the trap is located (previously, the trap shape was assumed to be centered on the origin). This basically just moves the trap:

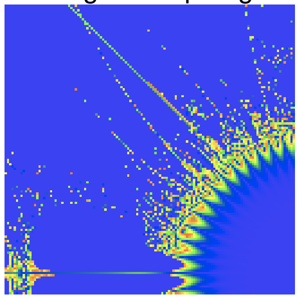

Trap X Origin -0.3

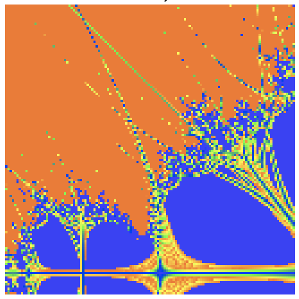

Trap Y Origin -0.3

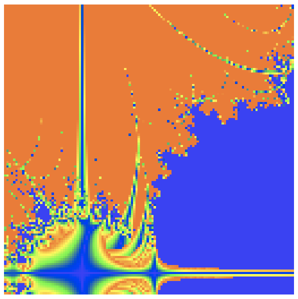

Trap X & Y Origin -0.3

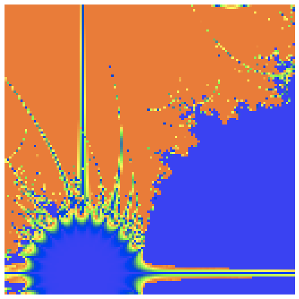
The other ability results in moving the trap origin at each iteration, based on the value itself.


Relative


Delta

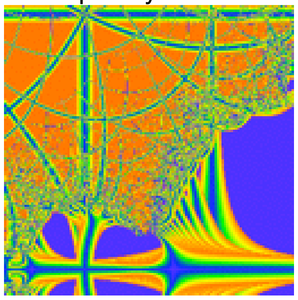

Product

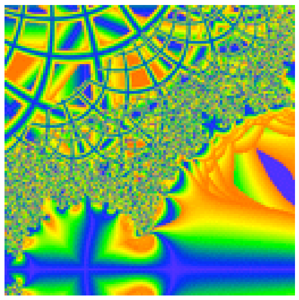

Average

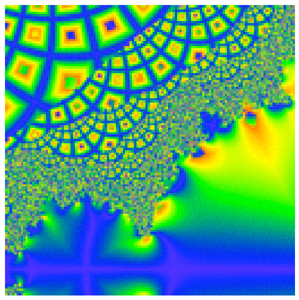

Lagging

The effect is even more noticeable and interesting when zoomed into some of the fractal details.

The "trap limit" parameter controls the size of the trap - the smaller the limit, the larger the resulting trap appears. Taken to extremes, the resulting traps (especially if grid is enabled) will completely overwhelm the underlying fractal, resulting in a image similar in shape to the original fractal, but with a completely different texture:

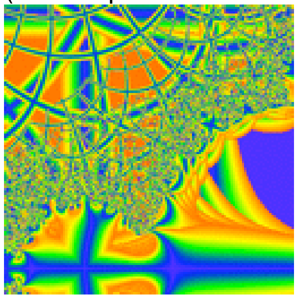

Trap Limit = 1.5

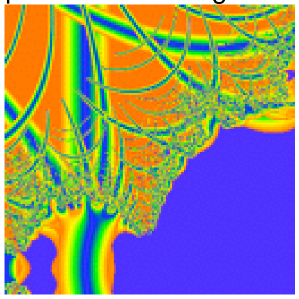

Trap Limit = 0.8

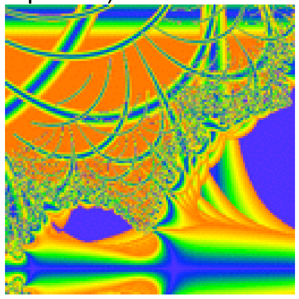

Trap Limit = 0.4

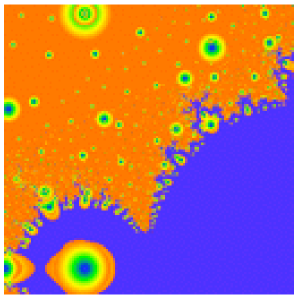
The traps are based on the same shapes as are used for Regular Shapes texture node (among others), with many additions. Omega Cross is the underlying X and Y axis of the image (and if grid is enabled, it effectively is like the Grid Lines texture node). Omega Real is just the real component of the number (the X coordinate) while Omega Imag is the imaginary component of the number (the Y coordinate). Thus, Omega Cross is the same as combining Omega Real and Omega Imag.

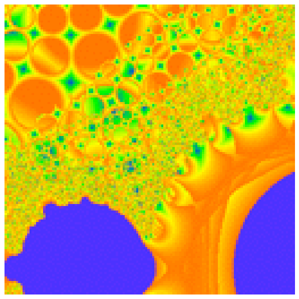
(NB: Trap Limit and other parameters changed for emphasis)

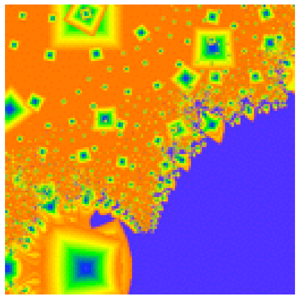

Omega Cross

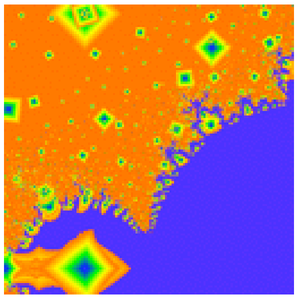

Omega Real

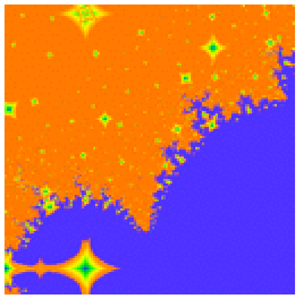

Omega Imag

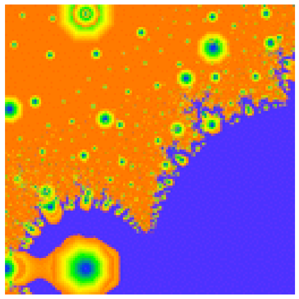

Circle

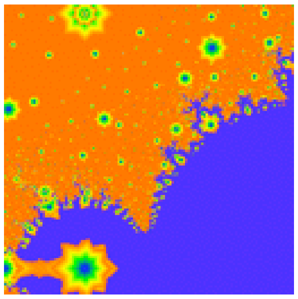

Ring

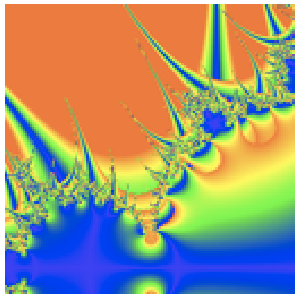

Square

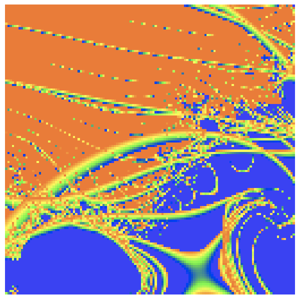

Diamond

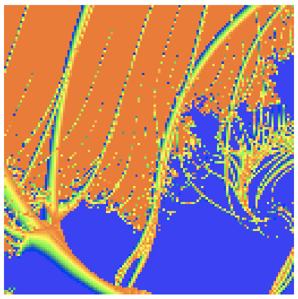

Astroid

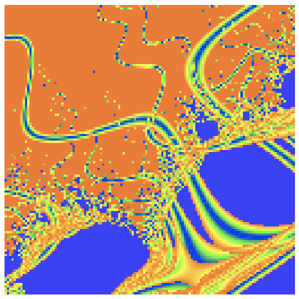

Gem

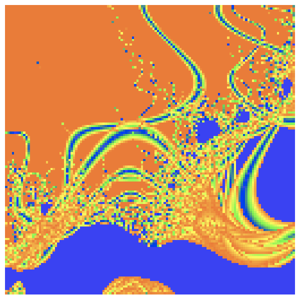

Star

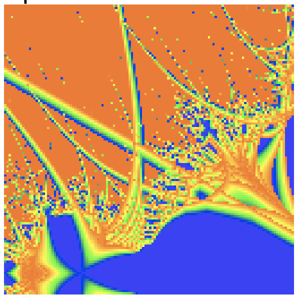

Hyperbola

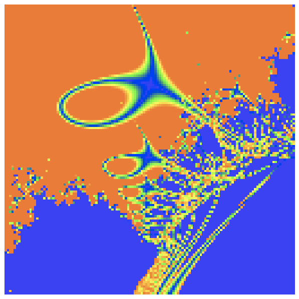

Spiral +

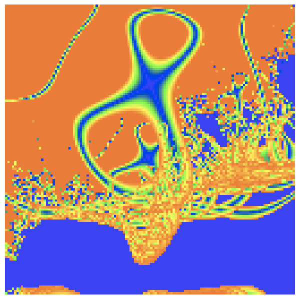

Spiral -

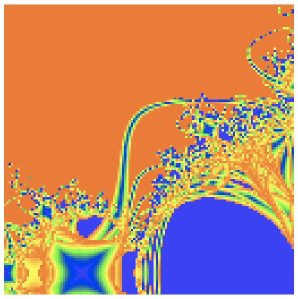

Waves X

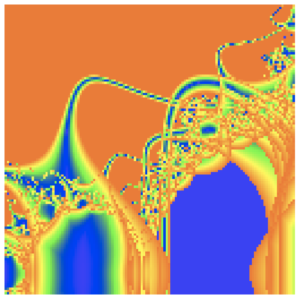

Waves Y

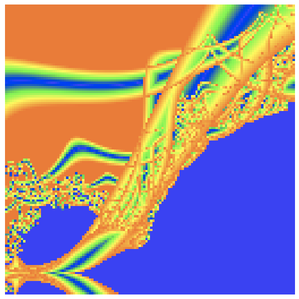

Petals

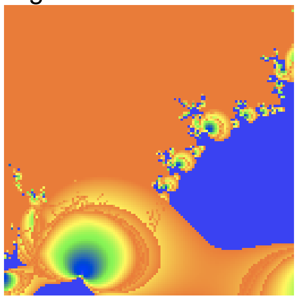

Decarte Folium

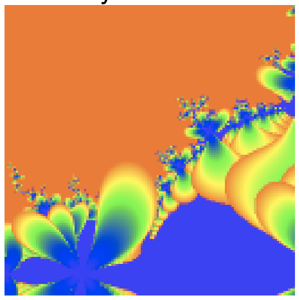

Devils Curve


Eight Curve

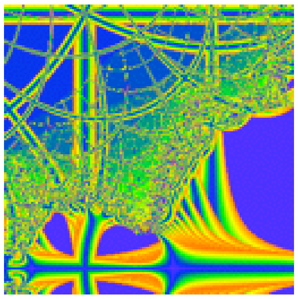

Butterfly

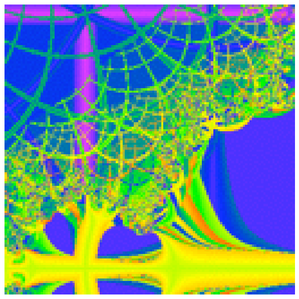

Lips


Cardoid


Lemniscate

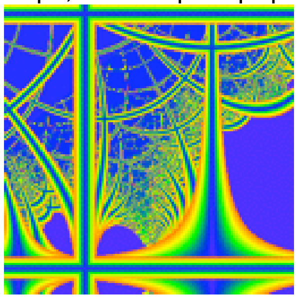
Finally, there are some other coloring styles for the traps - their basic coloring (set to a solid value, or some subtle shading) as well as a way to "partition" the traps between even and odd (but extended to more than just two - all the way up to eight). The actual effect is often difficult to anticipate, and sometimes quite subtle, so it's worth playing with to see which looks best for your particular case:

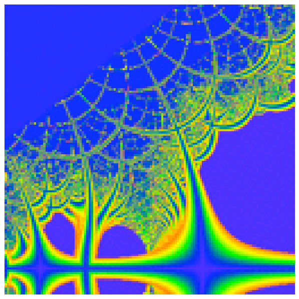

1 Level

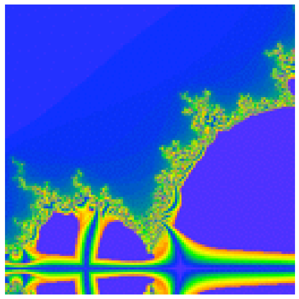

5 Levels

Note how in "1 Level" all the traps have the same color, while at 5 there are a number of different colors.

Often with traps, the first trap is often large and completely obscures subsequent traps (especially with the traps set to show the "First" trap as opposed to the "Last" trap). To hide some of the earlier traps, the "Trap Skip" parameter can be increased:


Trap Skip = 0.0


Trap Skip = 0.076


Trap Skip = 0.273

Notice how the traps on the outside disappear when the trap skip parameter is increased - this is because those values initially hit one or two early traps, and then escape, while the internal traps often hit many traps. If you view the these traps as "layers" you can see how the top trap layers are "peeled" off to reveal detail below it.

Appendix C - Custom Formulas

In quadrium, one normally connects various nodes together to form the mathematical basis for an image. This works well for many things, but there are some special cases where it is easier to type in a formula than to try to wire things up and set the parameters accordingly. Also, for various fractals, there was no way to experiment with arbitrary formulas (since you couldn't rewire things "inside" a node). quadrium 1.1 solves this problem by introducing special "user custom formula" versions of these various nodes.

In general, these formulas follow the syntax of algebraic formulas found in spreadsheets or other computer languages. There are, however, some special extensions provided to support *complex math* (which is extremely common in fractal research). If you are unfamiliar with complex math, the basic idea is that instead of just a single number, there is actually a pair of numbers, representing a coordinate on the "complex plane". The first number is the *real* number (which corresponds to regular, ordinary numbers), and the second is the *imaginary* number (so called because it represents a multiple of the imaginary value *i*, which represents the square root of -1). There are then various rules to explain how to apply standard mathematic operations on these complex numbers, but the details of this aren't important here - for now, you can just think of these as an (x,y) coordinate pair.

Types
quadrium formulas support the following types:

Type	Details
integer	Any whole number, such as 5, -3, or 1052
float	Any real number, such as 1.34, -65.232, or 634.3421
complex	A complex number {1.3, 0}, {12, 23.54}, {-5.5, 0.12}
boolean	The result of a comparison such as x > 5.3, or z == {0,0} or the constants **true** or **false**.

As you can see, complex numbers are written inside braces '{', '}'. The first number is the real component, and the second is the imaginary component. There is very little difference between "integers" and "floats" - there are only a few special cases where integer values are treated specially (for example, integer powers vs. real powers), but they are otherwise interchangeable.

For the most part, complex numbers and floats can also be interchanged, and quadrium will automatically promote real numbers to complex where needed (since real numbers are complex numbers with their imaginary component set to zero). However, there are places where real numbers are required (and don't make sense for complex numbers), and will produce an error if a complex value is provided.

Booleans, however, are another case - they are never promoted to floats, complex numbers or integers, nor are those every converted implicitly to boolean values. If a function uses a boolean value, it must be from a comparison, or the boolean constants.

Operators
Below is a table of the operators and their precedences (i.e, multiplication is performed before addition).

Class	Operators	Example
Nested expressions	()	x * (5 + 3)
Unary	-	-y
Power	^	x ^ 4
Multiplication	*, /	{3, 1.5} * x / 2
Addition	+, -	x + {4, 5}
Comparison	<,<=, >, >=, ==, !=	x > 5
Boolean 'and'	&	0 < x & x < 10
Boolean 'or'	|	x < 0 | x > 10

Functions
Besides the various common math operations, there are a number of special functions. The table below lists the functions. Unless otherwise stated, functions take a single real or complex parameter, and return the a value that is the same type (so sin(real) = real, sin(complex) = complex). Note that user functions are case insensitive, so "Sin", "sin", and "SIN" all refer to the same function.

Function	Notes
sin	Sine
cos	Cosine

Function	Notes
tan	Tangent
cosh	Hyperbolic cosine
sinh	Hyperbolic sine
tanh	Hyperbolic tangent
log	Logarithm
exp	Exponent (e^x)
sqrt	Square root
abs	Absolute value
norm, cabs	Normal (same as abs for real parameters, which is why the function can also be called “cabs”)
flip	Exchange the real and imaginary components of a complex number
conj	Complex Conjugate (real parameters are unchanged)
min	Minimum - only supported for real parameters, and can have two or more parameters
max	Maximum - only supported for real parameters, and can have two or more parameters
ave	Average, supports two or more parameters (all must be real for the result to be real, otherwise everything is converted to complex number)
if	Takes three parameters - the first is a boolean (such as a comparison), if that is true, then the function evaluates as the second parameter, otherwise it evaluates as the third parameter.

The table below lists functions that convert from complex numbers to real numbers. If a real number is passed as the parameter, either that original value or 0.0 will be the result (depending on the function).

Function	Notes
real	Real component (does nothing for real parameters).
imag	Imaginary component (same as 0.0 for real parameters).
theta	Theta, the “angle” of the complex number (0.0 for real parameters)

Parameters and Constants
Nodes that use custom formulas will have different ways of allowing access to the input values and the values of various sliders. In all cases, however, the result of the expression is used for output (or the next value for fractals). Similarly, the input connections are always referred to as “input1”, “input2”, etc... (Note that some nodes may refer to these values as other parameter names as well). The sliders are, in general, the same as their names (spaces and other punctuation are removed, combining the words of the title together).

There are also some common constants and other globally available values:

Name	Value
pi	3.1415926535897931
e	2.7182818284590451
sqrt2	1.4142135623730951