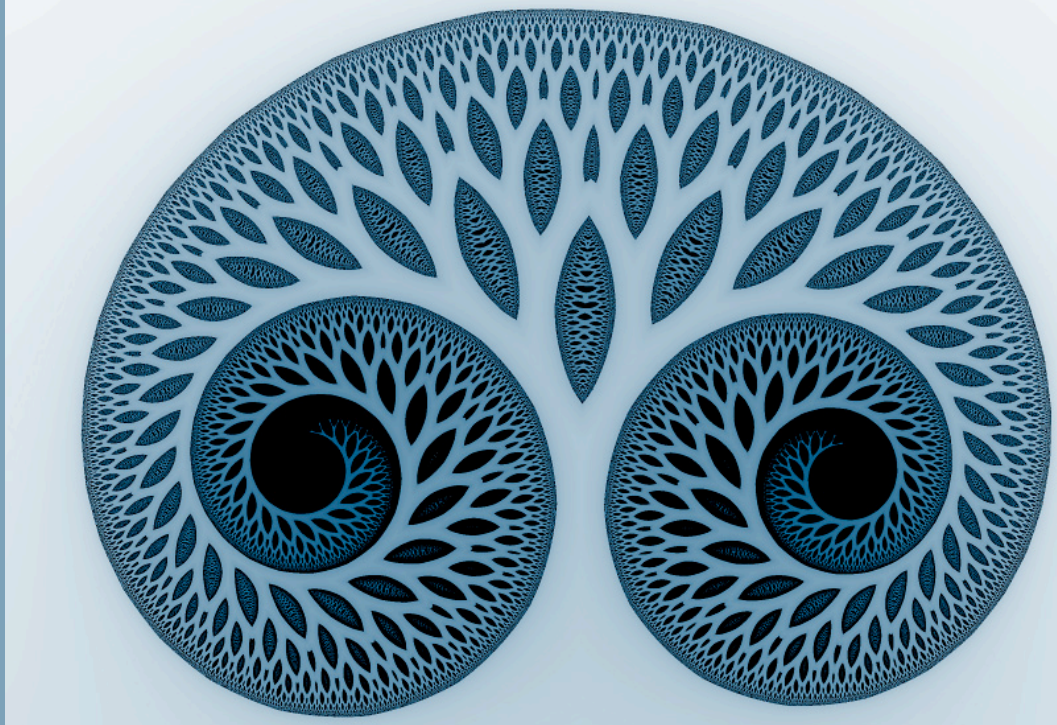
FRACTALEXPLORER

Pixel Bender filters for Adobe Pixel Bender Toolkit, Photoshop CS4 and After Effects CS4

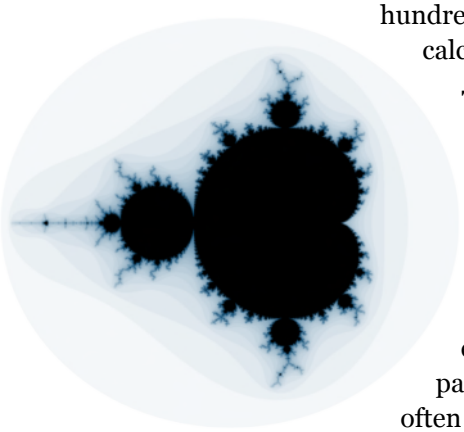


The Glynn Fractal
Julia set.
Power: 1.5
mu: -0.2, 0

Introduction

Benoit Mandelbrot was the first person to study the function $z' = z^2 + c$ with a computer and the result is the iconic fractal shape that now takes his name [1].

Rendering fractals is a computationally expensive process often requiring hundreds of repeated calculations per pixel.



Traditional fractal programmes would take seconds, minutes or even hours to render high resolution images making exploration a tedious process, especially as tiny parameter changes can often produce dramatically different results.

By utilising the highly optimised parallel pixel processing capability of a modern graphics card GPU it is possible to generate high resolution fractal images in real-time.

Adobe recently released the Pixel Bender Toolkit which enables you to write image processing filters and effects that run on the graphics card GPU.

These filters can be used in Photoshop, AfterEffects and Flash CS4 dramatically opening up the potential for custom plugin creation.

Fractal Explorer is actually two Pixel Bender filters that will generate Mandelbrot and Julia set fractals to any power. The first filter is for standard fractal colouring whereas the second is optimised to use a technique called 'orbit trapping' to map an image into fractal space.

Getting started

Installation

Download and install the free Pixel Bender Toolkit (PBT) and plugin for Photoshop CS4:

http://labs.adobe.com/technologies/pixelbender

If you don't have Photoshop CS4 you can still create and save images directly from PBT.

Download the Fractal Explorer filter from my website:

http://www.subblue.com/projects/fractal_explorer

If you are using the Pixel Bender plugin for Photoshop then copy the `.pbk` files into the Pixel Bender Files folder in your Photoshop installation directory.

For use in After Effects CS4 copy the `.pbk` files into the Plugins/Effects directory.

Quick start

Open the `FractalExplorer.pbk` file in PBT and press the Run button. You should see something like the image below.

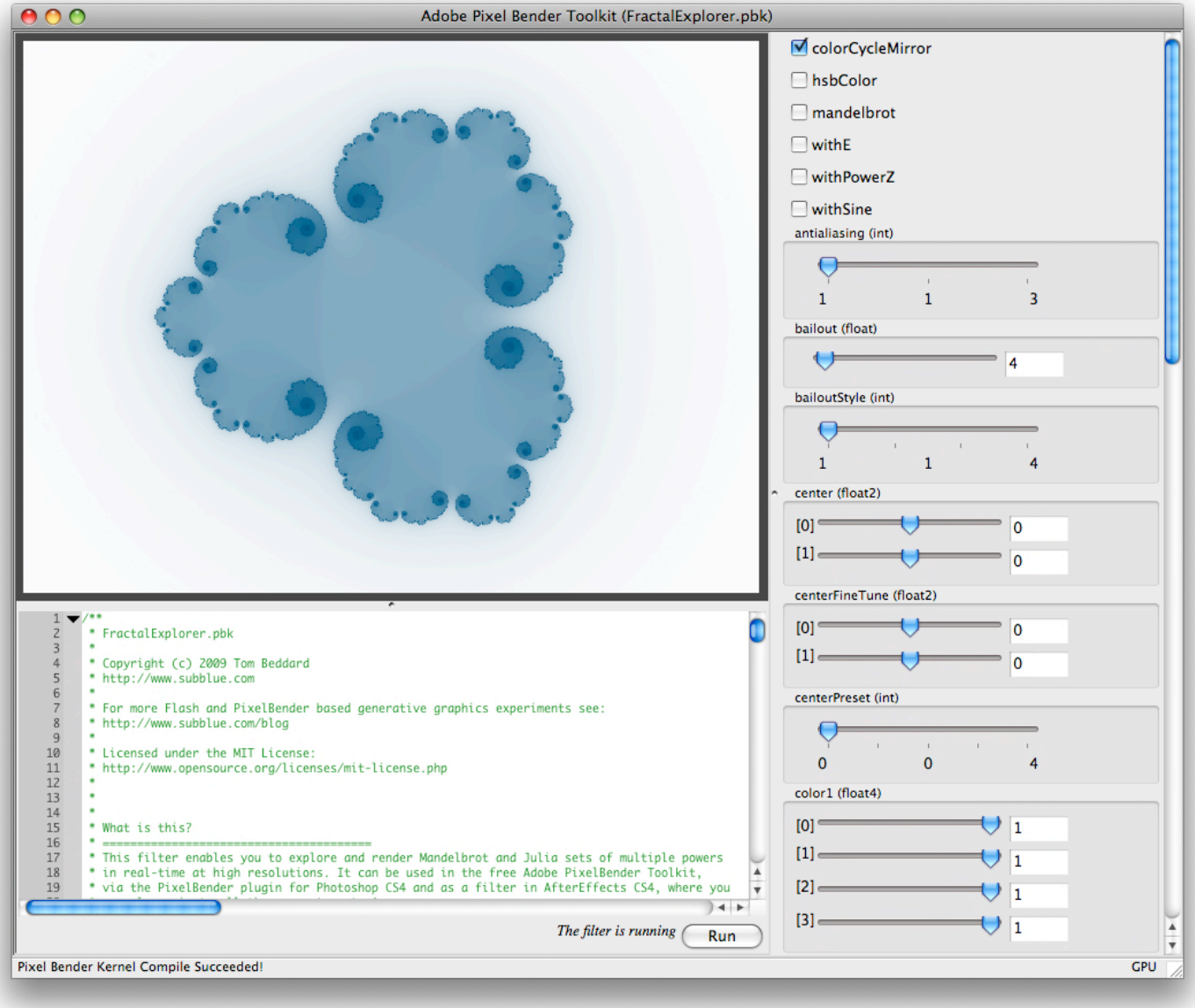
The default settings are of a power 3 Julia set fractal. The filter uses the general formula:

$$z' = z^{power} + \mu$$

where z and μ (mu) are complex numbers.

For each pixel *z'* is repeatedly calculated and fed back into the equation until the magnitude of z exceeds a 'bailout' value or the maximum number of iterations is reached.

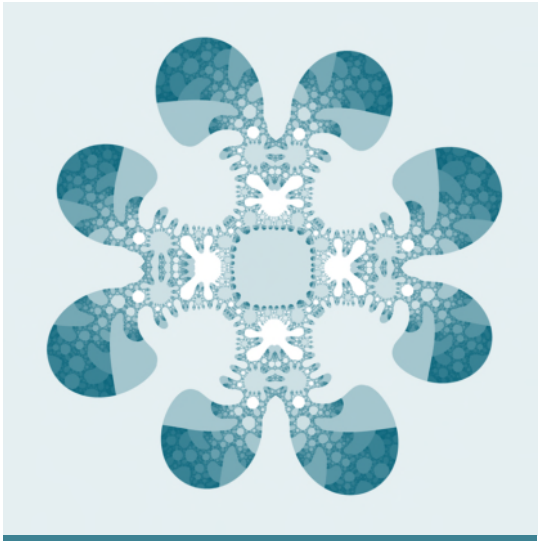
The number of iterations each pixel takes to reach the bailout value determines its colour.



The Julia set differs from the Mandelbrot set in the way the μ parameter is used. In a Julia set μ is the same for every pixel, whereas with the Mandelbrot set μ is defined by each pixel's coordinate as the real and imaginary values of the complex number μ = x + y*i*.

Have a play around with the `power` and `mu` parameters. These are the main controls for exploring the fractal space.

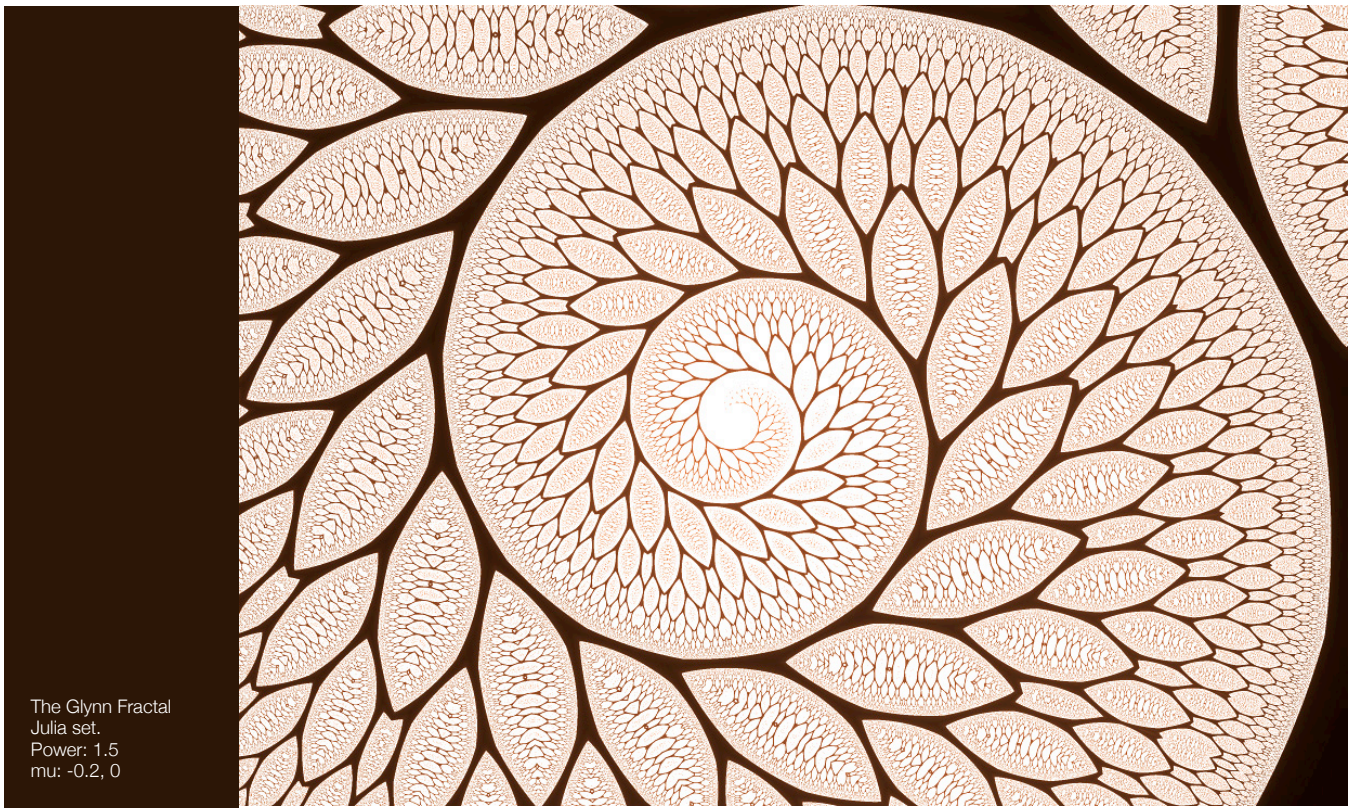
Try the `centerPreset` to quickly explore some interesting areas of the Mandelbrot set. Note, you won't be able to change the `center` or `power` parameters when the centerPreset is being used.



Unusual non-typical fractals can be generated using negative powers and the iterationOffset parameter.

Tips

- Set the `size` parameter to the output size you want to render the fractal at.
- Keep the `iterations` parameter as low as possible, only increase to add detail at the edges.
- Increase the `antialiasing` setting only when you are ready for final render.
- A whole number `power` value will give the most symmetrical results.



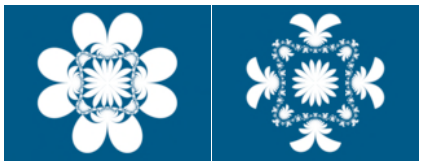
The Glynn Fractal
Julia set.
Power: 1.5
mu: -0.2, 0

General Parameters

antialiasing
Crude oversampling. The number of samples^2 per pixel, so 1, 4 or 9.

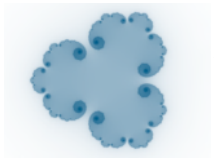
Only increase this value once you are ready for the final rendering.

bailout
This sets the limit for the fractal calculation to stop at. This is best used in conjunction with the `bailoutStyle` and `colorMode` settings as shown below.

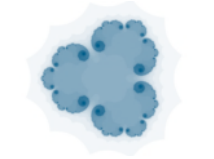


Low bailout | Higher bailout

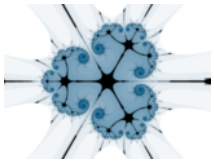
bailoutStyle
This changes between different methods of calculating the bailout condition to stop the calculation.



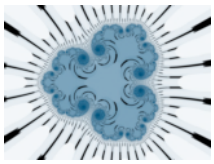
Style 0: smooth gradient



Style 1: spiky



Style 2: stalks

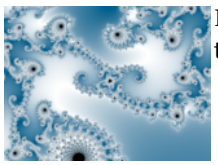


Style 3: swirls

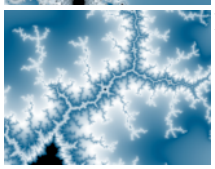
The `bailout`, `colorScale` and `colorCycle` controls are useful to tweak when changing the `bailoutStyle`.

center & centerFineTune
Pan the fractal. The fine tune option is handy for small tweaks especially at large zooms.

center preset
Three preset areas of interest in the Mandelbrot set.
When greater than `0` the other controls apart from zoom and `centerFineTune` are disabled.
Set back to `0` for manual control.

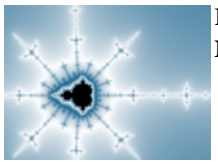


Preset 0: seahorse tail



Preset 1: lightning

Preset 2: spiral



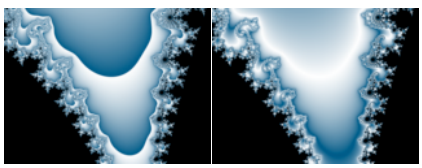
Preset 3: mini Mandelbrot

color 1 & 2
Inside and outside colours. The four slider components correspond to red, green, blue and alpha, (RGBA).

colorBackground
The background colour, RGBA.

colorCycle
The number of times the colour gradient will repeat across the range of the fractal.

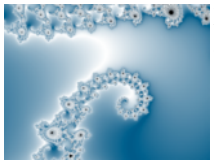
colorCycleMirror
This will reflect the colour gradient so that it cycles smoothly.



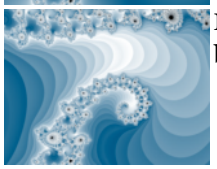
colourCycleMirror disabled | colourCycleMirror enabled

colorCycleOffset
Change the start point for the gradient colouring.

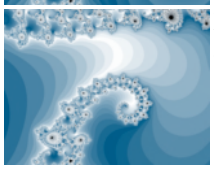
colorMode
There are 6 different colouring modes that can be used to greatly change the appearance of the fractal.



Mode 0: smooth colouring



Mode 1: shelf banding



Mode 2: solid colour banding

Mode 3: binary deconvolution

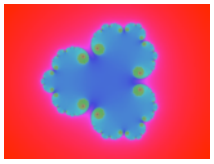
Mode 4: contour bands

Mode 5: spikes

Modes 3-5 create two-tone mappings. Increasing the bailout value will change the shape of the coloured areas.

colorScale
Controls the strength of the colour mapping from `color1` to `color2`.
Increase to get greater contrast.

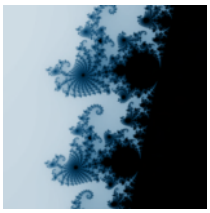
hsbColor
This changes the colour sliders to be hue, saturation and brightness mode instead of RGB.



iterations
This sets the number of iterations for each pixel before `bailout`.

Increase this to get more detail into the edges of the fractal as illustrated below:

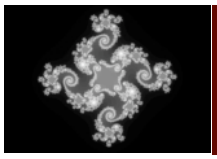

Low iterations


High iterations

iterationsOffset
This will ignore results that bailout less than this value. It can be useful to reveal details when using negative powers.

power
This is the power that the fractal equation is raised to.

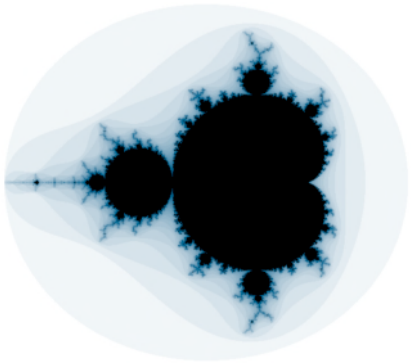
The standard Julia and Mandelbrot sets have a power of `2.0`


Power 4.0


Power 8.0

powerFineTune
For fine tweaking. The most symmetrical results will be achieved with whole number powers.

mandelbrot
Switch from the default Julia mode to the Mandelbrot set mode.



In Julia mode the `mu` value is used as the starting point for every pixel whereas in Mandelbrot mode `mu` is derived from the pixel coordinates, so the mu sliders won't do anything here.

mu & muFineTune
These parameter sliders define the real and imaginary parts of the value `mu` in the fractal equation.

Changing these will give the biggest variety of images. When you hit on something interesting use the `muFineTune` sliders to tweak it more accurately.

rotate
Rotate the fractal plane around the origin.

size
This is the output size of the rendered fractal.

withE
Adds e^z to the fractal equation:

$$z' = z^{power} + e^z + \mu$$

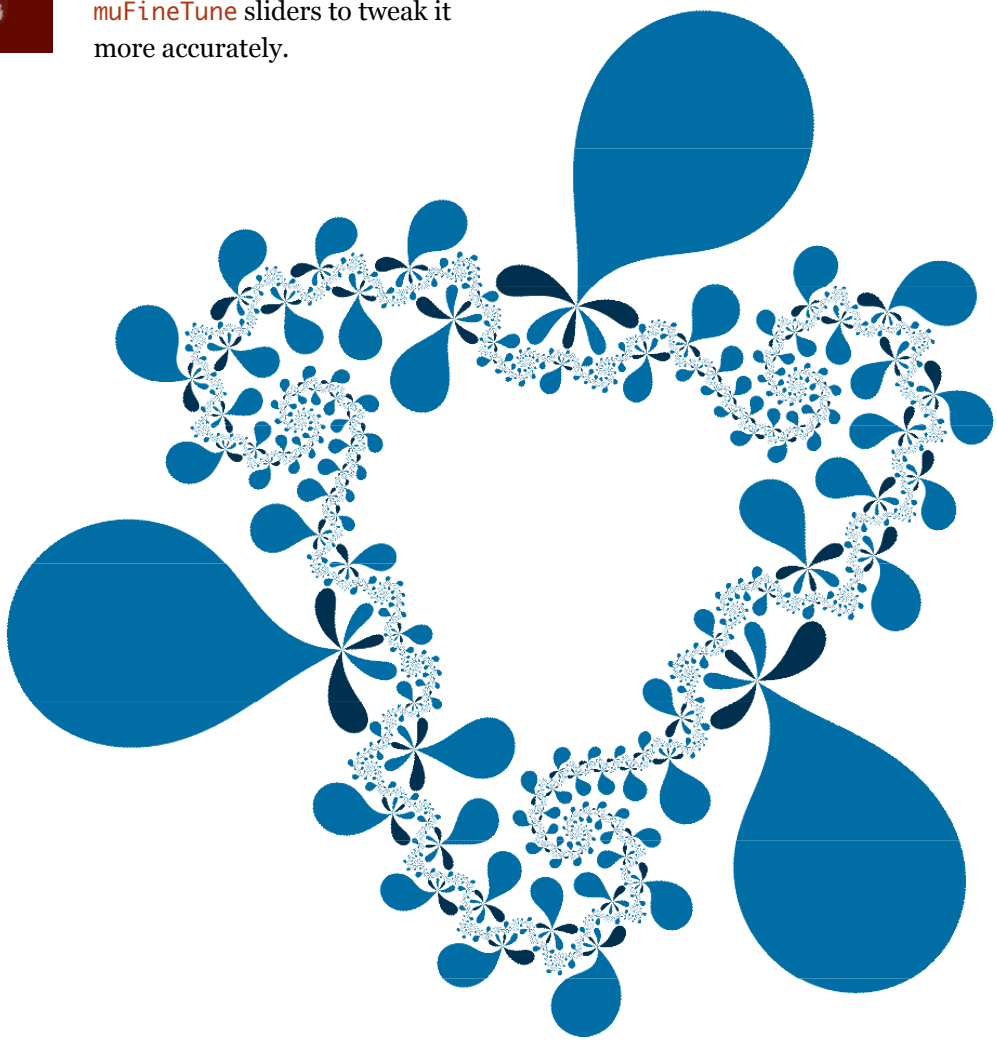
withPowerZ
Adds z^z to the fractal equation:

$$z' = z^{power} + z^z + \mu$$

withSine
Adds sin(z) to the fractal equation:

$$z' = z^{power} + \sin(z) + \mu$$

zoom & zoomFineTune
Main zoom which increases exponentially. Use the fineTune option to zoom linearly once that area of interest has been found.



Orbit Traps

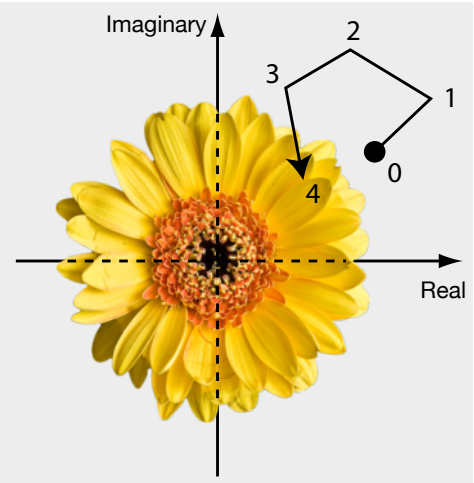
A Venetian mask mapped onto a 3rd power Julia set.

Source image credit: http://www.flickr.com/photos/audringje/535107837/

What are orbit traps?

Orbit traps were first described by Clifford Pickover [2] and is surprisingly straight forward.

For each iteration calculating z' in the fractal equation the real and imaginary parts of z' are mapped to x, y coordinates. The position of z' jumps all over the place and is called the orbit, see below.



If you 'trap' or stop the iteration loop when the position of z' matches some criteria (in our case a non transparent pixel in the source image) and use the colour from the trapped position then you end up mapping the image into fractal space.

How to use

Prepare a suitable source image with a transparent outer area. The best results will be obtained with images that have a very clean transparent edge.

Run the `FractalExplorerOrbitTraps` filter. The first thing to set is the `sizeInput` to match the source image size.

Adjust the `orbitTrapScale`, `orbitTrapOffset` and `orbitTrapRotation` to refine the image mapping once an interesting fractal shape has been found.

Keep the `iterations` low until you are ready to add more detail to spirals or the edges of fractals.

The `iterationsOffset` parameter is useful to prevent large image versions from obscuring more interesting details.

Uncheck the `orbitTrap` option to show the standard fractal colouring to help quickly find interesting fractal shapes.

Orbit Trap Parameters

orbitTrapEdgeDetail
This sets the threshold for the transparency level required to trap the orbit. Changing in conjunction with the `backgroundColor` alpha slider will remove give a smooth edge to the image shape by removing fringing.

orbitTrapOffset
By default the image will be centered at `0`,`0` on the x,y plane. The offset will push it to one side allowing you to fit the image more accurately into the fractal pattern.

orbitTrapRotation
Rotates the image centered at the origin before it is translated via the `orbitTrapOffset`.

orbitTrapScale
Scales the image relative to the x,y plane.

orbitTrapSpin
This rotates the translated image around the origin of the x,y plane.

sizeInput
This should match the pixel size of the source image. If set too small clipping might occur.

sizeOutput
The size of the rendered fractal.

A flower mapped onto a 4th power Julia set.

The same flower but with tweaked offset and mu parameters.

Credits & License

Author

The Fractal Explorer Pixel Bender filters were created by Tom Beddard, tom@subblue.com

Download the latest versions at:
http://www.subblue.com/projects/fractal_explorer

For other fractal and generative art projects see:
http://www.subblue.com

Last updated: *14 July 2009*

Further reading

[1] “The Fractal Geometry of Nature”, Benoit B. Mandelbrot, 1983, Freeman

[2] “Computers, Pattern, Chaos and Beauty”, Clifford A. Pickover, 1991, St Martins Press

License

The Fractal Explorer filters are licensed under the MIT License:
http://www.opensource.org/licenses/mit-license.php

Finally...

If you have fun or success using these filters for personal or commercial projects then please let me know (tom@subblue.com).
It’s good to get feedback on the work that has gone into this!