Saturn & Titan
Version 4.1 User Manual

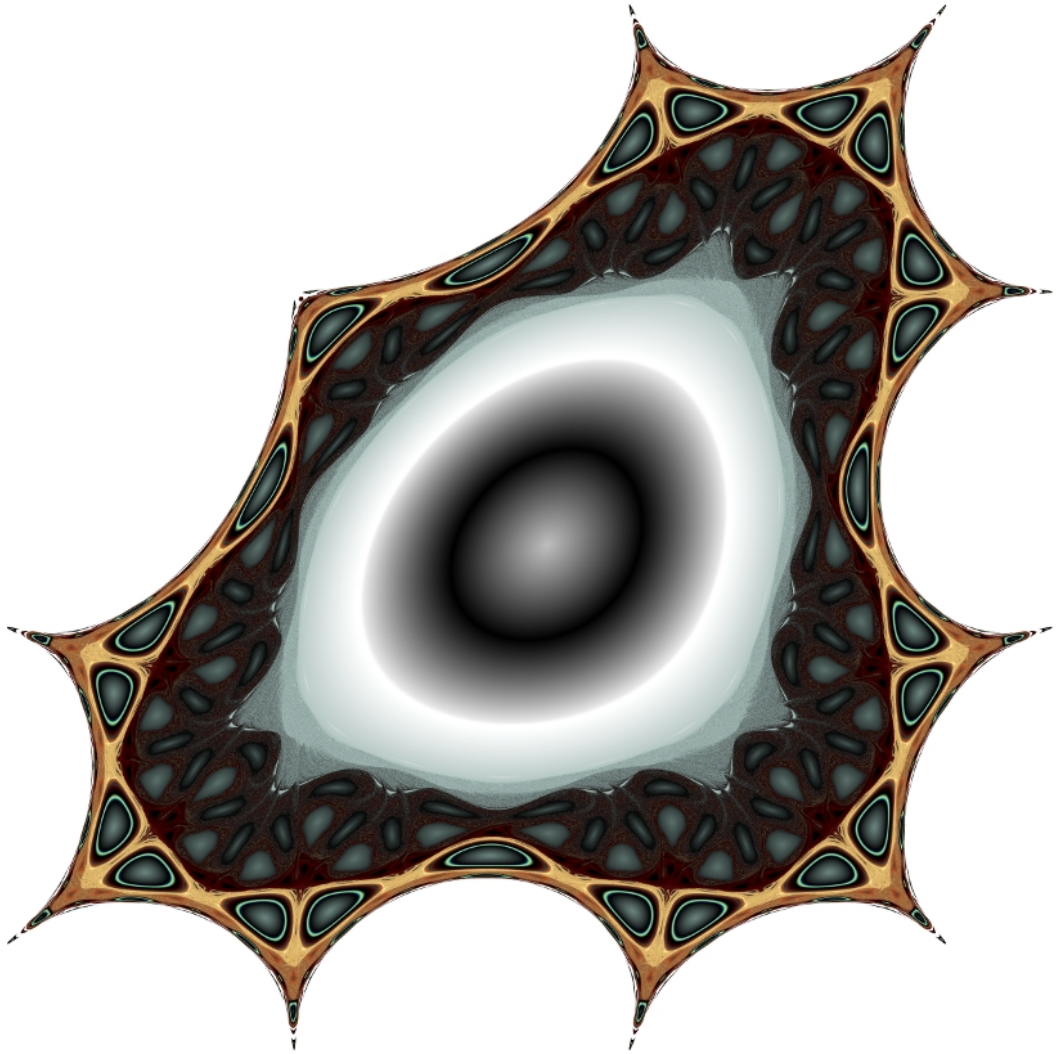


Table of Contents

1 Introduction

I am not a technical author so I apologise in advance if you find this manual confusing, I've endeavoured to make Saturn and Titan easy to use so you may find it easier to just use the software and dis-regard this manual.

Saturn and its sister program Titan are programs for generating fractal images. Saturn is used for exploring fractals and Titan is used for expanding images saved from Saturn.

The names of these programs follows a theme of planets and their moons and are a culmination of several years work, they were preceded by Venus, Mars, Phobos, Terra and Luna.

Venus was a simple program supporting just one fractal type a simple quadratic:

$$z \leftarrow z^2 + z + c$$

Venus served its purpose of trying out the Gtk+ toolkit using C. Mars was then embarked on using Gtkmm and C++ which started with just a few fractal types and ended with over a hundred. Mars could only produce limited size images so Phobos was conceived to produce much larger images by expanding files saved from Mars. It was found that some images took a very long time to expand leading to new pair of programs Terra and Luna, Terra performed the expansion with the expansion data held in a database and Luna provided the user interface. Terra and Luna could be stopped and the computer shut down, when the computer was restarted Terra could continue from where it left off. So Terra and Luna could expand an image that took days or even weeks to complete.

As new features were added it became obvious that major infrastructure changes were required resulting in Saturn and Titan, replacement of Terra and Luna is pending. (this was true when version 1.0 was released, version 4.0 has now been released and there are currently no plans to replace Terra and Luna).

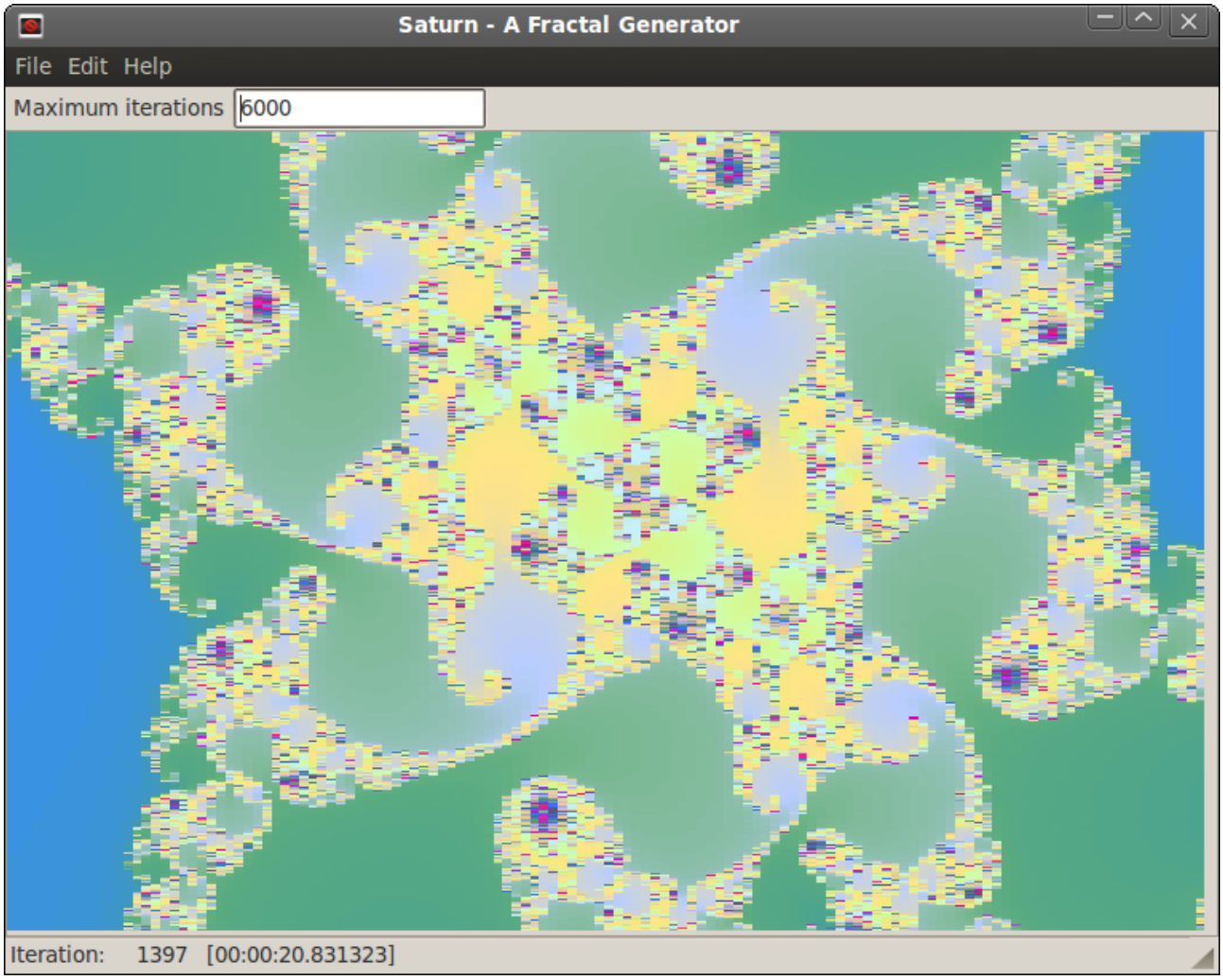
The version 1.1 update introduced function parameters allowing the variation of Pickover Popcorn (PP) fractals in their Mandelbrot, Julia and orbit plotted forms to be greatly increased.

Version 2.0 extends orbit plotting to all fractals except the Lyapunov fractal, the separate Pickover Popcorn orbit fractals have been removed as they can be produced using the PP Julia fractals. So the number of fractal types has actually gone down, however more functions are available for function parameters adding yet more variations to the PP fractals. The other major feature introduced with version 2.0 is the multi-threading of fractal calculation, the number of threads used is determined by type of processor used, the more processor cores the more threads there are, on a dual core processor 2 threads are used on a quad core processor 4 threads are used, Saturn and Titan automatically scale the number of threads used so the more processor cores you have the quicker the fractals are generated. If the CPU also has “hyperthreading” the number of threads is doubled so for a quad-core i7 processor the number of threads is 8.

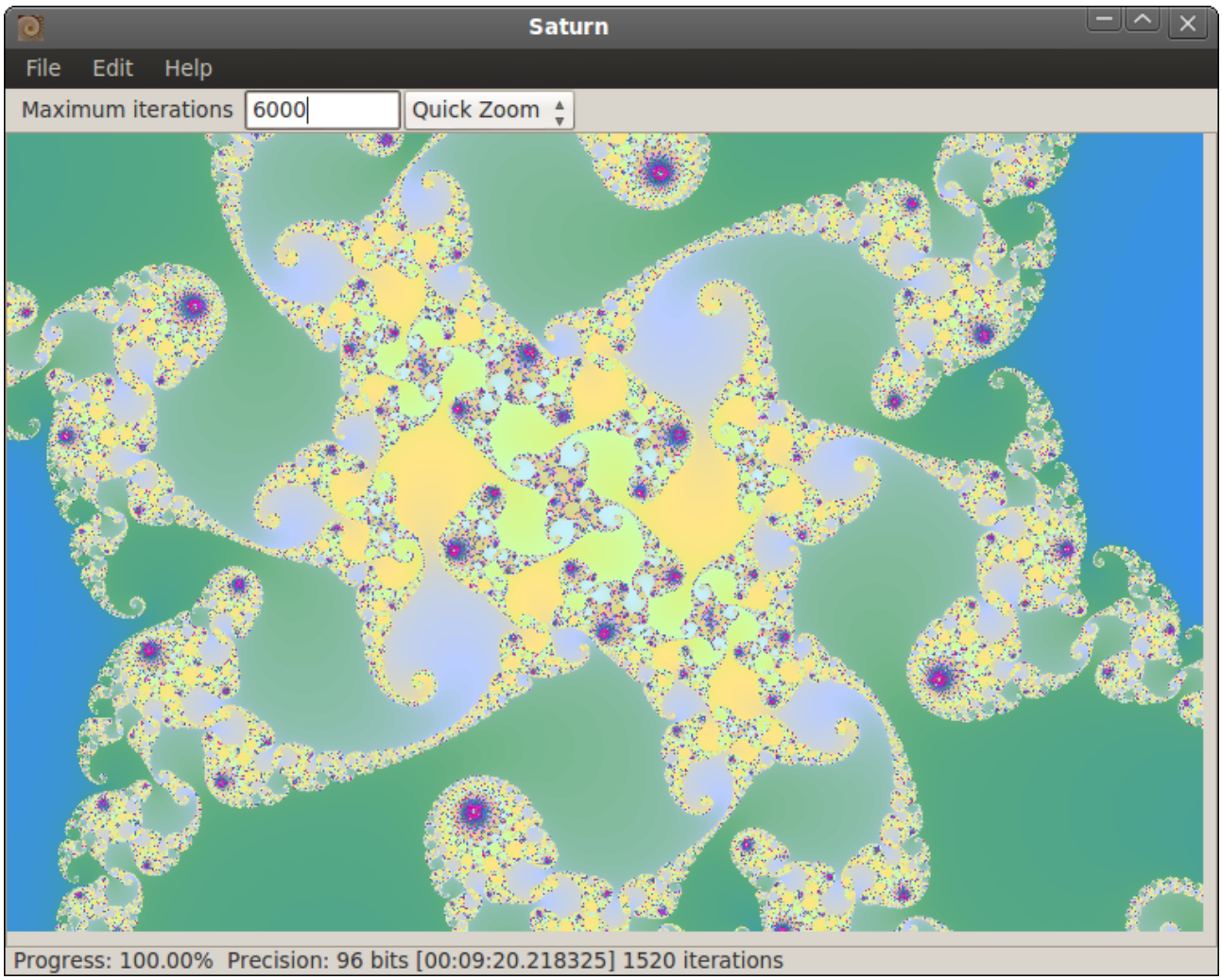
Version 3.0 introduced multi-precision mathematics for all fractals types except when orbit plotting is enabled. Prior to version 3.0 there was limit to how deep you could zoom into a fractal, when the limit is reached the pictures become blocky and there was little point in trying to zoom further in. Version 3.0 attempts to detect when there is insufficient precision to render an image correctly and increases the precision used so zooming in can continue until again there is insufficient precision and again precision is increased and so on. The fractal formulae have been revised such that the same formula type is used for both Mandelbrot and Julia algorithms and any complex parameter can be substituted with the location in the complex plane. The number of functions and complex

functions available to the function and complex function parameters has been increased and finally transforms are now defined using complex functions with a limit of 26 transform sets instead of 2.

Example of insufficient precision using version 2.1.2:



The same fractal with increased precision using version 4.0.0:



Version 4.0 changed the development toolkit from Gtkmm to Qt as there is better support for OS X. This release would have included an OS X version except that the iMac used for development acquired a GPU fault and can only boot into command line Linux. Version 4.0 has a revised user interface with greater control of zooming, a few more fractal types, and a new method of creating colour maps.

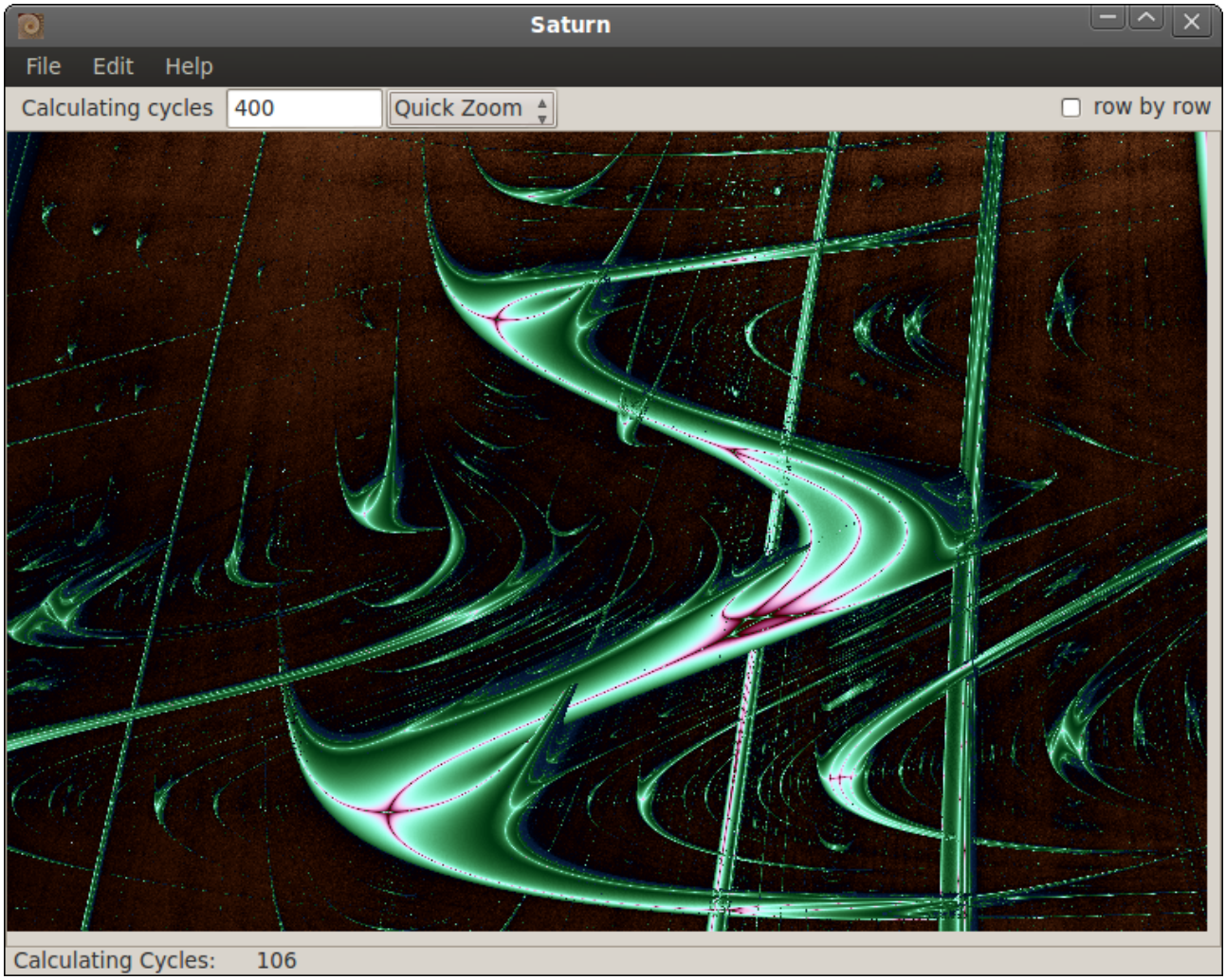
2 Definitions

seed file	a portable network graphics file (PNG) with all the necessary fractal parameters embedded in it so that it can be reproduced when it is opened by either Saturn or Titan. WARNING: if this file is processed in any way by a program over than Saturn or Titan the embedded data will be lost.
parameter file	a file containing all the parameters necessary to recreate a fractal. The data is in XML and the files have an spf extension (for Saturn Parameter File).
colour map	a colour map contains the colour definition use by the colouring method, there are three types “auto”, “manual” and component. Colour maps are embedded in seed files and included in parameter files, a maximum of 512 distinct colours can be defined in a colour map.
colour map file	Saturn can import colours from 3 different file types “.map” (FractInt), “.ugr” (Ultra Fractal) and “.scm” (Saturn Colour Map). Saturn can also export colour maps as either “.ugr” or “.scm” files.
z	the iterating complex variable
i	The imaginary unit number such that i^2 is equal to -1.
suffix .r	the real part of a complex variable e.g. z.r is the real part of z
suffix .i	the imaginary part of a complex variable e.g. z.i is the imaginary part of z.
subscript r	the real part of a complex variable, this may used in the formula section of the fractal tab (see fractal settings).
subscript i	the imaginary part of a complex variable this may used in the formula section of the fractal tab (see fractal settings).
Greek letters	The Greek lower case letters α, β,γ, δ, ε, ζ, η, θ these are the variables used as parameters in many fractal formulae in the formula section of the fractal tab, they are also used with .r and .i suffixes. In the summary tab of Titan however their names are used instead alpha, beta, gamma etc.
abs()	when applied to a complex number z the result is the magnitude, i.e. the square root of z.r*z.r + z.i*z.i. when applied to a real number it forces a negative value into a positive.
norm()	the norm of a complex number is z.r*z.r + z.i*z.i

3 Saturn

3.1 Main Window

Here is an example of the main window:



3.1.1 Menu

The menu consists of, File, Edit, View and Help.

File sub menu:

- Open – this can be used to open a seed or parameter file. The file contains the definition of the fractal and the colour map used.
- Save image … – save the currently displayed fractal image as a seed file.
- Save parameter … – save the definition of the currently displayed image as a parameter file.
- Import Colour Maps … – this option displays the import colour map window.
- Export Colour Maps ... – this option displays the colour map export window.
- Quit – exit Saturn.

Save image and save parameter both display a file save dialogue, the name used to save the file will have either .png or .spf appended if the extension is missing from the file name. The file saved by the image option is a seed file which can be used to load into Saturn or create a much larger image using Titan. The parameter file is a text file containing the parameters to recreate the fractal and they are defined in XML.

Parameters files are much smaller than seed files and are easier to share, any processing of seed file will result in the parameter data being lost resulting in ordinary png file. Parameter files can only be loaded into Saturn, if you you have an spf file which you want to create a large version of the fractal you must use Saturn to save a seed file which can then be used by Titan.

Edit sub menu:

- Fractal Settings … - this option will display the fractal settings window.
- Revert – revert to the starting complex plane, see the position tab of the fractal settings window.

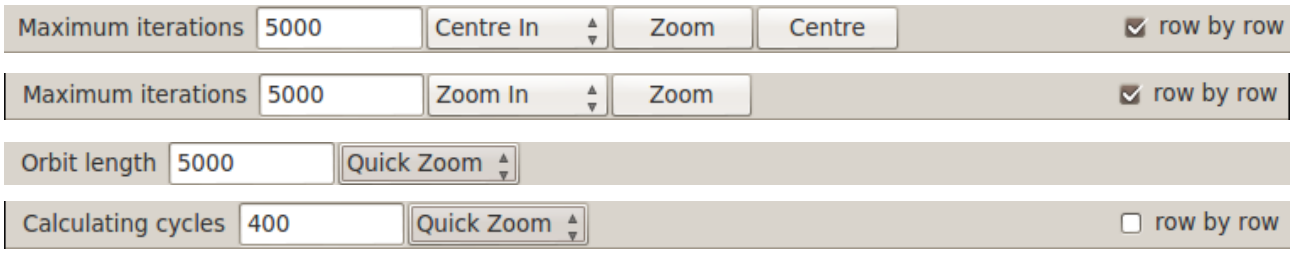
The help menu just provides an about option which displays version information.

3.1.2 Above the Fractal Picture

There are several items above the fractal picture and are context sensitive. The items are:

1. One of “Maximum iterations”, “Orbit Length” or “Calculating Cycles”.
2. An entry box showing the maximum number of iterations, orbit length or number of calculating cycles.
3. Zoom type selection: “Quick Zoom”, “Zoom In”, “Centre In”, “Centre Out” and “Zoom Off”.
4. Zoom button, displayed when the zoom type is “Zoom In”, “Centre In” or “Centre Out”.
5. Centre button, displayed when the zoom type is “Centre In” or “Centre Out”.
6. Row by row check box, not displayed for orbit plotted fractals or when multi-precision calculation is being used.

Examples:

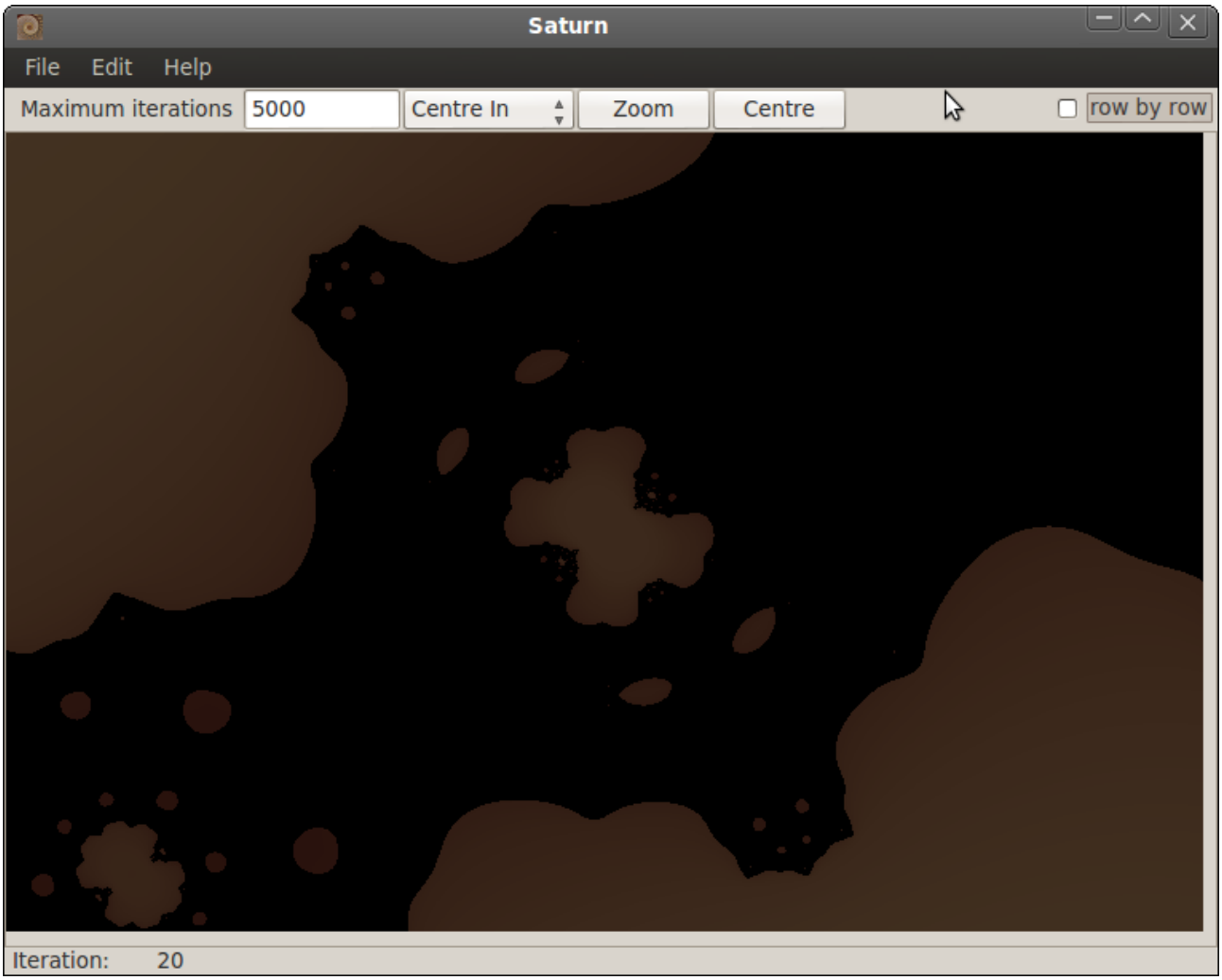


The value in the entry box can be changed at any time. If fractal calculation is in progress and the new value is greater than the old value then calculation of the image will continue, if the value is less than the old value calculation will restart. If fractal calculation is complete and the new value is greater then calculation will resume. If the fractal is orbit plotted any change will result in calculation restarting.

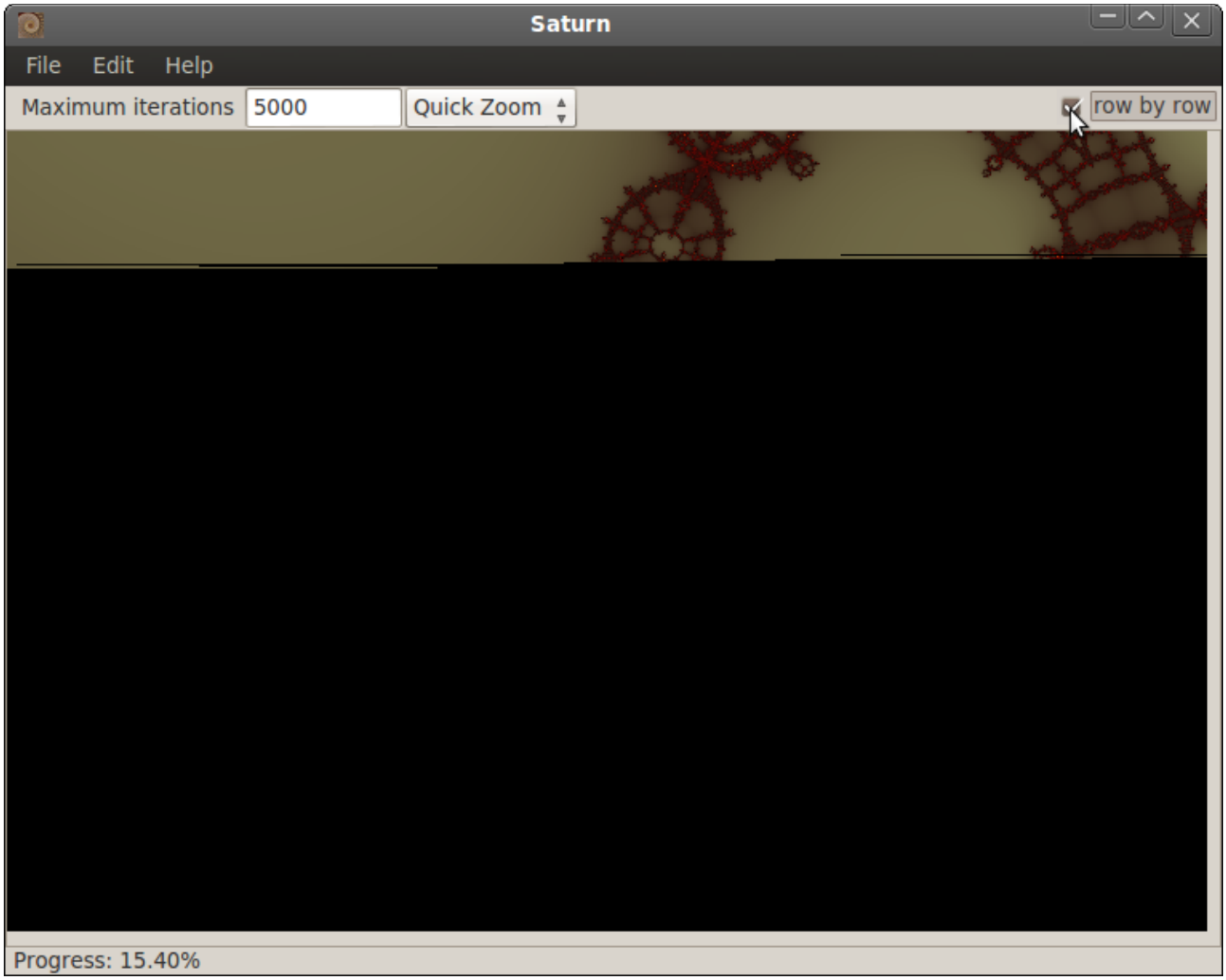
3.1.3 Image

The fractal image is drawn either “progressively” or “row-by-row”. When the image is updated progressively the entire picture is updated every 5 or 10 iterations or each calculating cycle for Lyapunov fractals. When the image is updated row by row the image is updated when one of the threads has completed a row, rows being calculated by other threads are also updated but the line may not be complete.

Progressive calculation:



Row by row calculation:

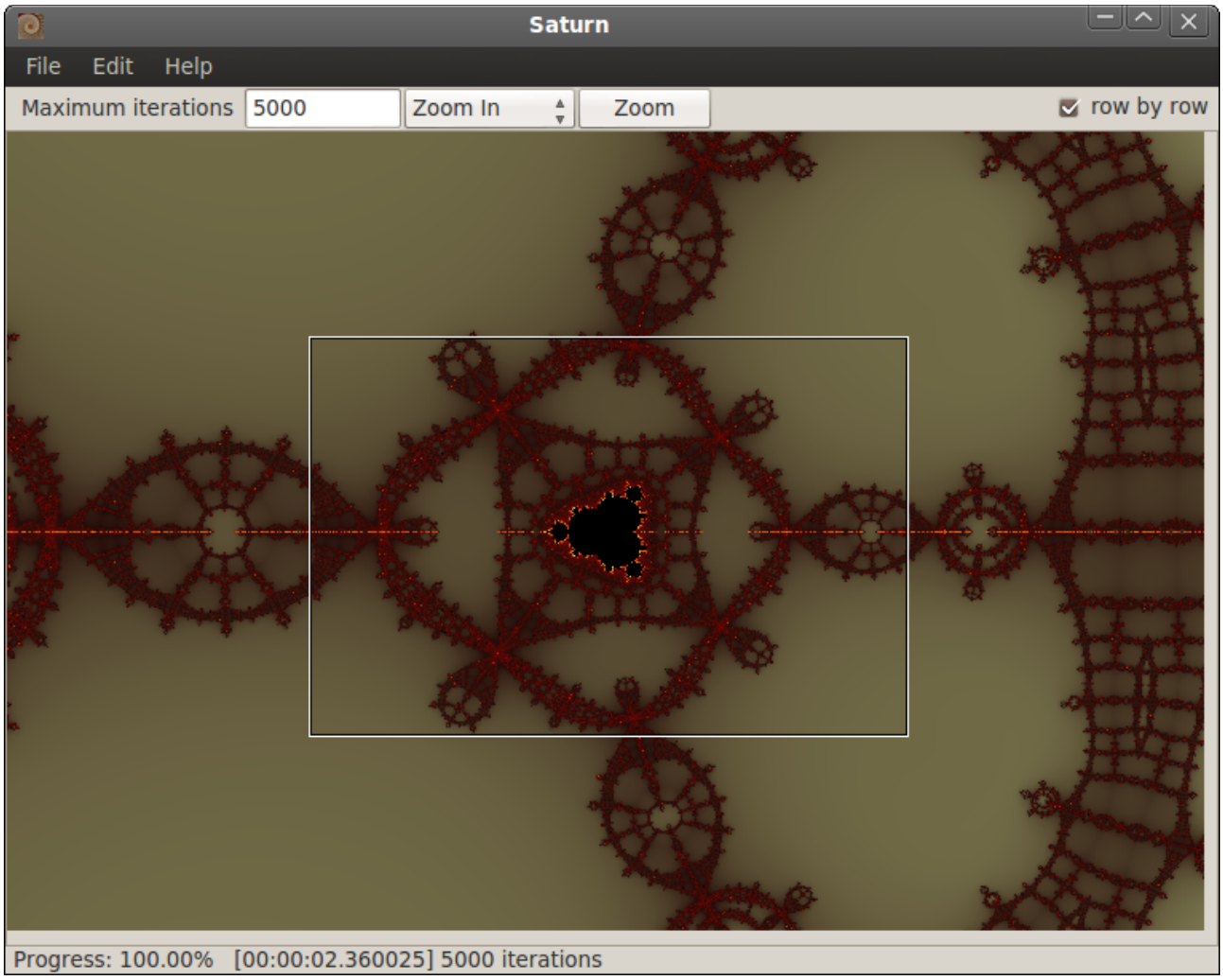


The area can also be used as a target for drag and drop of a seed or parameter file, the file will be opened and the fractal reproduced. If a png file which does not contain the embedded fractal data is dropped on the area a dialogue box will displayed informing the user that it does not contain the required fractal data. Files with extensions other than png or spf will be ignored.

Example seed files can be downloaded from http://element90.wordpress.com.

3.1.4 Zooming

Zoom rectangles are drawn as two rectangles, a white lined rectangle with a blacked lined rectangle within so that is can always be seen regardless of background.



3.1.4.1 Quick Zoom

This zoom type is the original zoom method, a zoom rectangle is drawn while the left mouse button is clicked an held down, the rectangle changes its dimensions as the mouse cursor is moved (it will disappear if it is too small). When the mouse button is released the picture is redrawn. The zoom type also allows a quick zoom out by clicking on the right mouse button, the image is recentred on the position of the cursor when the mouse button was clicked and the magnification is reduced by a factor of 4 (i.e. the dimensions of the complex plane are doubled).

3.1.4.2 Zoom In

When the left mouse button is clicked a zoom rectangle half the dimensions of the picture is displayed centred on the position of the the mouse cursor.

The rectangle can be grabbed when the cursor changes to an open hand inside the zoom rectangle, clicking the left mouse button grabs the rectangle which can then be moved while the mouse button is held down. While the rectangle is grabbed the mouse cursor is a closed hand. The rectangle is released when the the mouse button is released.

The size of the zoom rectangle can be changed “grabbing” one of the sides, when the cursor is near a side a double headed cursor will be displayed clicking the left mouse button and holding it down the size of rectangle can be altered by moving the mouse. Releasing the left mouse button releases control of the zoom rectangle's size.

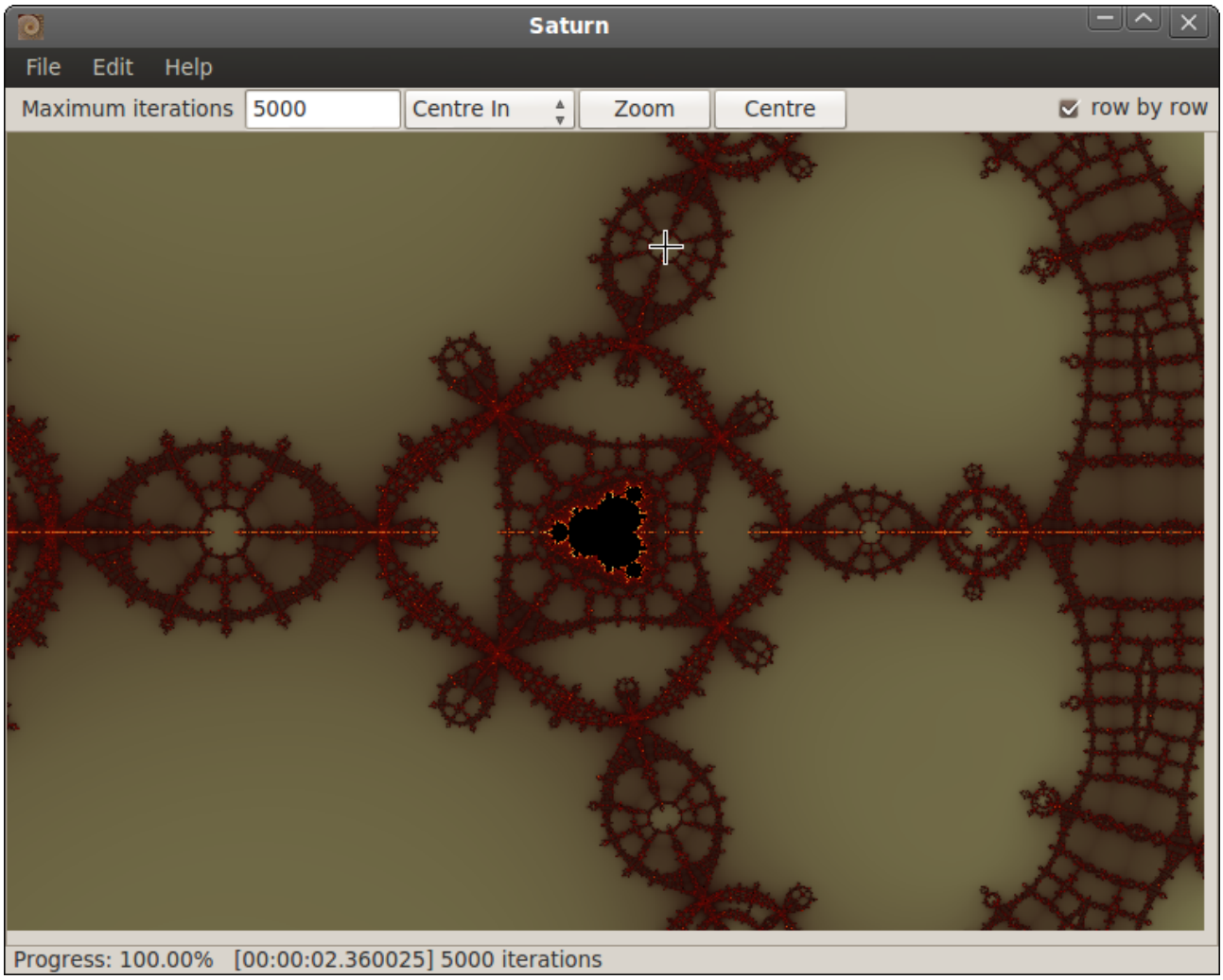
Calculation of the new zoomed image will only start when the zoom button is pressed.

The zoom rectangle can be dismissed by clicking the right mouse button or by changing the zoom type.

3.1.4.3 Centre In and Centre Out

These zoom modes are for recentring the image and for recentring the image and zooming in or out. The zoom factor is always 4 i.e. doubling or halving the complex plane dimensions.

Clicking the left mouse button displays a cross.



The cross can be grabbed when the mouse cursor changes to an open hand (over the cross), clicking the left button grabs the cross and it can be moved while the button is held down, the cursor changes to a closed hand while it's grabbed, the cross is released by releasing the mouse button and the cursor changes.

Calculation of the new zoomed image is started when either the zoom button is pressed. To just recentre the image the centre button is pressed.

The centring cross can be dismissed by pressing the right mouse button or changing the zoom type.

3.1.5 Status Line

The status line is used to show how the generation of a fractal has progressed and consists of several items:

1. Percentage progress, displayed for row by row calculation and orbit plotting.
2. Settling Cycle, displayed for the settling phase of the Lyapunov fractal. Not displayed when row by row is enabled.
3. Calculating Cycle, displayed for the calculating phase of the Lyapunov fractal. Not displayed when row by row is enabled.
4. Precision, displayed only when multi-precision calculation is being used.
5. Duration enclosed in square brackets and is displayed when calculation is complete.
6. Maximum number of iterations reached, displayed when row by row calculation has completed. Not displayed for Lyapunov fractals.

Examples of the status line can be seen below the fractal picture in examples of the main window in this section.

3.2 Fractal Settings

The fractal settings window consists of two parts, on the left there is a list and on the right there are tabs, the list contains the names of fractal types for all tabs except for the Colour and Colour Maps tabs when it contains the names of colour maps.

When it is a fractal type list the cursor will be displayed on the current fractal type i.e. the type currently used to generate the image in the main window. Changing the cursor with change the current fractal being displayed. **Note: as the changes for version 4.1.0 are minor the fractals list in the exampes does not include fractals renamed are added for version 4.1.0.**

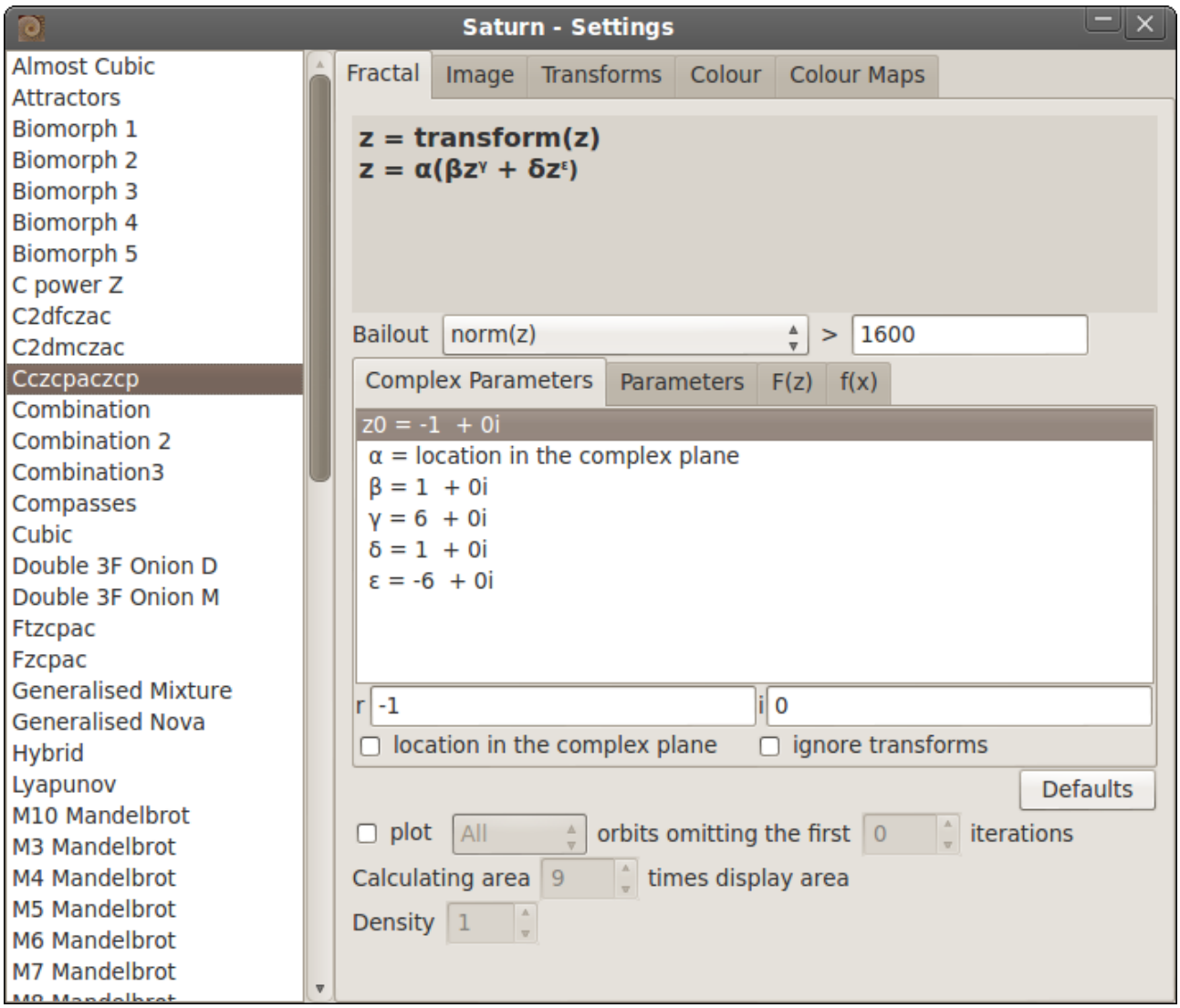
When it is a colour map list the cursor will be on the current colour map to be edited, or the colour map used by the fractals colour methods. The cursor is removed if the colour type is a single colour.

3.2.1 Fractal Tab

There are two versions of the fractal tab depending on the fractal type, these versions are for:

1. Escape time fractals
2. The Lyapunov fractal.

3.2.2 Escape Time

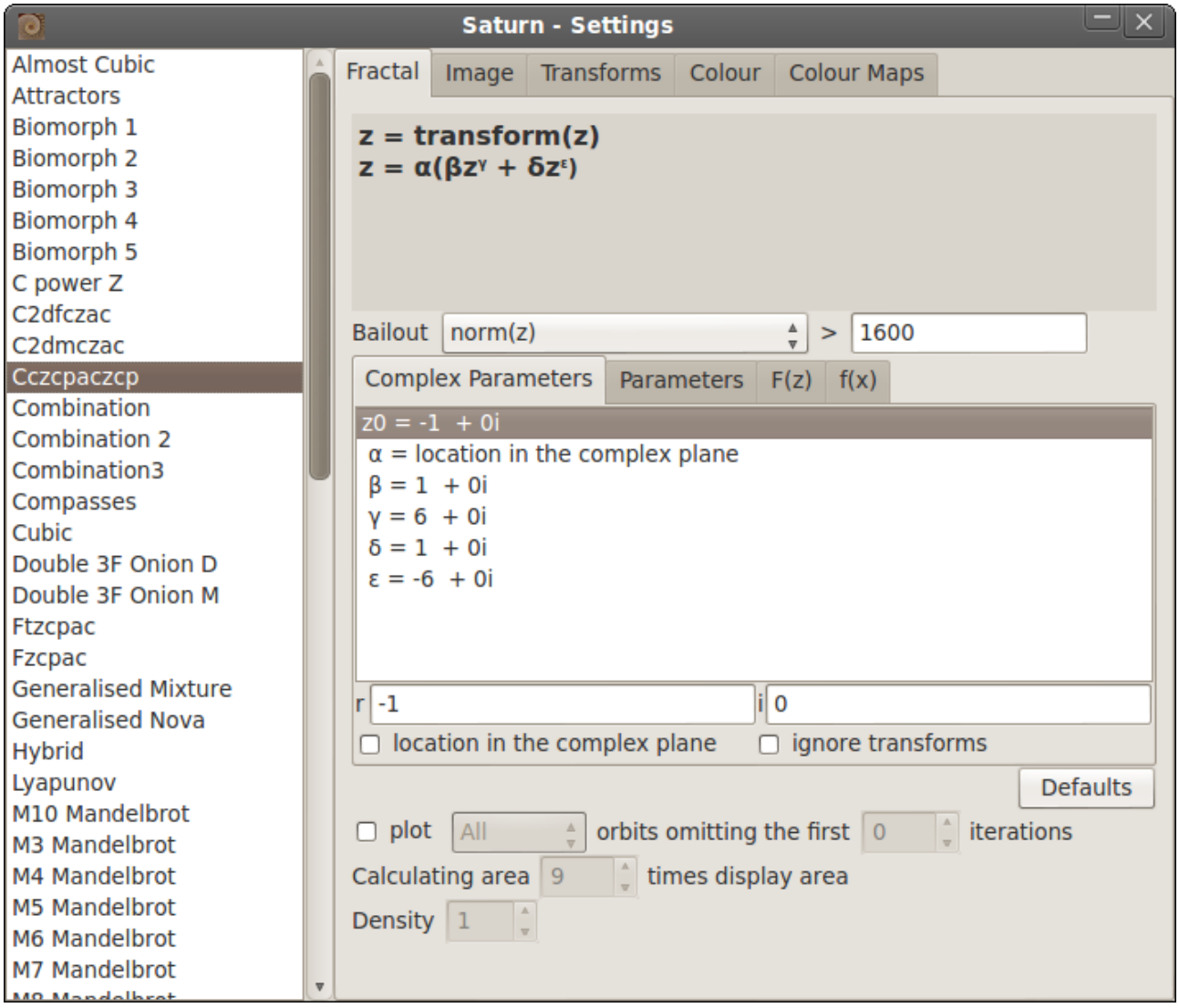


The formula

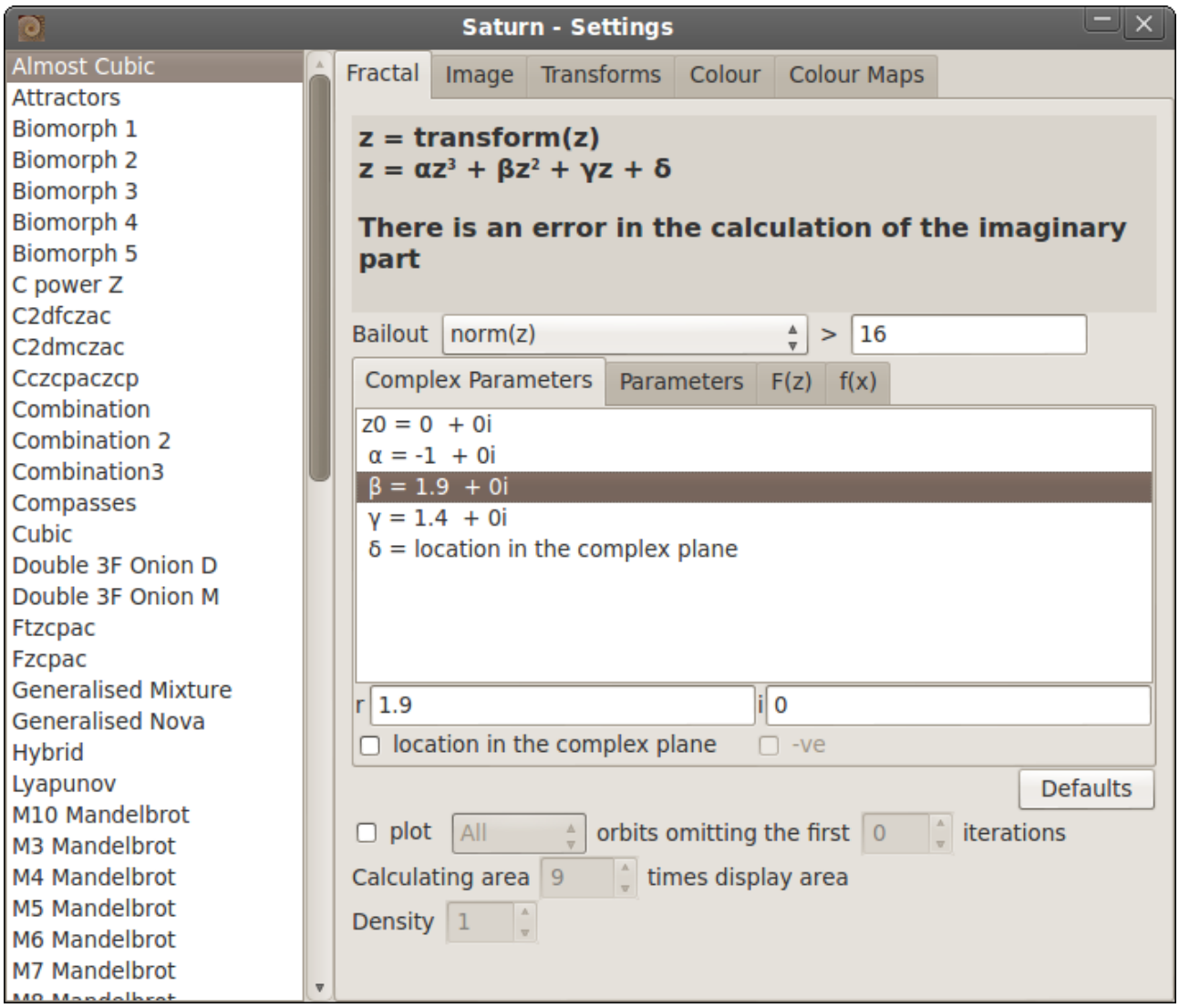
The formula used to generate the fractal, showing the parameters used. The are four classes of parameter, complex number parameters identified using z0 and Greek letters, number parameters identified using capital letters, complex functions identified by Fn and functions identified by fn. (n is a digit in the range 0 to 7).

Parameter Tabs – Complex Parameters

The complex parameters tab always is always populated. The other tabs will display “no parameters defined”, “no complex function parameters defined” and “no function parameters defined” where no parameters are defined for that particular class.



The z0 parameter is the initial value used for calculating the fractal and has a fixed value, or it can be set to the location in the complex plane. If transforms are defined for complex plane the untransformed location can be used by setting the “ignore transforms” option.

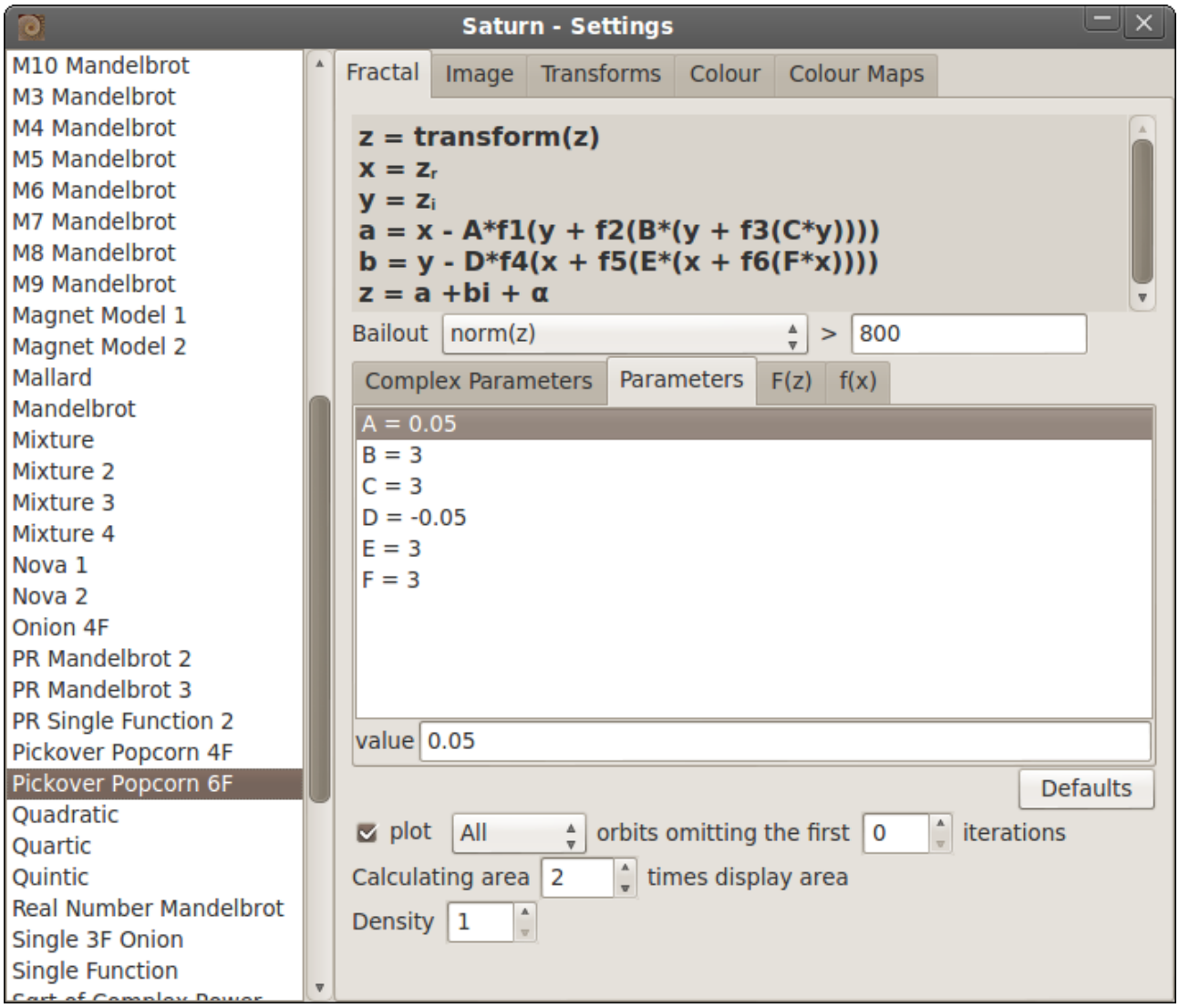


Parameters other than z0 can be set to a fixed value or can be set to the location in the complex plane (transformed), or the negative location in the complex plane by setting the “-ve” option. The “-ve” option is only enabled when “location in the complex plane” is set. The values for r and i are disabled when location in the complex plane is set.

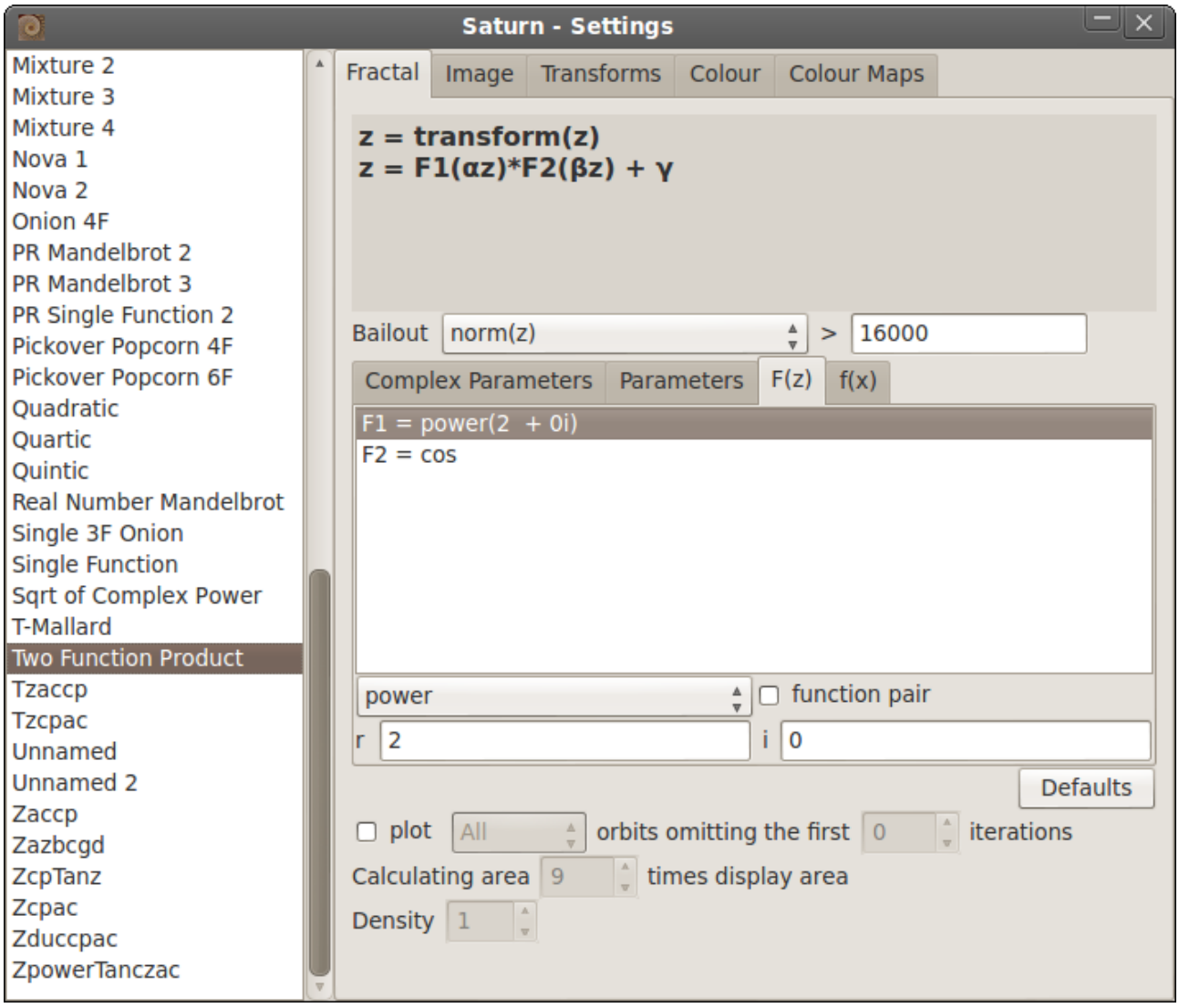
If none of the Greek letter parameters are set to the location or negative location in the complex plane the Julia algorithm will be used for calculating the fractal. The value of z0 is also forced to be the location in the complex plane and ignore transforms option is cleared and disabled. If any parameter is set to the location or negative location in the omplex plane the Mandelbrot algorithm is used.

For fractals that use the Mandelbrot algorithm care should be taken when setting a fixed value so that the result of the evaluated formula is not infinity (division by zero) or undefined (log of zero), undefined results in “not a number”, infinity or not a number will be treated as though the bailout condition has been met. ***Note: for “No Bailout” the condition is not evaluated so calculation will continue regardless and will be VERY SLOW once z becames “not a number”.***

Parameter Tabs – Parameters

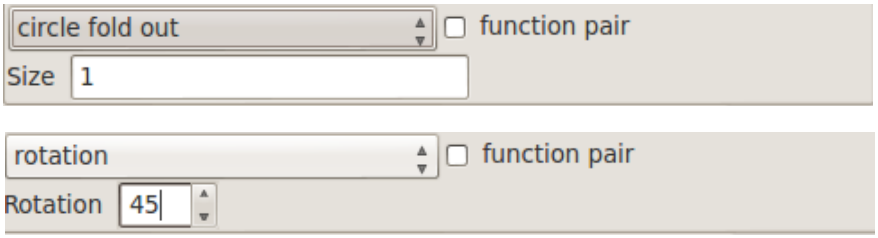


Parameter Tabs – F(z)

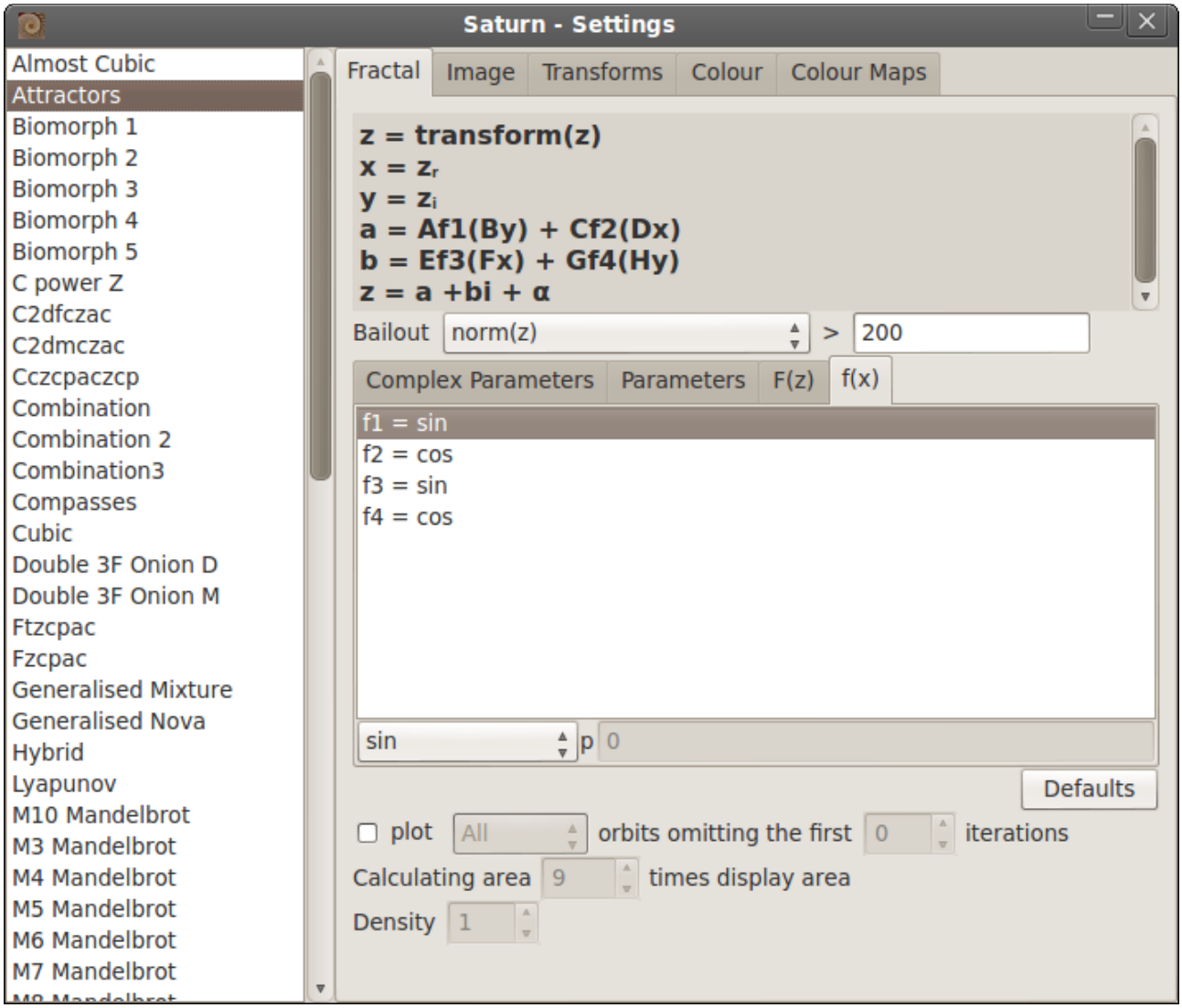


Complex functions can be applied to complex numbers as a whole or as function pairs applied to the complex number components. Functions appear in the list by name with any parameters in brackets, function pairs are separated by a comma.

Some functions have parameters, r, and i, Size or Rotation, when there are no parameters r and i will be displayed and will be disabled.

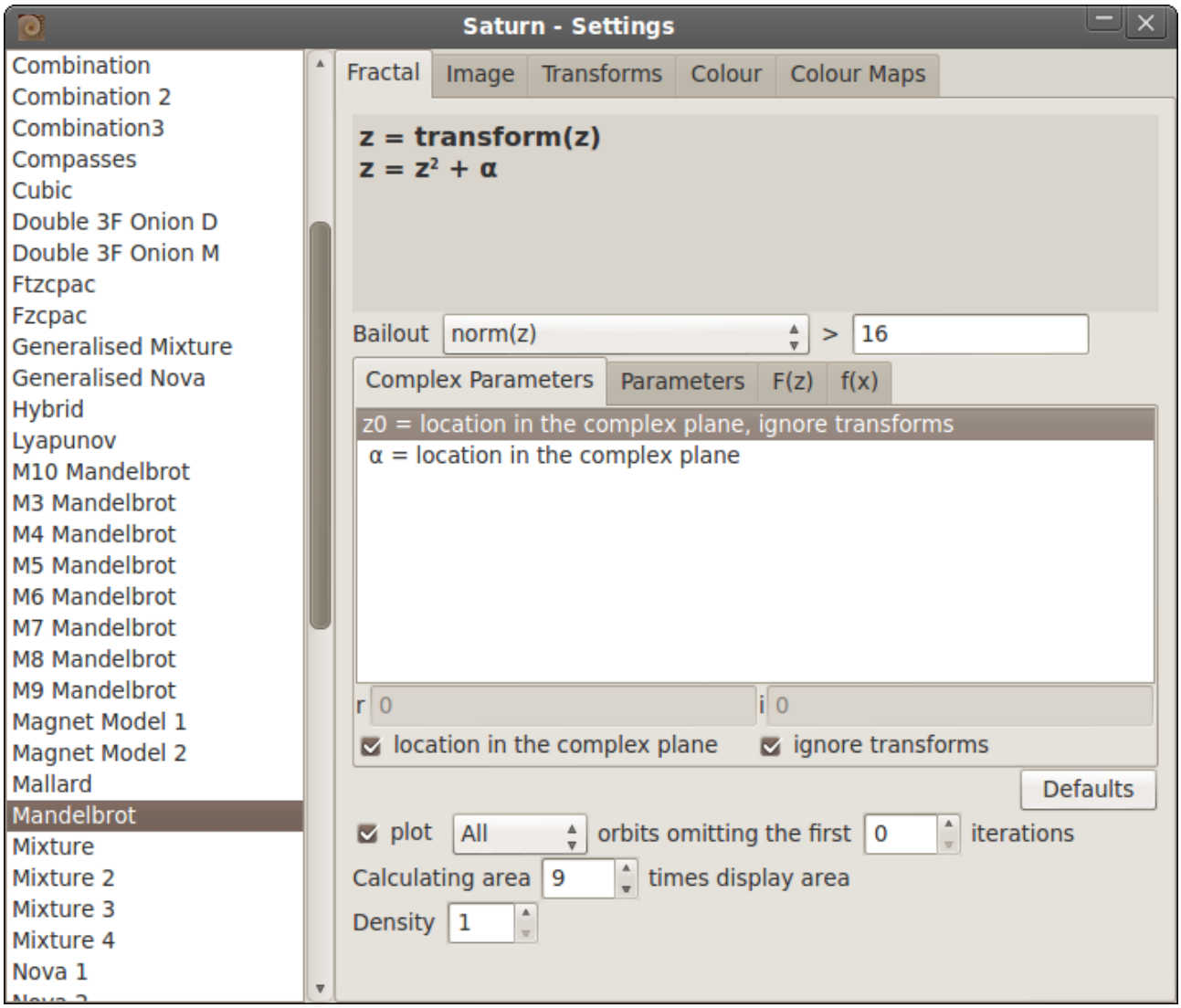


Parameters – f(x)



Some parameters have a parameter which is set by setting the value of p.

Plot orbits



The orbits of the fractal are plotted when this option is used. There are three plot variations, all orbits, escaped orbits and captive orbits, a number of iterations at the start of each of the orbits can be omitted from being plotted.

The nature of orbit plotted fractals means that detail is lost as you zoom into the picture as orbits that would form part of the picture are never calculated. To reduce the effect, the area used for calculating the fractal can be increased up to a maximum of 1000 times the display area. The larger the calculating area the greater the time required to generate the fractal picture.

Increasing the density value increases the number of orbits calculated for the calculating area, e.g. fo a density 100 times more orbits are plotted than when the density is 1.

The orbit plot option is disabled if there are complex plane transformations defined for the fractal.

Note: for single orbit fractals there are no options below the “Defaults” button.

Bailout Condition

The bailout condition used for the fractal. There are two sets of bailout conditions depending on whether the bailout is divergent or convergent. For divergent bailout the the value of z is used to determine whether the bailout condition has been reached.

The divergent bailout conditions are as follows:

- No bailout. See Note 1
- abs(z)
- norm(z)
- imag(z)
- imag(z) squared
- real(z)
- real(z) squared
- abs(real(z)*imag(z))
- abs(real(z)) + abs(imag(z))
- real(z) + imag(z)
- abs(real(z) + imag(z))
- max(real(z), imag(z)) squared. See Note 2
- min(real(z), imag(z)) squared. See Note 2
- abs(real(z) – imag(z))

So, for example if the bailout condition is norm(z) and the limit is 16 the following is evaluated:

norm(z) > 16

if the result is true then calculation for the location has completed.

For convergent bailout the condition uses the difference between the value after the formula has been evaluated and the value before the formula has been evaluated, this value is shown as Δz.

- No bailout. See Note 1
- abs(Δz)
- norm(Δz)
- imag(Δz)
- imag(Δz) squared
- real(Δz)
- real(Δz) squared
- abs(real(Δz)*imag(Δz))
- abs(real(Δz)) + abs(imag(Δz))
- real(Δz) + imag(Δz)
- abs(real(Δz) + imag(Δz))
- max(real(Δz), imag(Δz)) squared. See Note 2
- min(real(Δz), imag(Δz)) squared. See Note 2
- abs(real(Δz) – imag(Δz))

So, for example if the bailout condition is real(Δz) and the limit is 0.01 the following is evaluated:

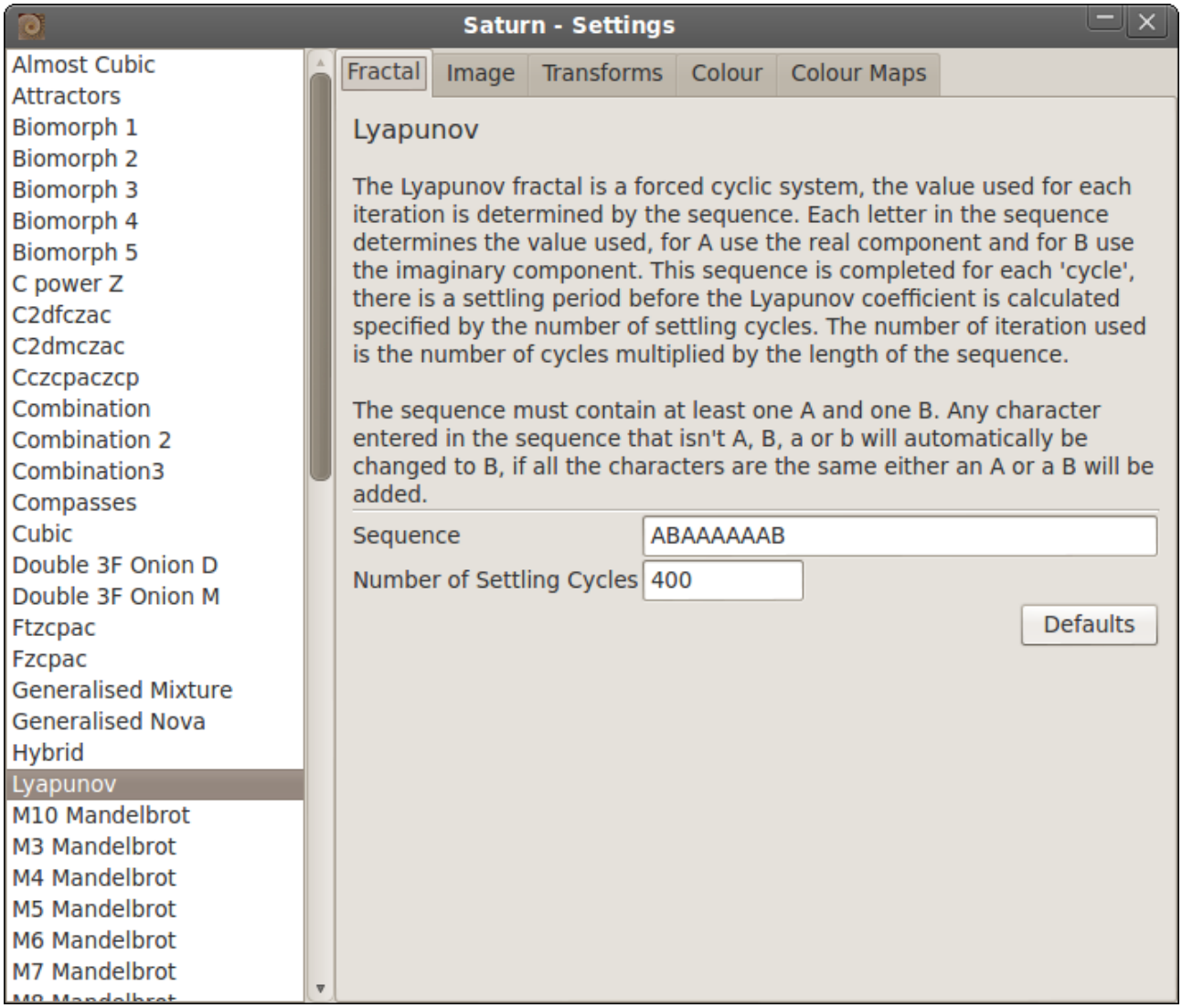
real(Δz) < 0.01

if the result is true then calculation for the location has completed.

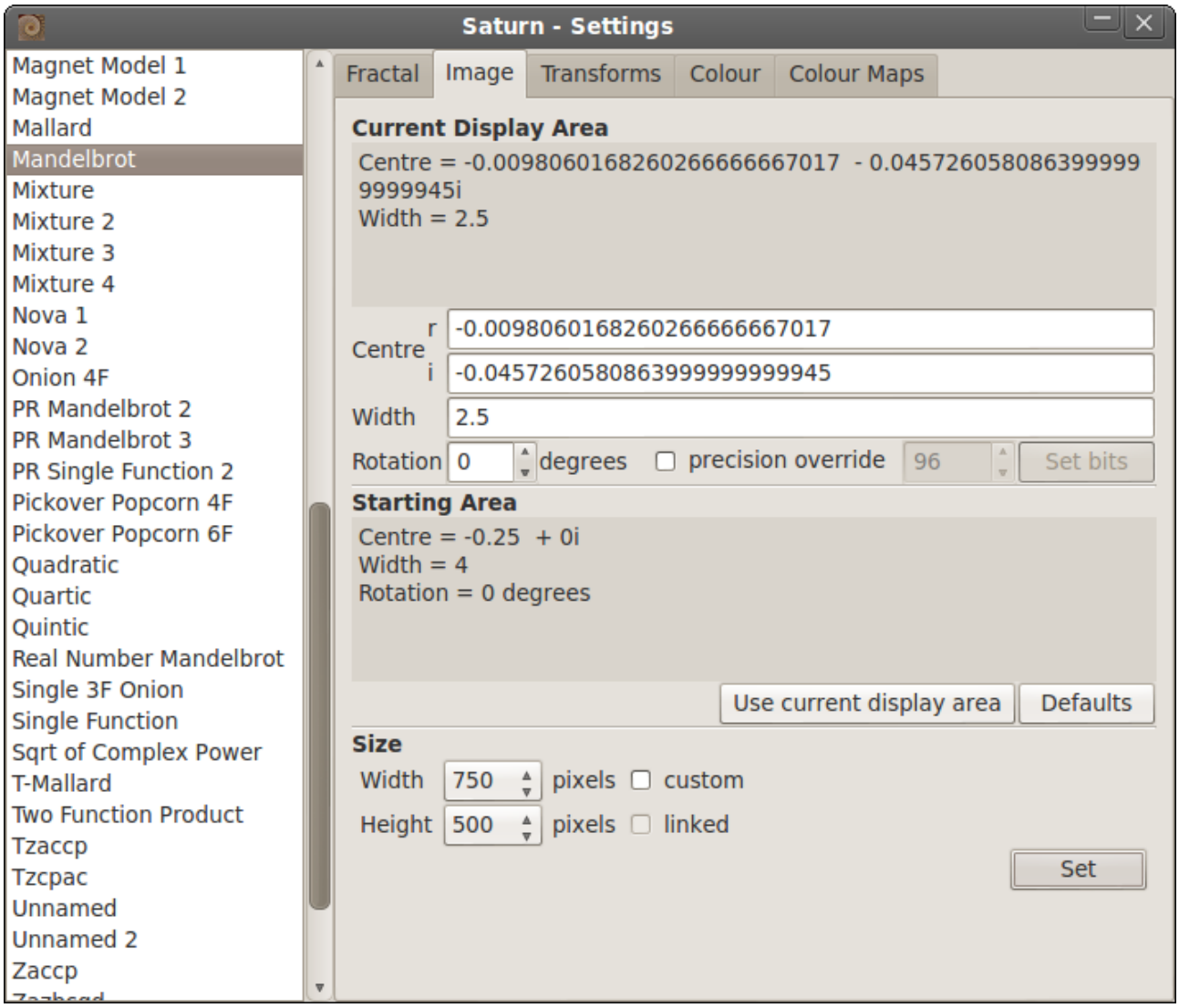
Note 1: both sets of bailout conditions include "No bailout" which means calculation continues until the number of iterations has completed. Care should exercised when using this option for fractals with divergent bailout as the value of z may reach infinity (inf) and in turn become "not-a-number" (nan), if this occurs calculation of each iteration will become VERY SLOW. An other symptom is a single block of colour displayed for a colour method such as Magnitude for all options except minimum, iteration @ minimum and iteration @ maximum.

Note 2: the component parts are squared BEFORE determining maximum or minimum.

3.2.2.1 Lyapunov



3.2.3 Image Tab



Current Display Area

There is a read only text area which displays the location of the fractal in terms of the centre of the image and the width. All available digits are displayed, the number of digits will increase as the precision used to calculate the fractal is increased. The text area will include a vertical scroll bar when the text becomes too large for the display area.

Below the text area are the entry boxes for setting the location of the fractal manually. The mechanism for determining the required precision breaks down for some fractals, the symptoms are blocky or fuzzy images, the precision can bet set manually using the precision override option, the values are in intervals of 16 bits starting at 96 (80 on Windows).

The data in the the Current Display Area are automatically updated when zooming and centring the image in the main window.

Starting Area

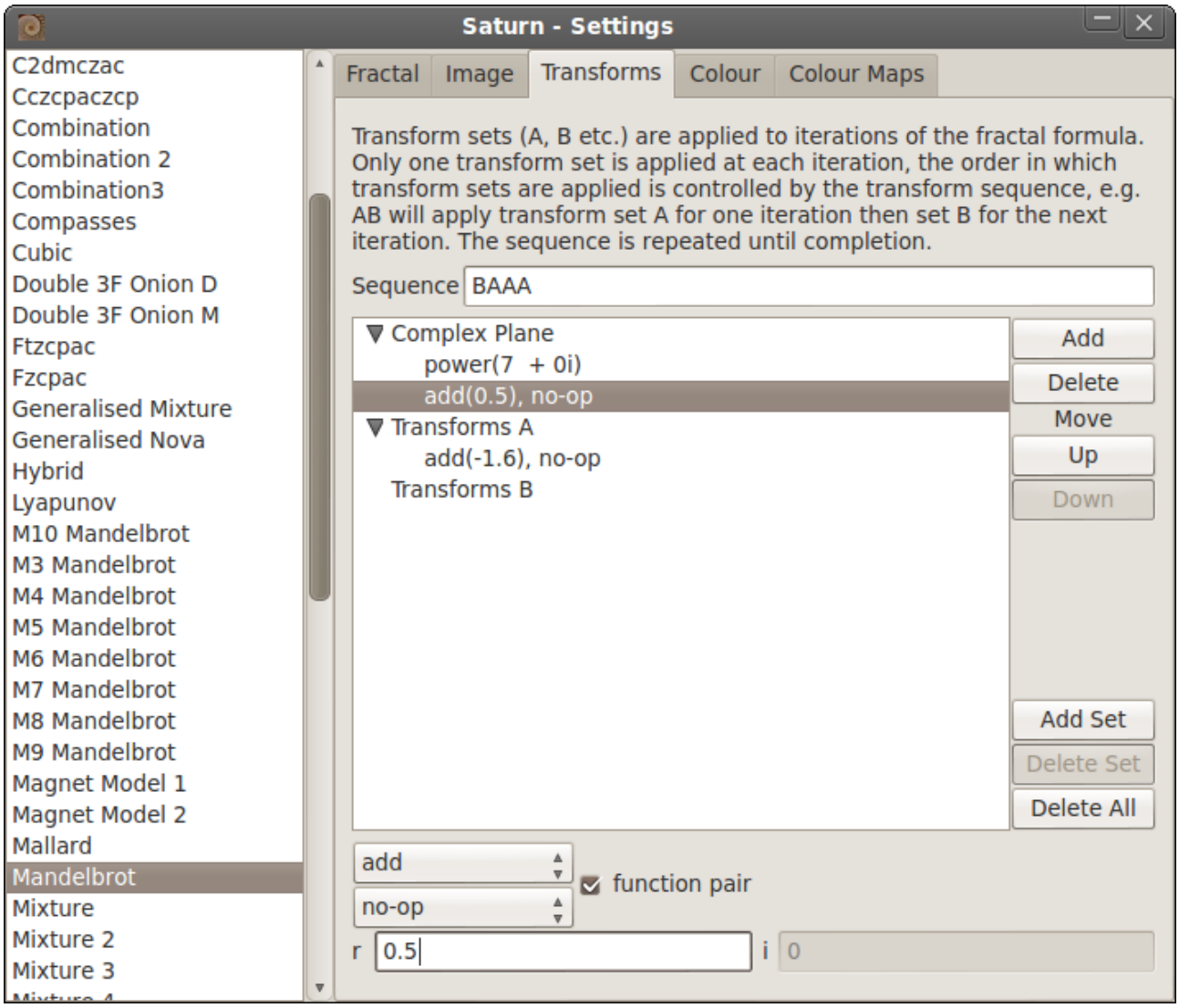
The starting area displays the initial default location and width of the fractal. The current location and width are set to the starting area values when the revert option in the edit menu is used. The location to revert to can be changed by setting it to the current display location so that whenever the revert option is used the fractal will be calculated for that “starting position”, the starting position can be changed back to the system default at any time.

Size

The size defines the dimensions of the picture in pixels, there are a number of pre-defined widths and heights. In addition custom values can be used, setting the custom option changs the combboxes to spin boxes, the values can be changed independently or linked so that the aspect ratio is maintained.

The new size is only used when the set button is pressed.

3.2.4 Transforms Tab



Note: this tab is not applicable for Lyapunov fractals so the all the buttons, the sequence and the transform definition are disabled. If orbit plotting is enabled transforms can not be added to complex plane.

Sequence

The required transform sequence is entered in the sequence entry box. A string of any length can be entered, only letters for defined transform sets will be accepted all letters are made uppercase regardless of the case they were entered.

Transform Tree

The transform tree displays the transforms to be applied to the complex plane and for up to 26 defined transform sets. The transforms assigned to a set will be displayed indented below the transform set's name.

Transforms are defined in the same way as complex functions. No attempt is made to prevent ineffective transforms e.g. transforms that cancel each other out or duplicate transforms that have no

affect such as abs, no-op followed by abs, no-op. The use of ineffective transforms only serves to slow down the calculation of the fractal.

The order of the transforms in a set can be changed by moving them up or down in the set, the order of transform sets can not be changed but the order in which they are applied can be changed by altering the transform sequence.

The definition of transforms were changed as of version 3.0 of Saturn and Titan. Conversion of old transform types to the current types is detailed in the appendix.

Transforms can be deleted individually, by set or all the transforms can be deleted in one go. When a transform set is deleted any following sets will be renamed e.g. deleting C would result in D becoming C, E becoming D etc. (if defined), all occurances of the deleted set letter will be removed from the sequence and any following set letters will be replaced.

The buttons for adding, deleting and moving transforms are enabled and disabled according to context, for example a transform at the beginning of a set can not be moved up.

3.2.5 Colour Tab

The appearance of the colour tab is dependent on fractal type. The list on the left hand side displays the names of colour maps instead of fractal types.

3.2.5.1 Colour – Common Settings

The colour tabs have some common settings relating to the use of colour maps.

Colour offsets

With the exception of fixed colour all colouring methods have colour offset spin buttons. There are three offsets one for each component, the values can be varied independently or in lock-step (i.e. if one is changed the other two are also changed). The colour offsets affect the starting positions in the map for each colour component, the effective range of colours is changed and the new set of colours is displayed in one of the colour map rectangles at the bottom of the tab.

Colour Component Order

Each colour in a colour map consists of three values assigned to red, green and blue (RGB). The values can be assigned to different colours for example GRB uses the red value for green and the green value for red. The are six different component order combinations which effectively increases the number of colour maps available. This may not result in six different sets of colours, for example greyscale colour maps with remain the same regardless of the colour component order,

Colour Map Rectangles

At the bottom of the colour tab there are either one (for Orbit Plot colouring) or two colour map rectangles, the upper colour map is for outer or positive (Lyapunov) colouring.

The colour offsets and component order affect how colours are extracted from the map, the colour map rectangle is 512 pixels wide and each pixel column represents a colour in the map starting at index 0 and finishing at index 511. The colours in the colour map are adjusted according to the colour offsets and component map and the resulting sequence of colours is displayed in the colour map rectangle.

Scale

The frequency in which the colours change can be altered by changing the scale value.

Smoothing

Most colour methods convert a number into a colour, the number is used to look up a colour from a colour map where 512 individual colours are defined. The number does not map precisely on to a colour in the colour map, instead it will fall between one colour and the next, if the smooth option is enabled the colour used is a blend of the colour below and the colour above the number, if smooth is not enabled the lower colour only is used. When blending colours more weight is given to the closer colour, for example if the first colour is yellow and the second is red the result will be an orange, if the number is closer to yellow then a yellower shade will be used.

For iteration colouring there is a direct match to individual colours, enabling smooth alters the number used to find the colours such that it falls between colours in the colour map and a blended colour is used. The alteration to iteration is detailed in the section on iteration colour.

Absolute log

When this option is enabled the absolute log of the value is used to determine the index of the colour in the colour map. If the logarithm of the value is negative the sign is removed. If the logarithm of the value is complex then magnitude of that complex number is used.

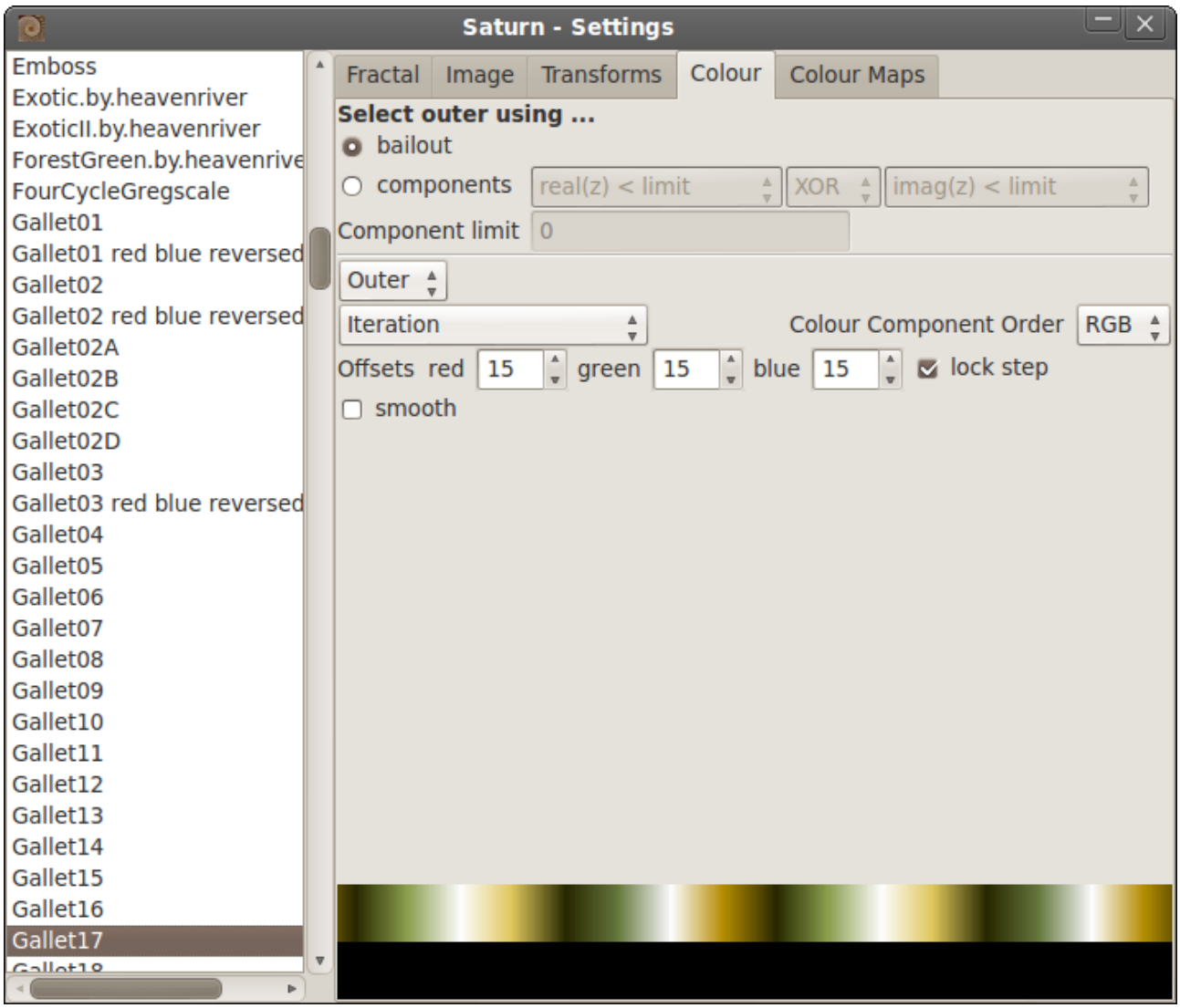
Result Statistics

For many of the colour methods a set of statistics is collected, the colours used are dependent of the statistic used. The statistics are:

1. Minimum
2. Iteration @ Minimum
3. Maximum.
4. Iteration @ Maximum.
5. Range.
6. Average.
7. Variance.
8. Standard Deviation.
9. Coefficient of Variation.
10. Fractal Dimension.
11. Exponential Sum.
12. Exponential Inverse Change Sum.

A different set of statistics is used for Gaussian Integer.

3.2.5.2 Colour – Escape Time Fractals

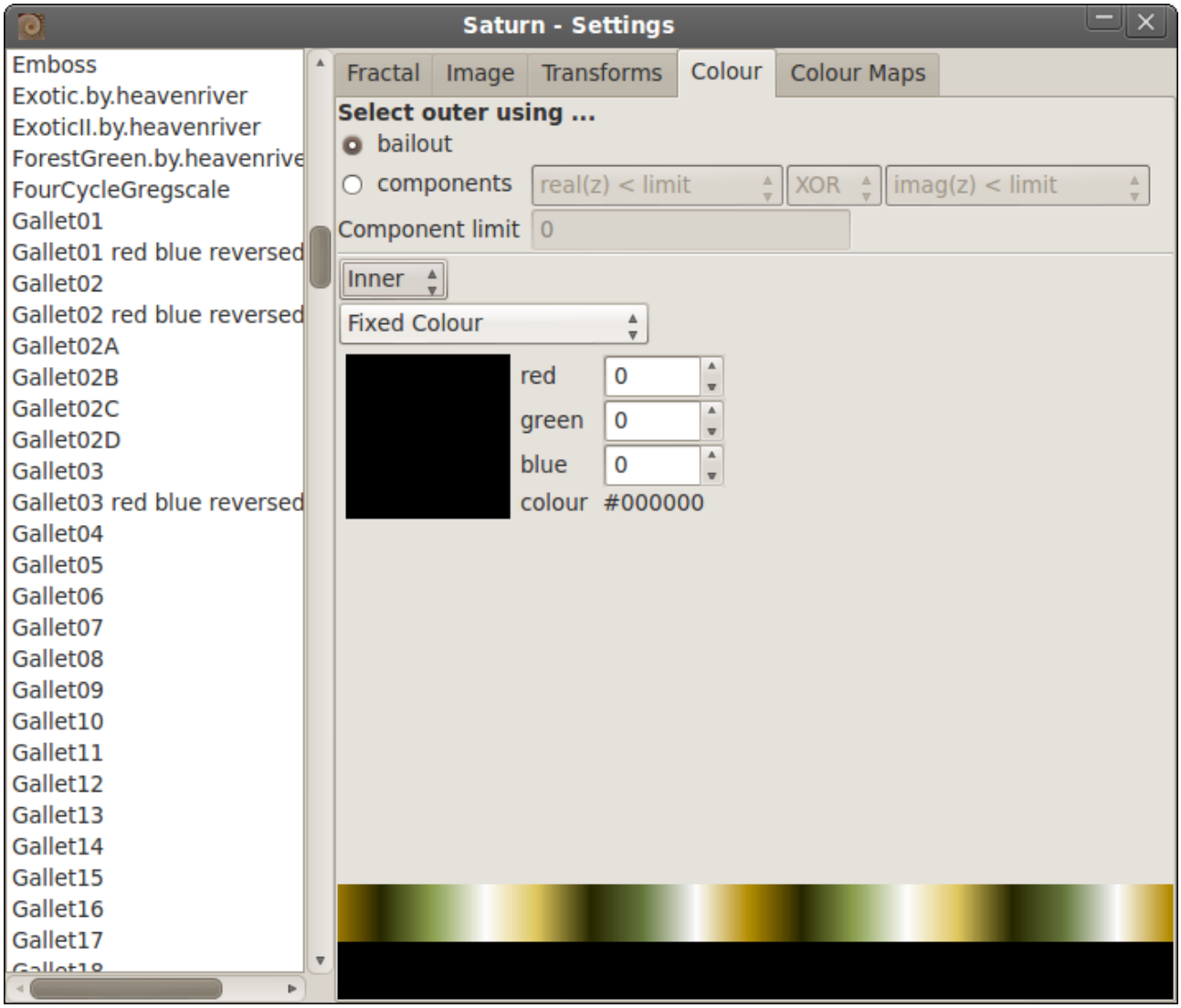


In most cases colour selection is dependent on bailout, for “biomorph” colour selection additional options based on component values of the complex number calculated at each iteration are used.

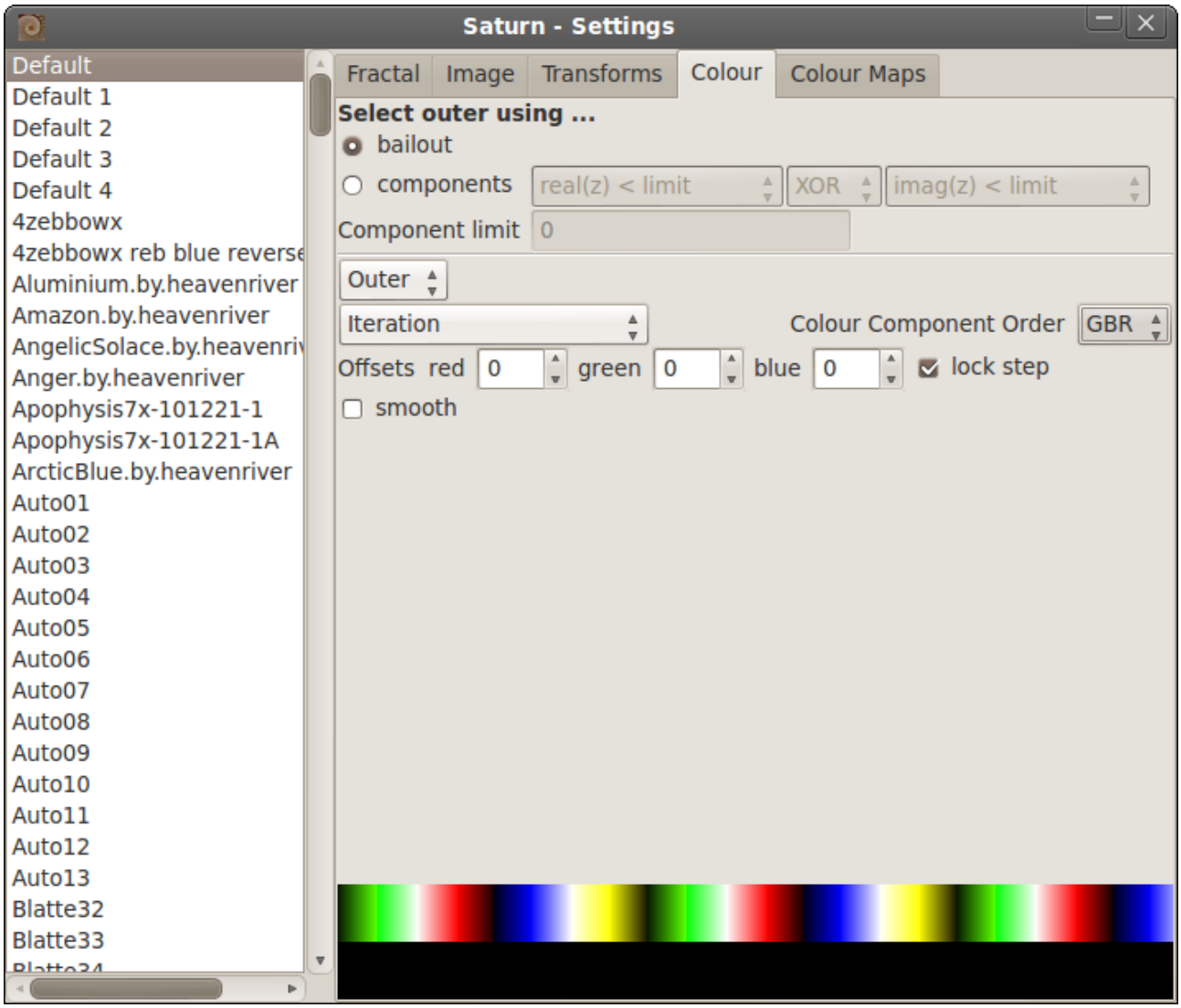
Points that meet the colour section condition are coloured as outer and those that do not are coloured as inner. The appearance of the tab is dependent on the colouring method.

Fixed Colour

This method colours all the selected points the same colour and is usually used for inner colouring. The default is black.



Iteration



The iteration colour method is used for outer colouring, the iteration used is the iteration where the bailout condition was met. The iteration number is used as an index into the colour map.

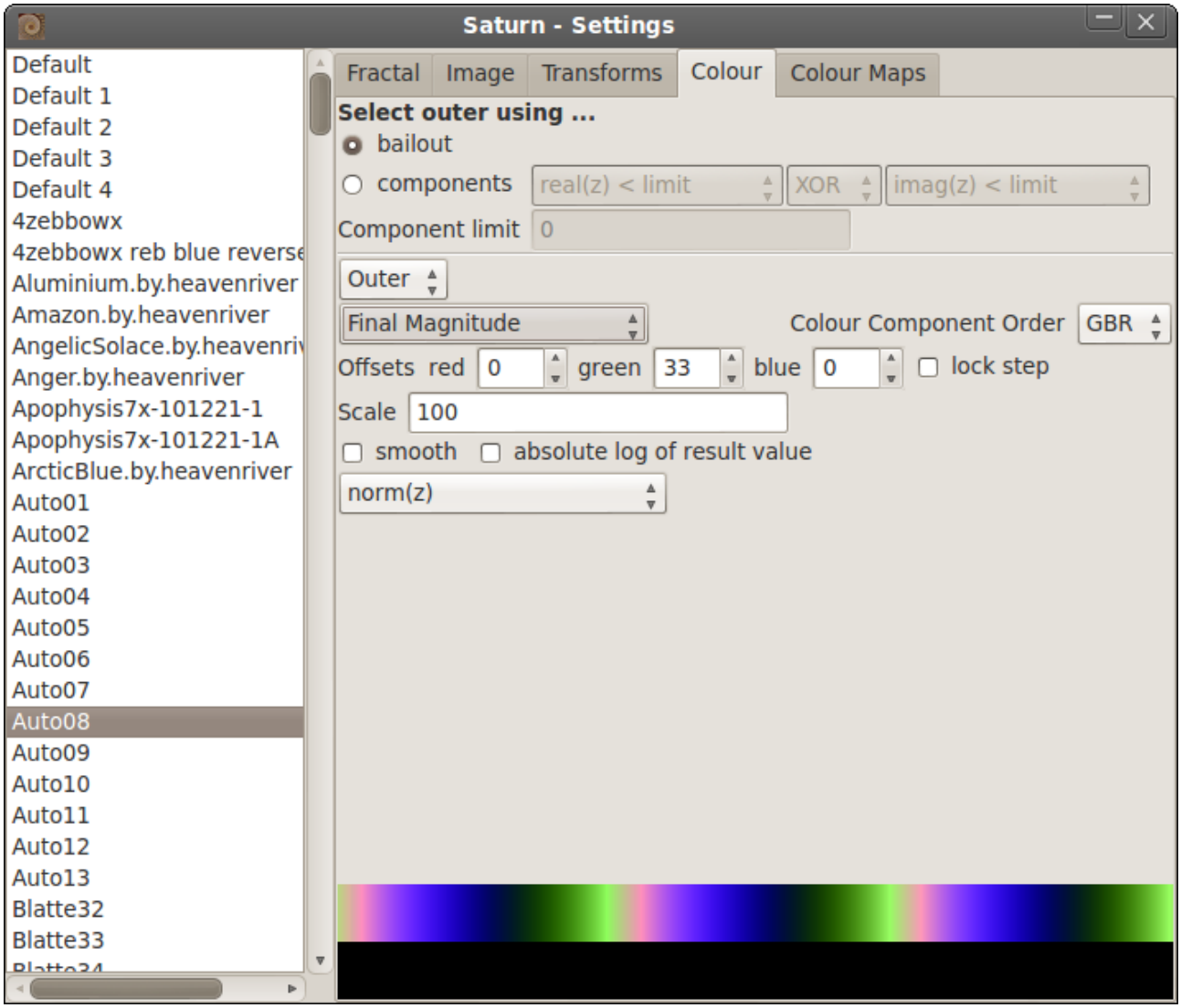
The smooth option can be enabled which modifies the index used by adding:

$$\text{bailout limit} \div (\text{abs}(z) + 0.000000001)$$

to the iteration number, so a blended colour is used.

This modified method is also known as “Continuous Potential”.

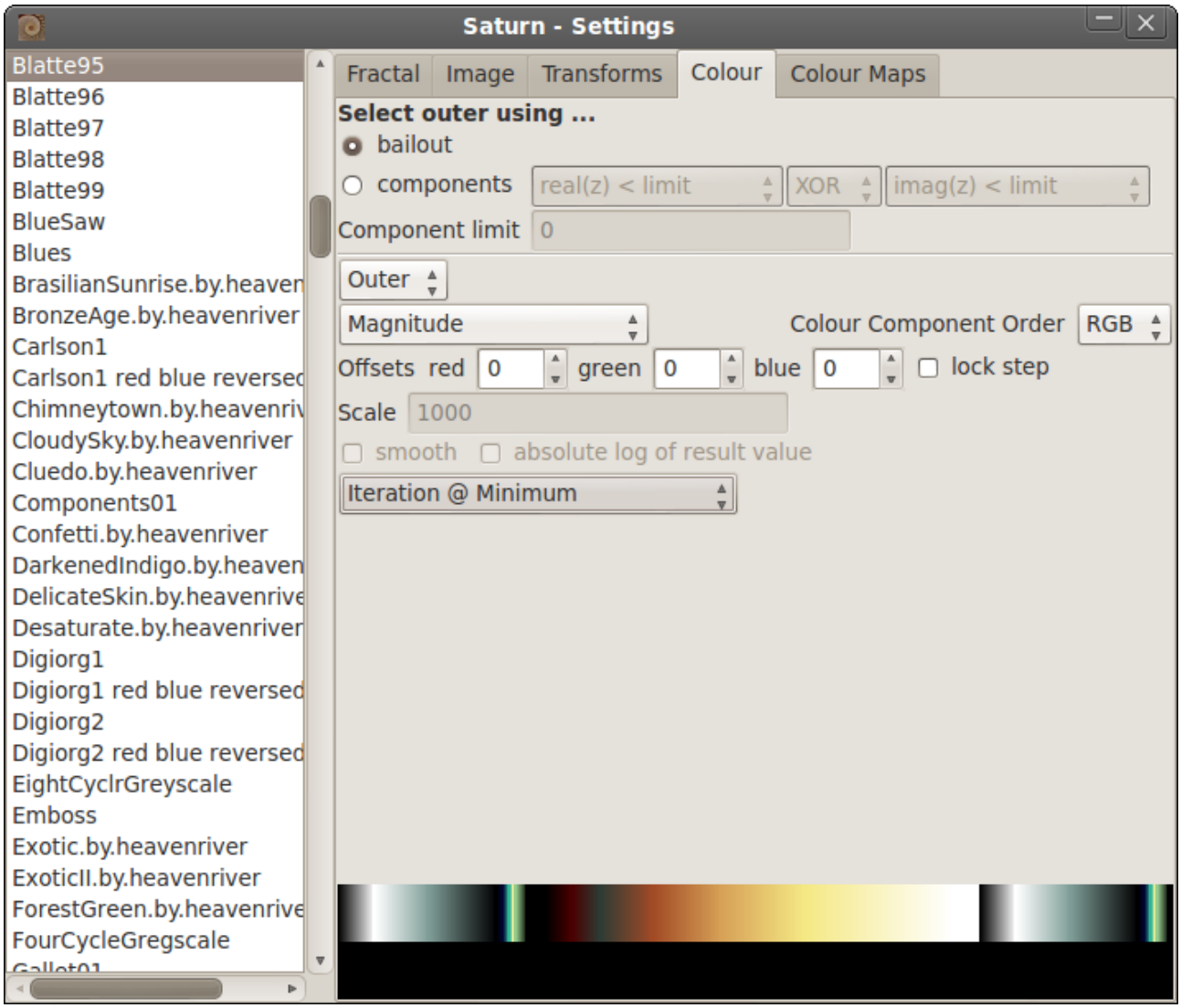
Final Magnitude



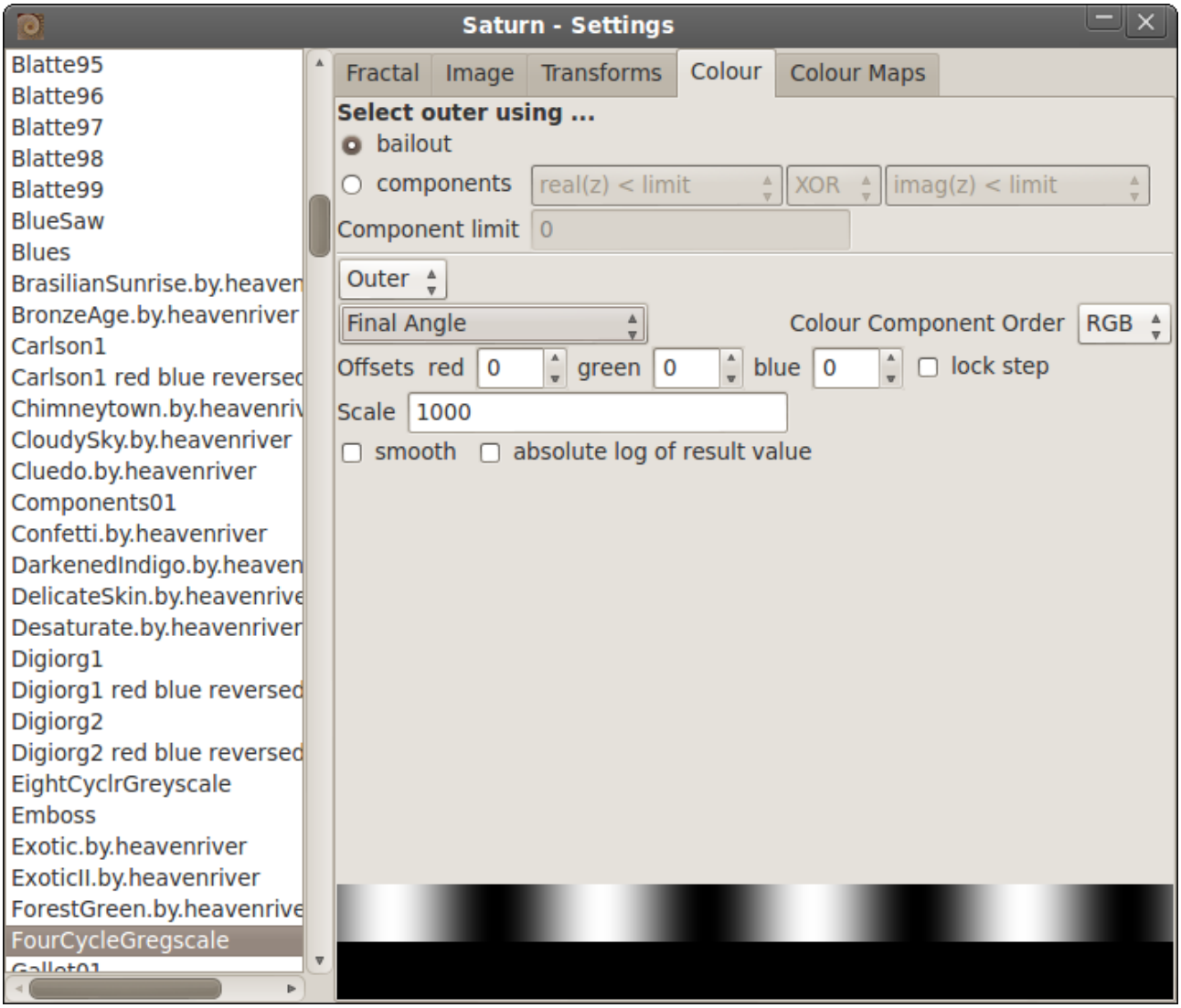
The options combo box provide the following modification of the final value to be used:

1. norm(z)
2. smaller of real(z) and imag(z)
3. larger or real(z) and imag(z)
4. real(z)
5. abs(real(z))
6. imag(z)
7. abs(imag(z))
8. real(z) + imag(z)
9. abs(real(z) + imag(z))
10. real(z)*imag(z)
11. abs(real(z)*imag(z))

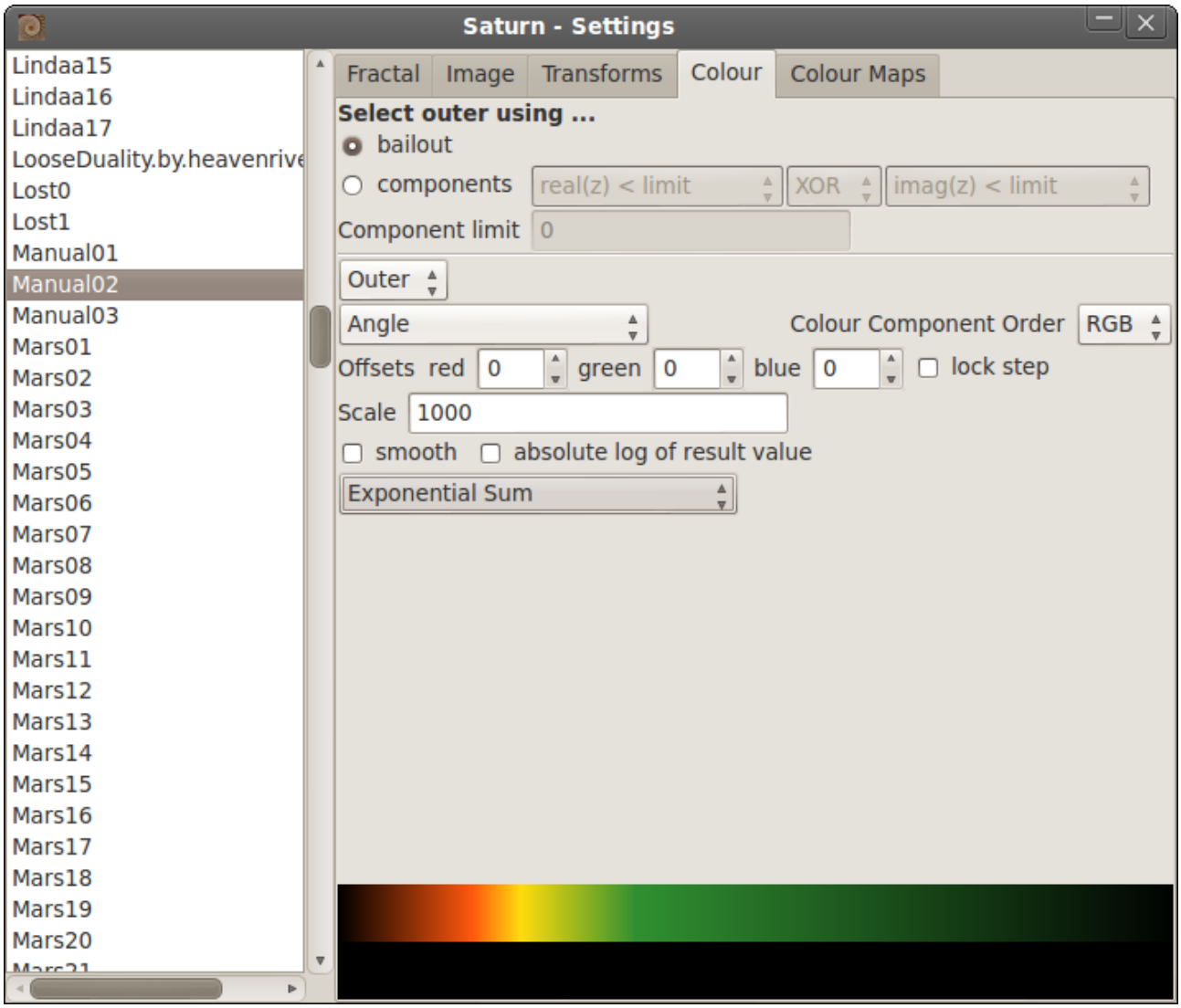
Magnitude and Change in Magnitude



Final Angle

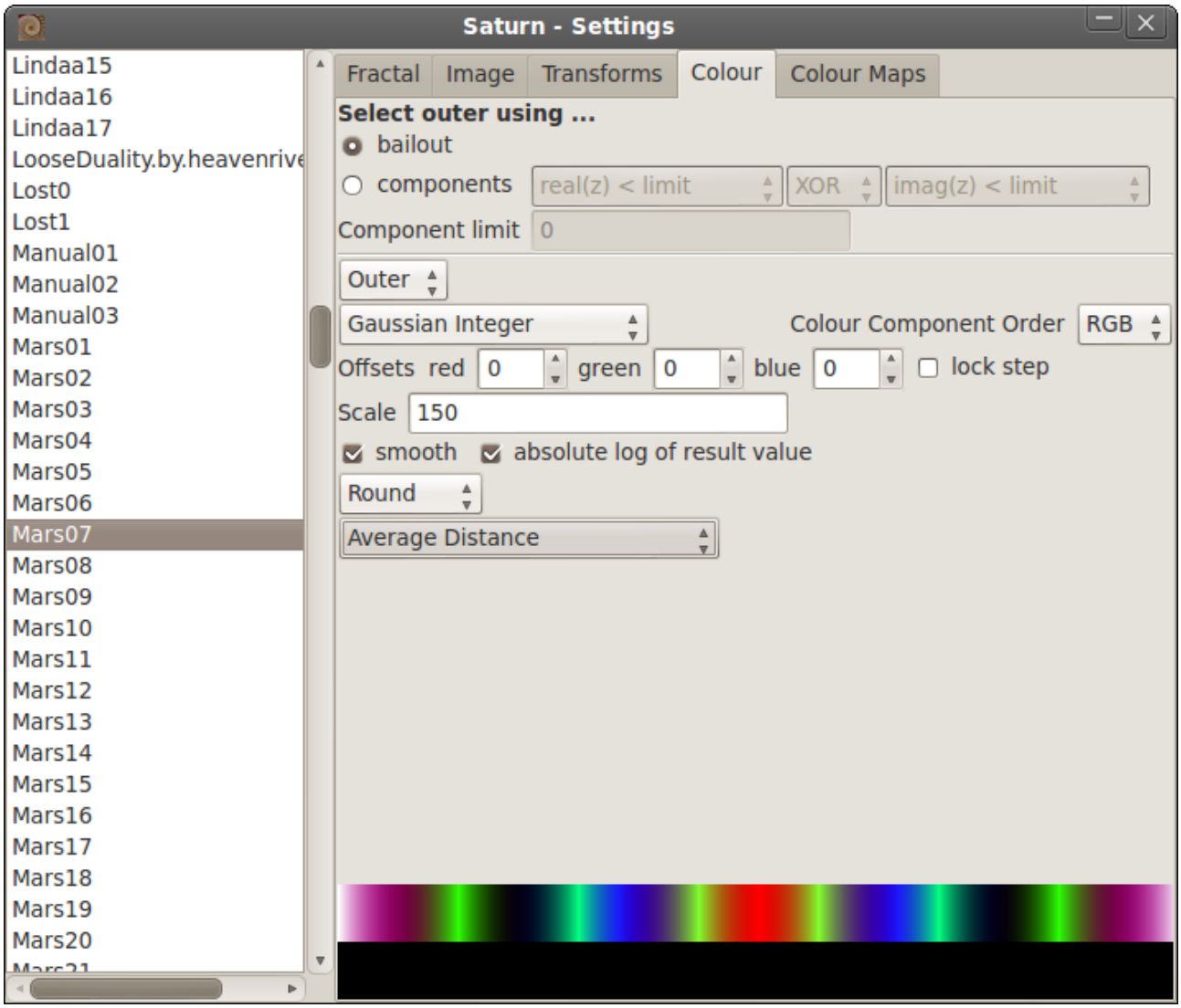

Saturn - Settings
Blatte95
Blatte96
Blatte97
Blatte98
Blatte99
BlueSaw
Blues
BrasilianSunrise.by.heaven
BronzeAge.by.heavenriver
Carlson1
Carlson1 red blue reverse
Chimneytown.by.heavenriv
CloudySky.by.heavenriver
Cluedo.by.heavenriver
Components01
Confetti.by.heavenriver
DarkenedIndigo.by.heaven
DelicateSkin.by.heavenrive
Desaturate.by.heavenriver
Digiorg1
Digiorg1 red blue reversed
Digiorg2
Digiorg2 red blue reversed
EightCyclrGreyscale
Emboss
Exotic.by.heavenriver
ExoticII.by.heavenriver
ForestGreen.by.heavenrive
FourCycleGregscale
Fractal
Image
Transforms
Colour
Colour Maps
Select outer using ...
bailout
components
real(z) < limit
XOR
imag(z) < limit
Component limit
0
Outer
Final Angle
Colour Component Order
RGB
Offsets red
0
green
0
blue
0
lock step
Scale
1000
smooth
absolute log of result value

Angle and Change in Angle



The argument of z after each iteration is found which is the angle of the vector from the origin to the complex number on the complex plane, the value is in the range -π to π radians, if the value is negative then 2π radians is added to the value. Change in value is the absolute difference between the current value and the value for the previous iteration, initial value is zero. The value in radians is used to accumulate statistics.

Gaussian Integer



The Gaussian Integer used for collecting statistics is one of four different types:

- Round
- Ceiling
- Floor
- Truncate

The value used to determine the colour is one of these statistics:

1. Minimum Distance
2. Iteration @ Minimum Distance
3. Angle @ Minimum Distance
4. Maximum Distance
5. Iteration @ Maximum Distance
6. Angle @ Maximum Distance
7. Average Distance
8. Minimum Angle
9. Average Angle

10. Maximum Angle
11. Maximum Distance/Minimum Distance
12. Range
13. Variance
14. Standard Deviation
15. Coefficient of Variation

Trap Distance and Change in Trap Distance

Most orbit traps have common parameters, some have an option for a extra point at the centre of the trap and the Steiner Chain trap has options controlling how the trap is constructed.

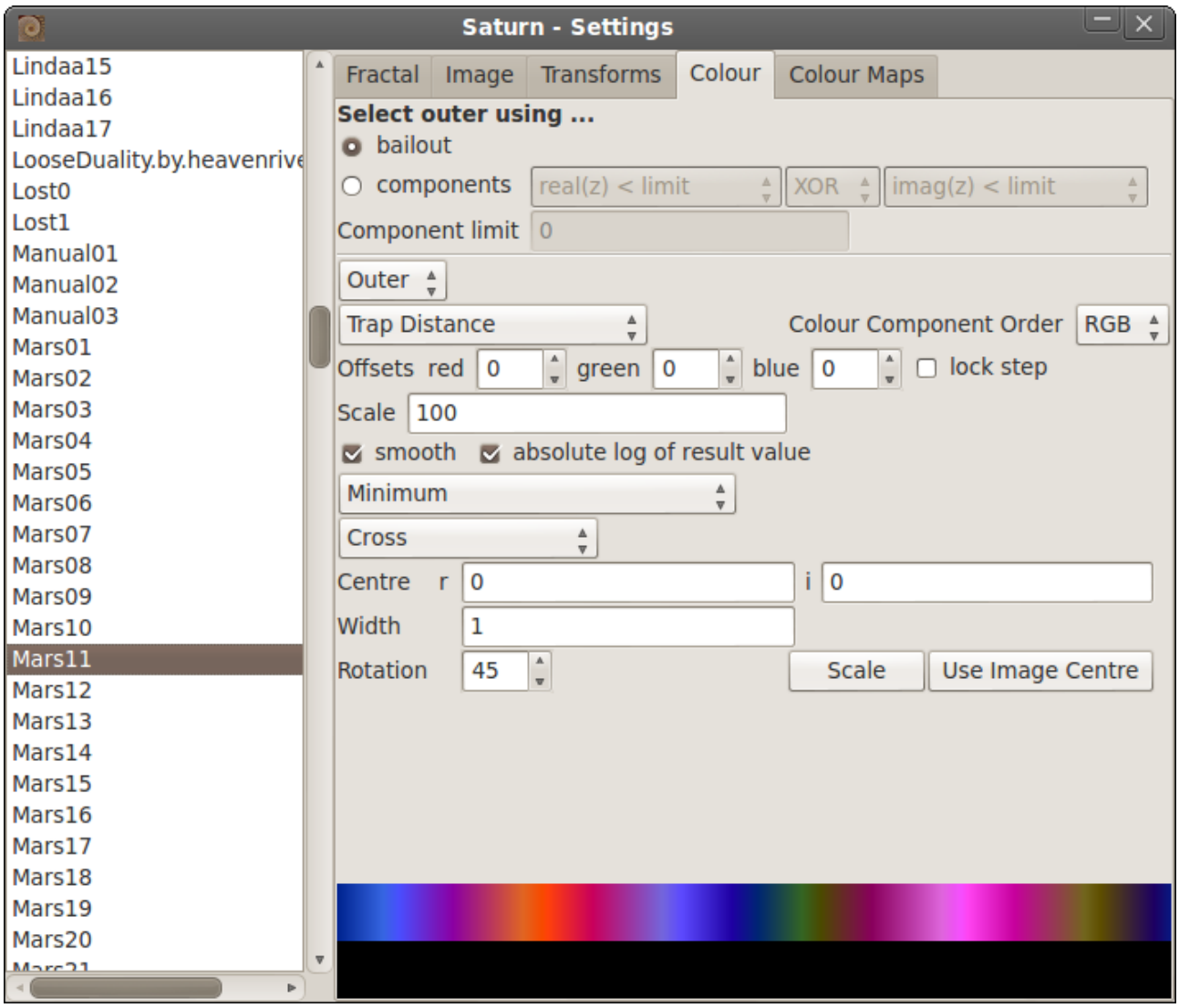
Statistics of the distance or change in distance to the orbit trap are accumulated and the statistic is used to determine the colour.

There are a number Orbit Trap shapes, the appearance of these orbit traps are shown in the Orbit Trap Appendix.

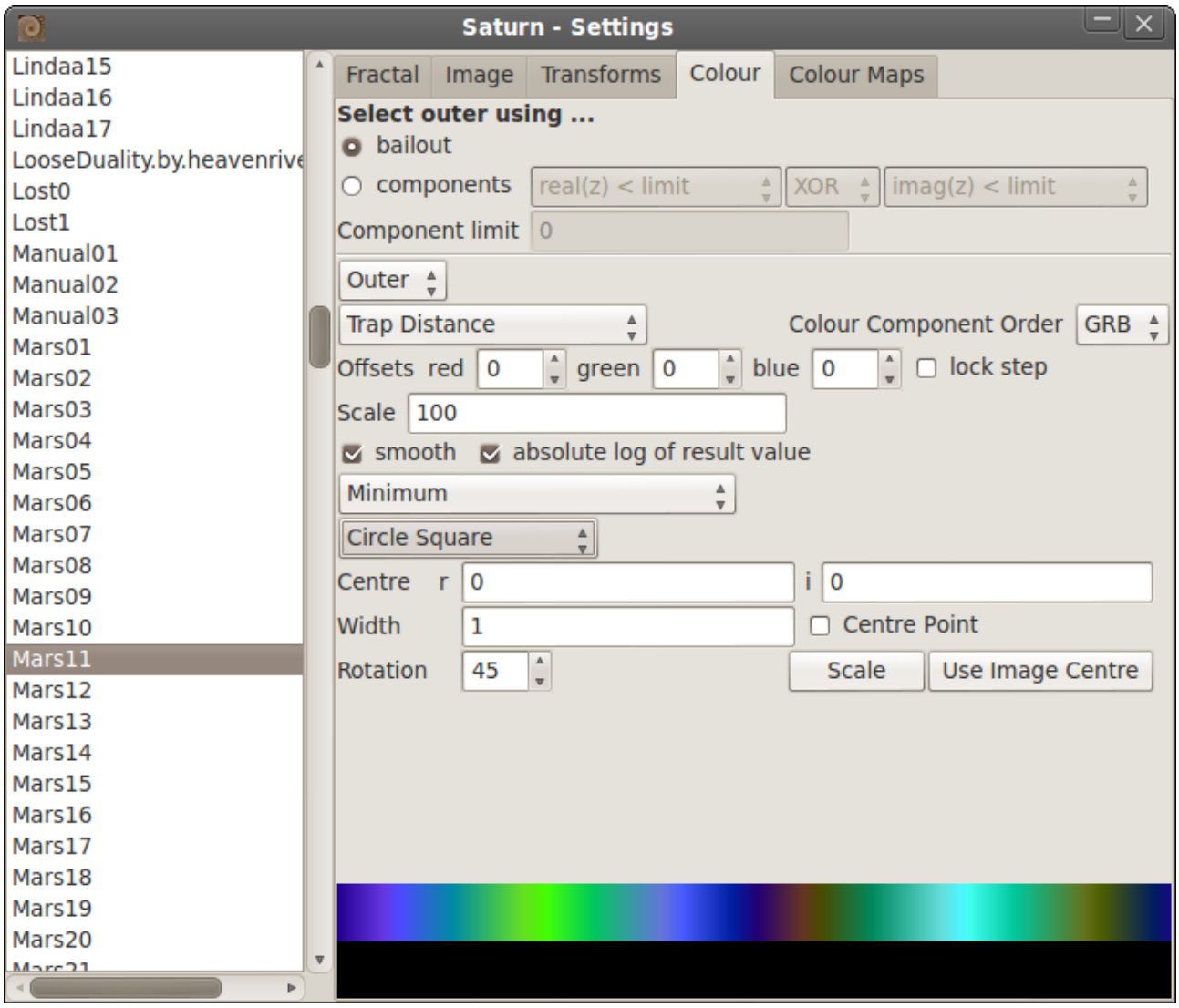
The size and position of the orbit trap are specified using the position and size entry boxes, the size is the width of the orbit trap e.g. for triangles it is the length of the base and for circles it is the diameter. The position of the trap can be set to the current image centre using the button below the position entry boxes and the size can be scaled to the image by pressing the button below the size entry box. The size set for scale to image is half the current width of the image. The size is disabled for the Point orbit trap as its size is always zero.

The orientation of the trap can be altered by rotating it about its centre, the spin button is disabled for all traps that are unaltered by rotation such as point and circle.

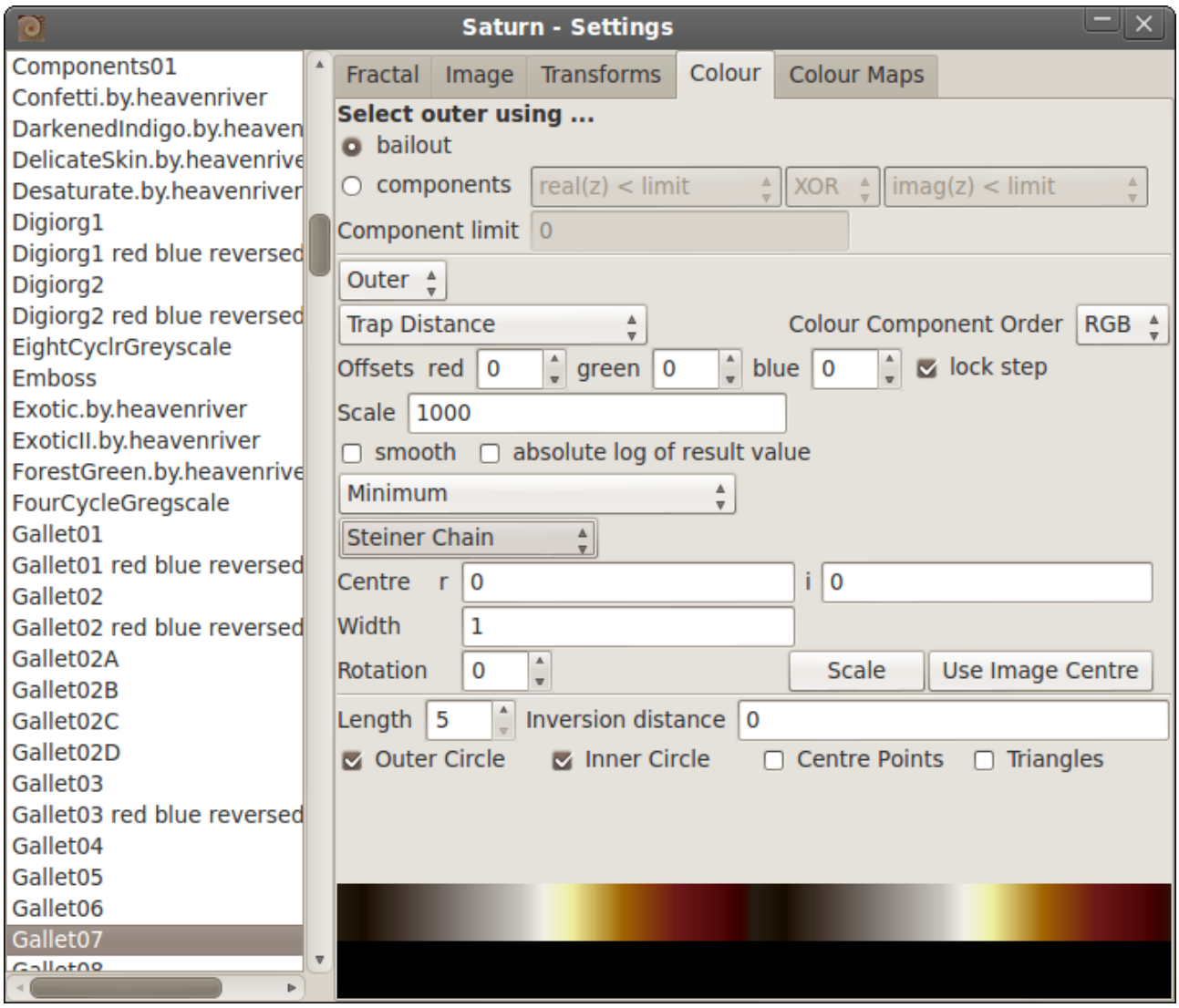
Note: the more complicated the orbit trap the greater the number of calculations required to find the distance to the orbit trap. There will be a noticeable delay before the first image and current number of iterations are displayed.



An orbit trap without an optional central point.



An orbit trap with an optional central point.

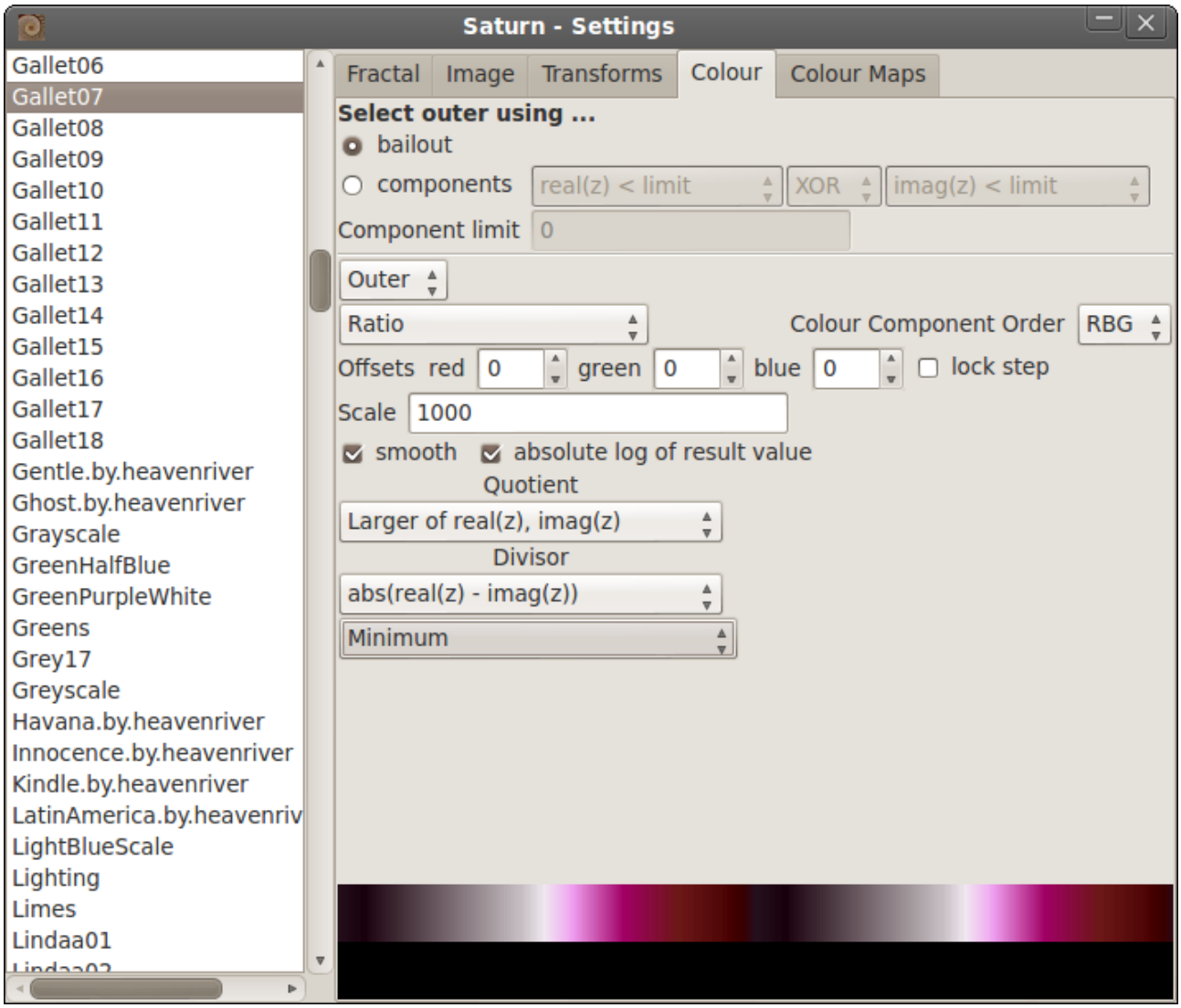


The extra options for the Steiner Chain Orbit Trap control how the chain is constructed:

1. Length, the number of circles enclosed within the inner and outer circles. The values are restricted to the range 5 to 64.
2. Inversion distance, a value of zero produces a ring of circls of equal sizes, non-zero values distort the chain producing a chain of circles of varying sizes. The values are restricted to the range -0.9 to 0.9.
3. Outer Circle, display of the outer enclosing circle is optional.
4. Inner Circle, display of the inner enclosing circle is optional.
5. Centre Points, an extra option not usually associated with Steiner Chains, if this option is set a centre point will be displayed for the chain circles.
6. Triangles, an extra an extra option not usually associated with Steiner Chains, if this option is set an inscribed triangle will be used in the chain circles. One of the corners of the inscribed triangles touches the inner enclosing circle.

Examples of Steiner Chain orbit traps can be found in the orbit trap appendix which show the shapes of the various orbit traps.

Ratio and Change in Ratio



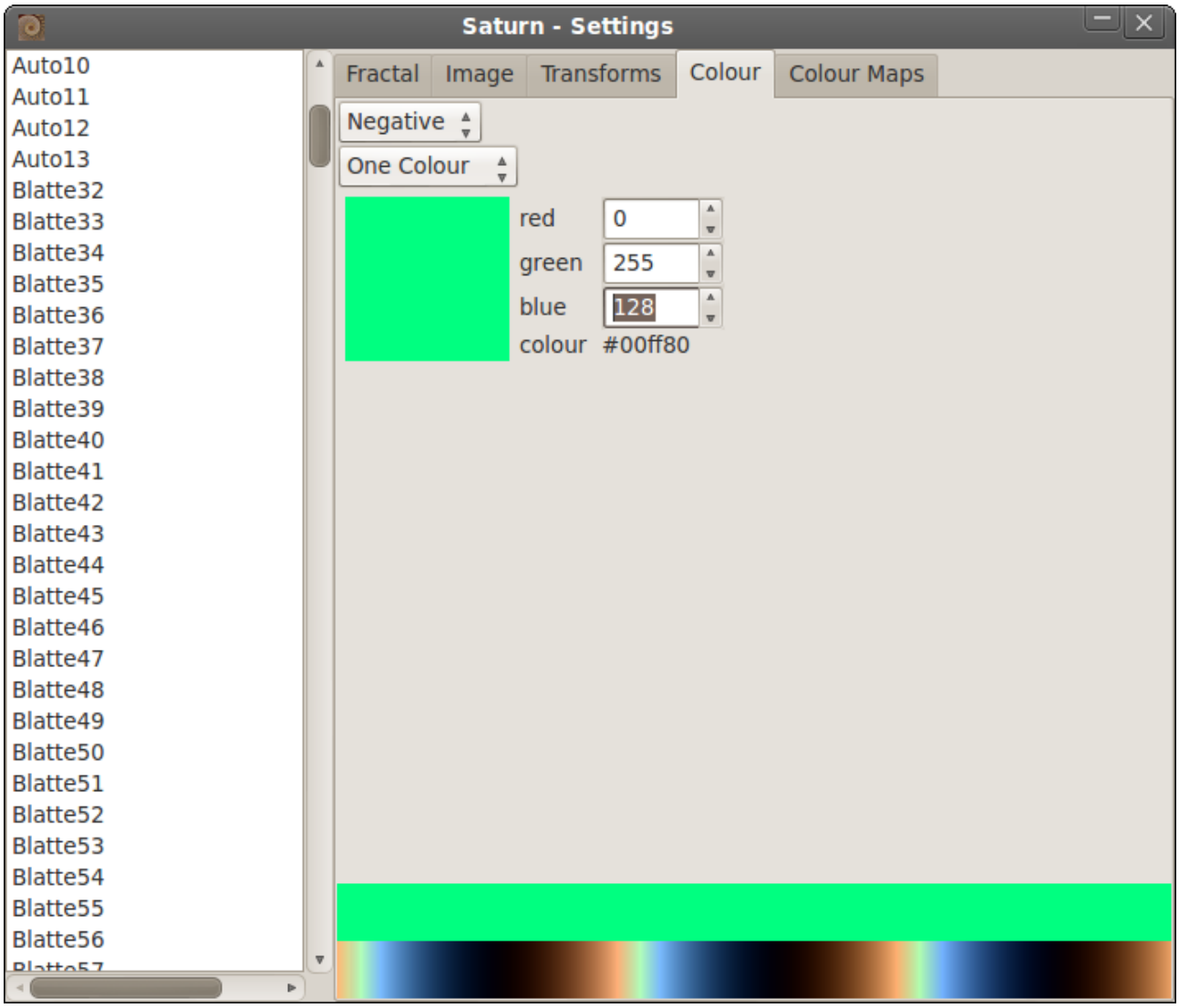
These colour methods select colour based on statistics gathered of the ratio of values, for change in ratio the values used are the differences between the current ratio and the previous ratio.

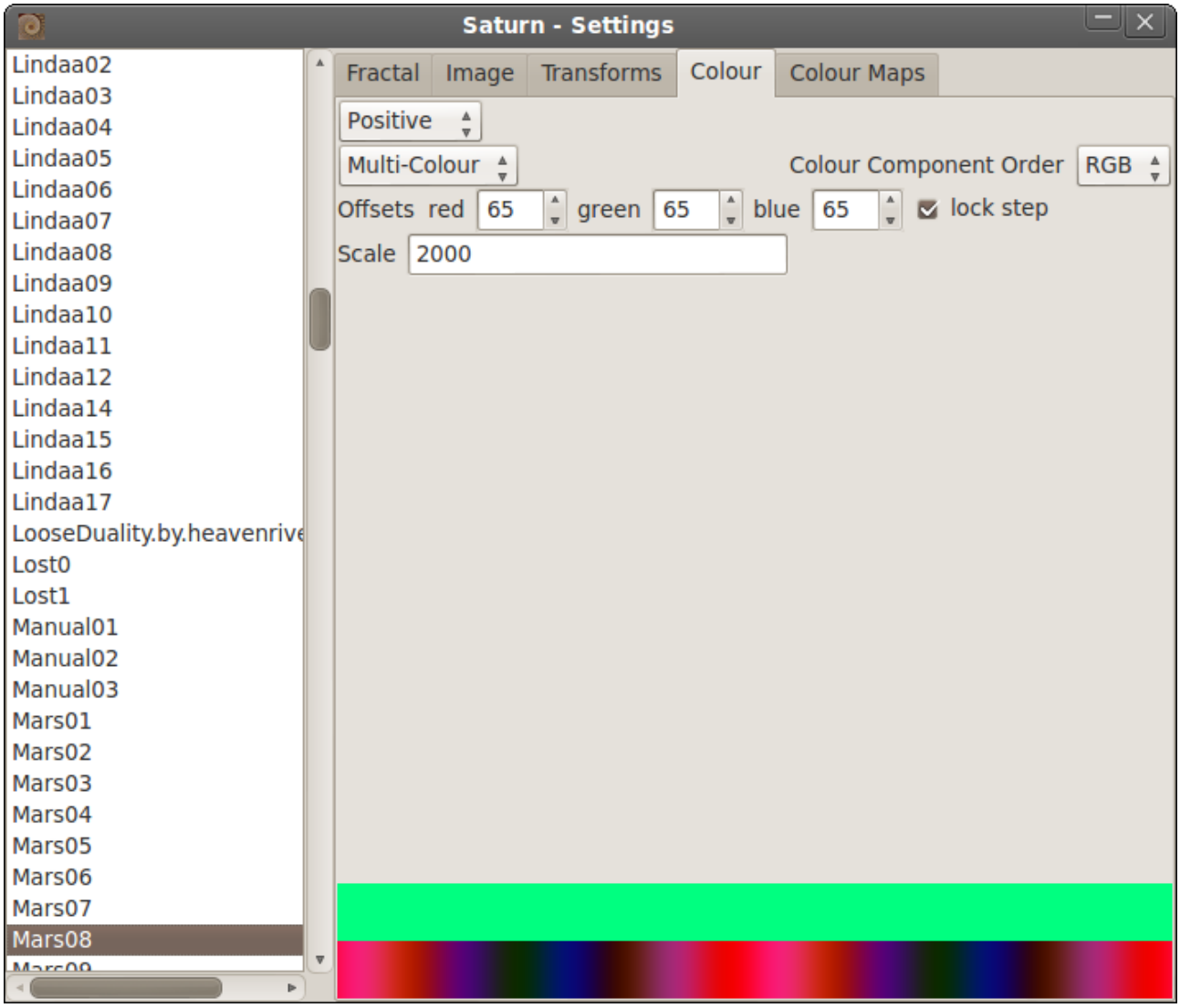
The quotient and divisor of the ratio can be selected from these values:

1. Smaller of real(z), imag(z)
2. Larger of real(z), imag(z)
3. abs(real(z))
4. abs(imag(z))
5. abs(real(z) + imag(z))
6. abs(real(z) – imag(z))
7. abs(real(z)) + abs(imag(z))
8. abs(abs(real(z)) – abs(imag(z)))
9. abs(real(z))*abs(imag(z))
10. abs(z)
11. norm(z)

Use of the same values for quotient and divisor will result in a uniform colour.

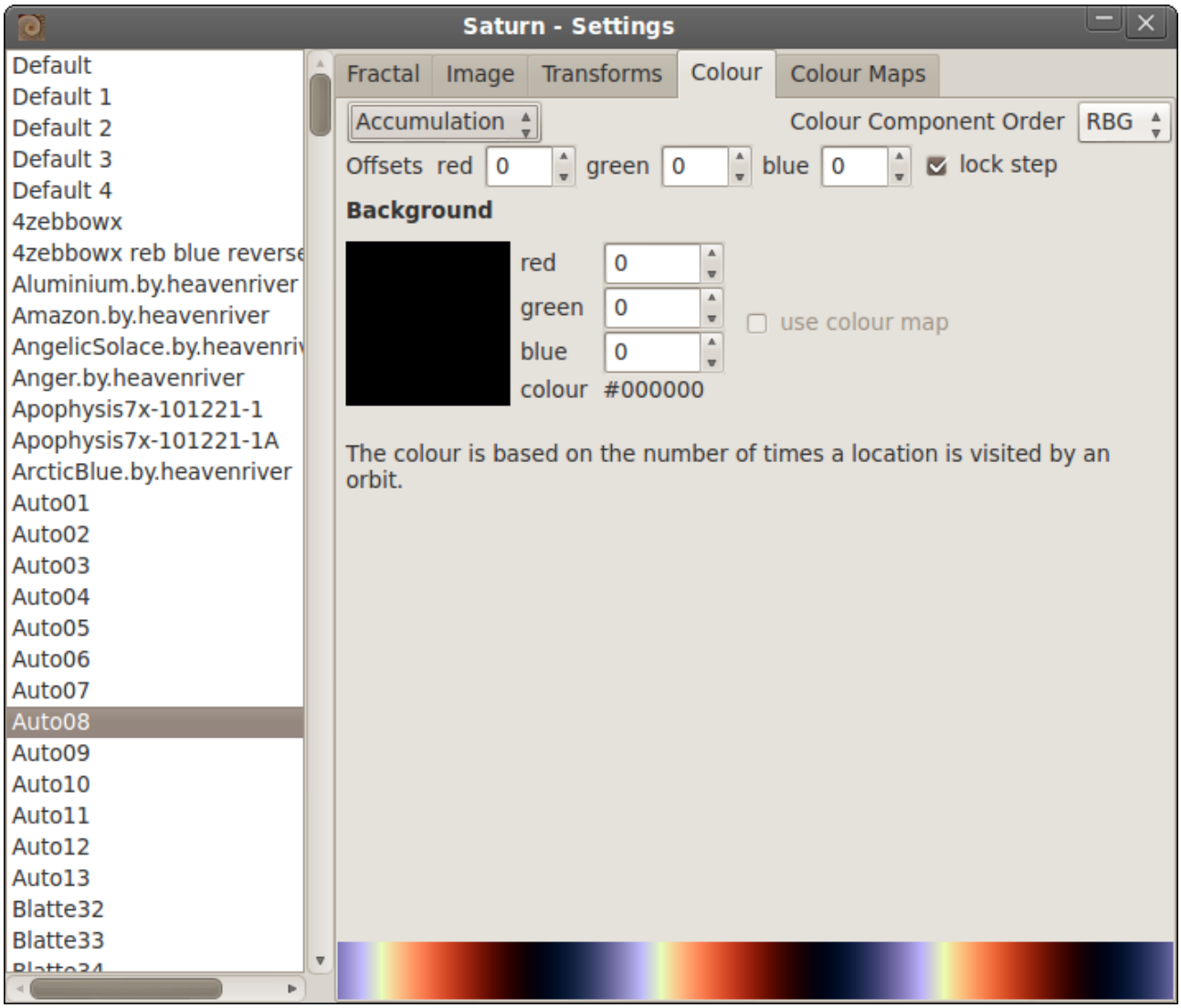
3.2.5.3 Colour – Lyapunov

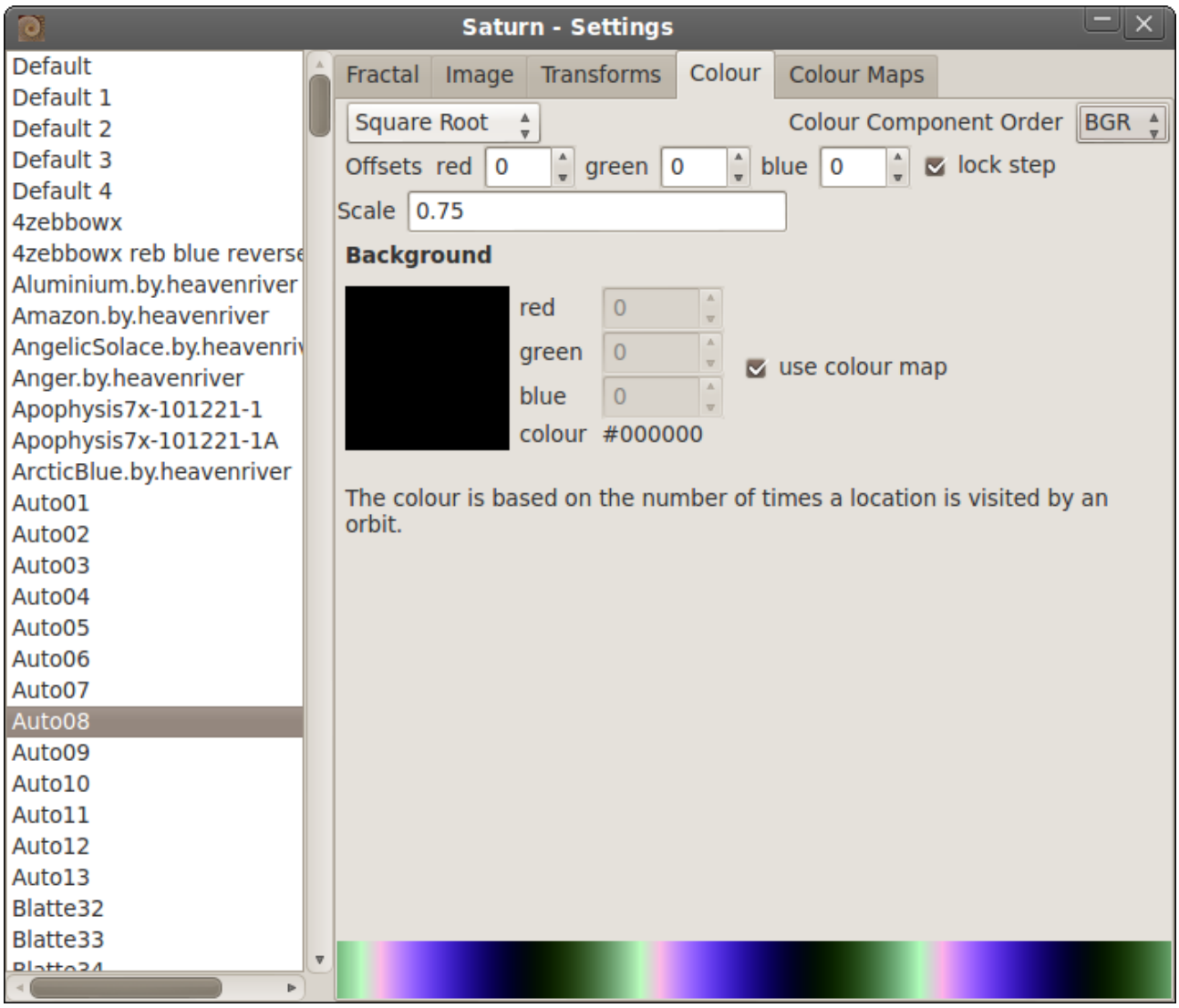


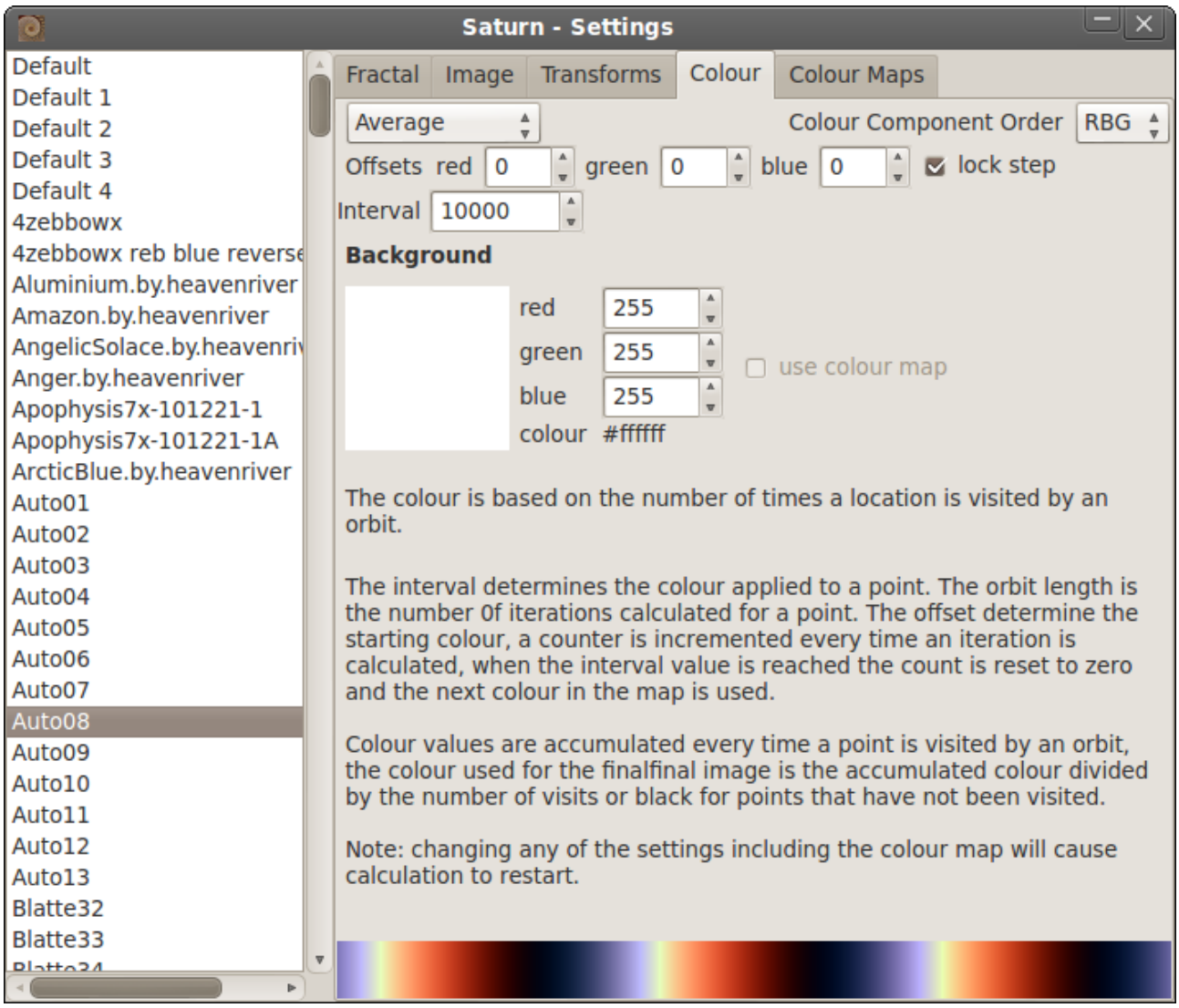

Saturn - Settings
Lindaa02
Lindaa03
Lindaa04
Lindaa05
Lindaa06
Lindaa07
Lindaa08
Lindaa09
Lindaa10
Lindaa11
Lindaa12
Lindaa14
Lindaa15
Lindaa16
Lindaa17
LooseDuality.by.heavenrive
Lost0
Lost1
Manual01
Manual02
Manual03
Mars01
Mars02
Mars03
Mars04
Mars05
Mars06
Mars07
Mars08
Fractal
Image
Transforms
Colour
Colour Maps
Positive
Multi-Colour
Colour Component Order
RGB
Offsets red
65
green
65
blue
65
lock step
Scale
2000

3.2.5.4 Colour - Orbit Plotted Fractals

The colour methods are based on the number of times a location is visited by an orbit (hits). Accumulation simply uses the hits value, square root and logarithm either take the square root or logarithm of the hits. A scale factor can be specified for both Square Root and Logarithm methods.




Saturn - Settings
Default
Default 1
Default 2
Default 3
Default 4
4zebbowx
4zebbowx reb blue reverse
Aluminium.by.heavenriver
Amazon.by.heavenriver
AngelicSolace.by.heavenri
Anger.by.heavenriver
Apophysis7x-101221-1
Apophysis7x-101221-1A
ArcticBlue.by.heavenriver
Auto01
Auto02
Auto03
Auto04
Auto05
Auto06
Auto07
Auto08
Auto09
Auto10
Auto11
Auto12
Auto13
Blatte32
Blatte33
Fractal
Image
Transforms
Colour
Colour Maps
Square Root
Colour Component Order
BGR
Offsets red 0 green 0 blue 0
lock step
Scale 0.75
Background
red 0
green 0
blue 0
colour #000000
use colour map
The colour is based on the number of times a location is visited by an orbit.



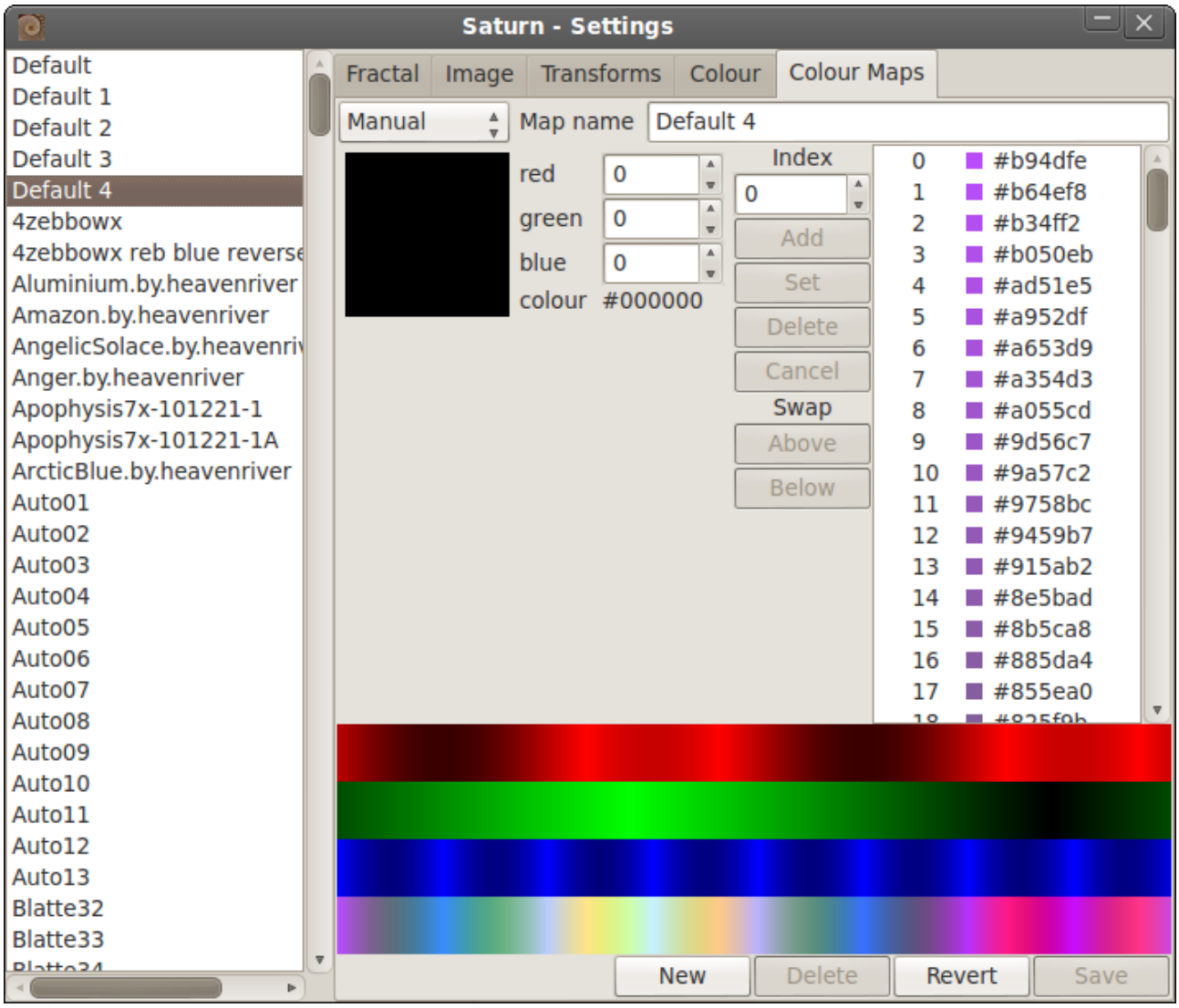
Note: this colour method only appears to work well for fractals that were originally called Pickover Popcorn and are now produced using Pickover Popcorn 4F & 6F fractals calculated using the Julia algorithm.

3.2.6 Colour Maps Tab

This tab handles the management of colour maps. Maps can be added, edited, renamed and deleted. The editing of a map is dependent of whether it is a manually edited map or an auto generated map and the appearance of the tab varies accordingly.

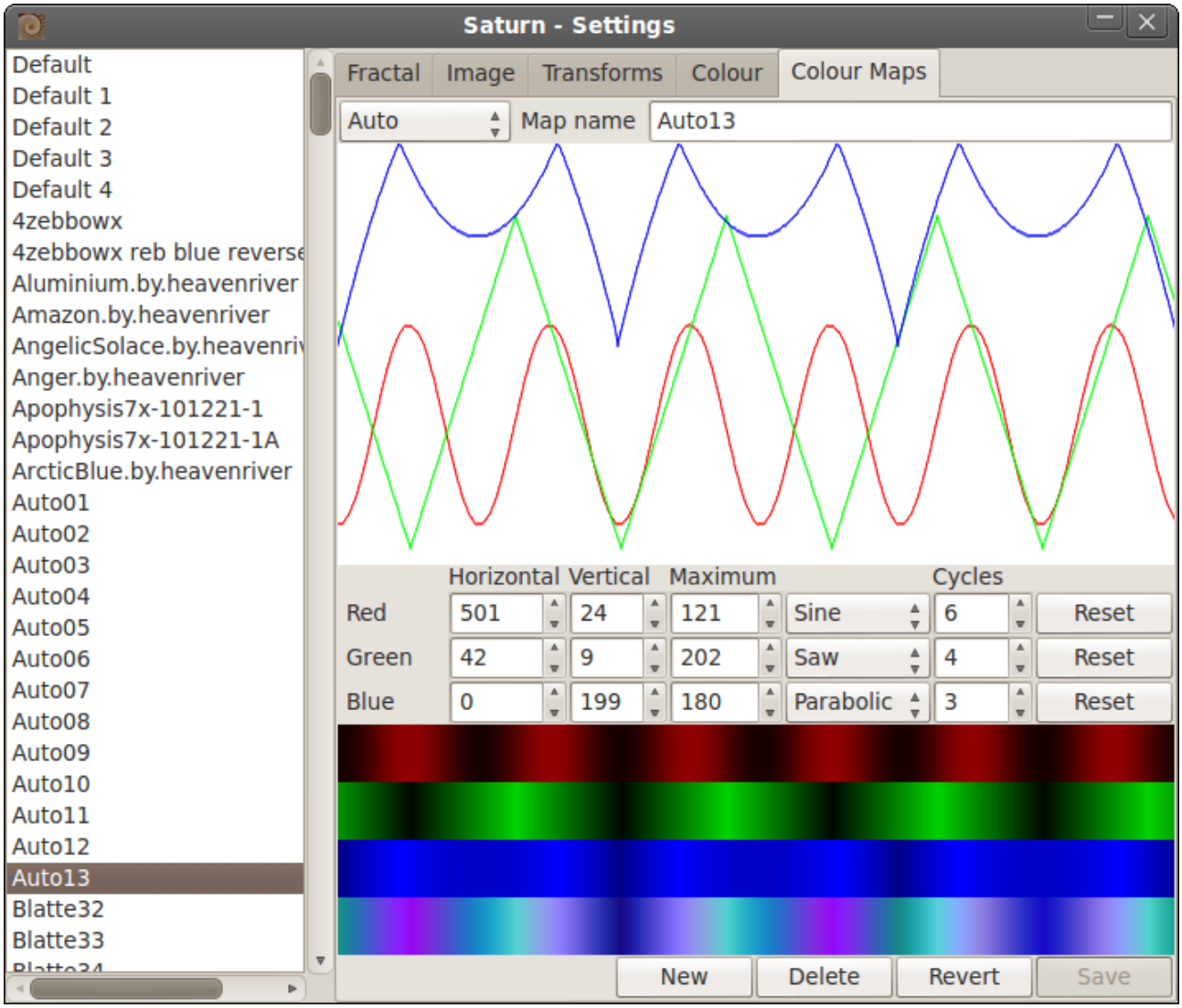
There are four colour map rectangles one for each component and one for the components combined.

Manual

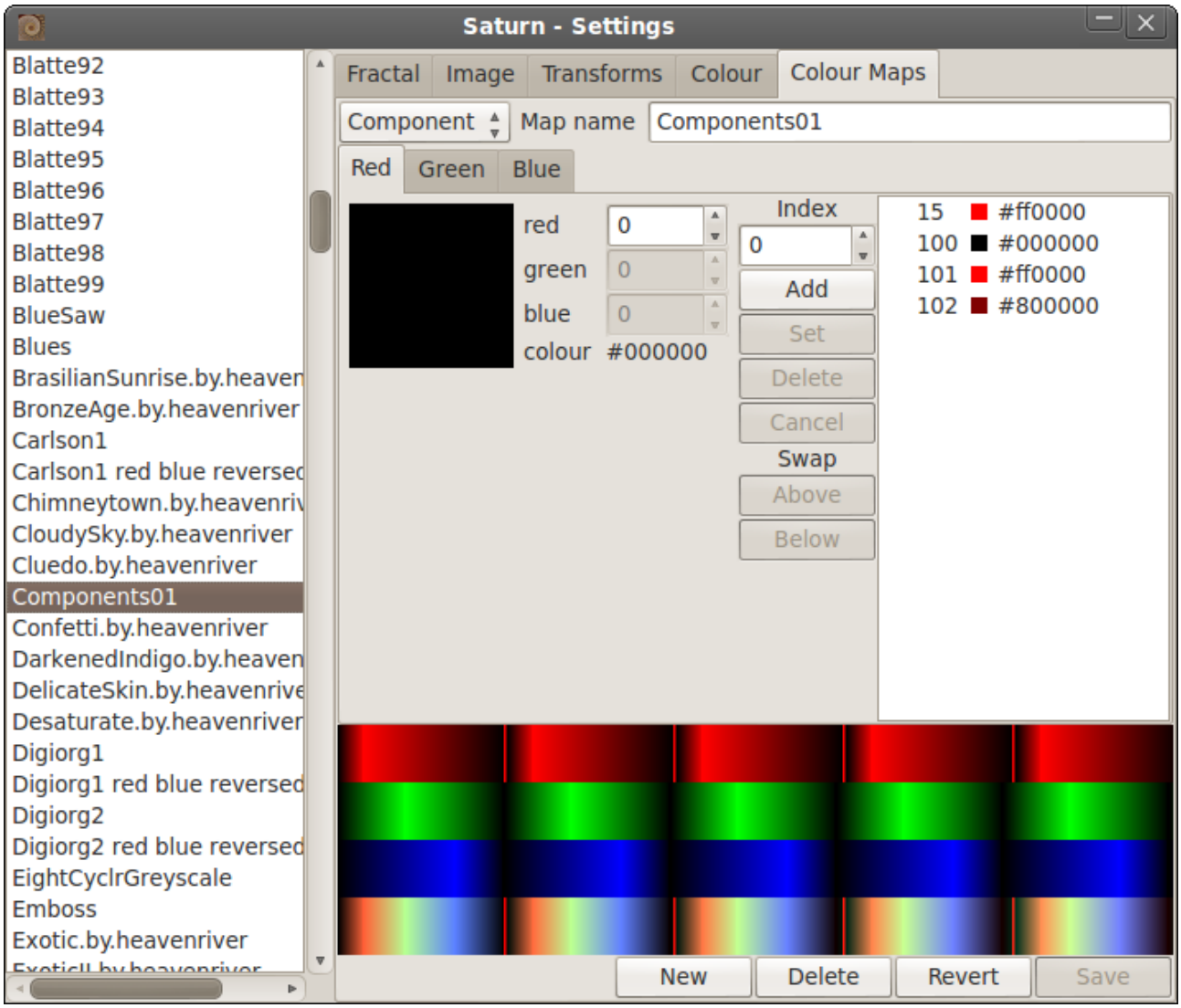


Auto

Auto colour maps are generated based on waveforms, the amplitude, horizontal and vertical positions can be adjusted. The type and number of cycles of the waveform can also be changed. If the part of the waveform is greater than 255 or less than 0 then that part of the waveform will be reflected.

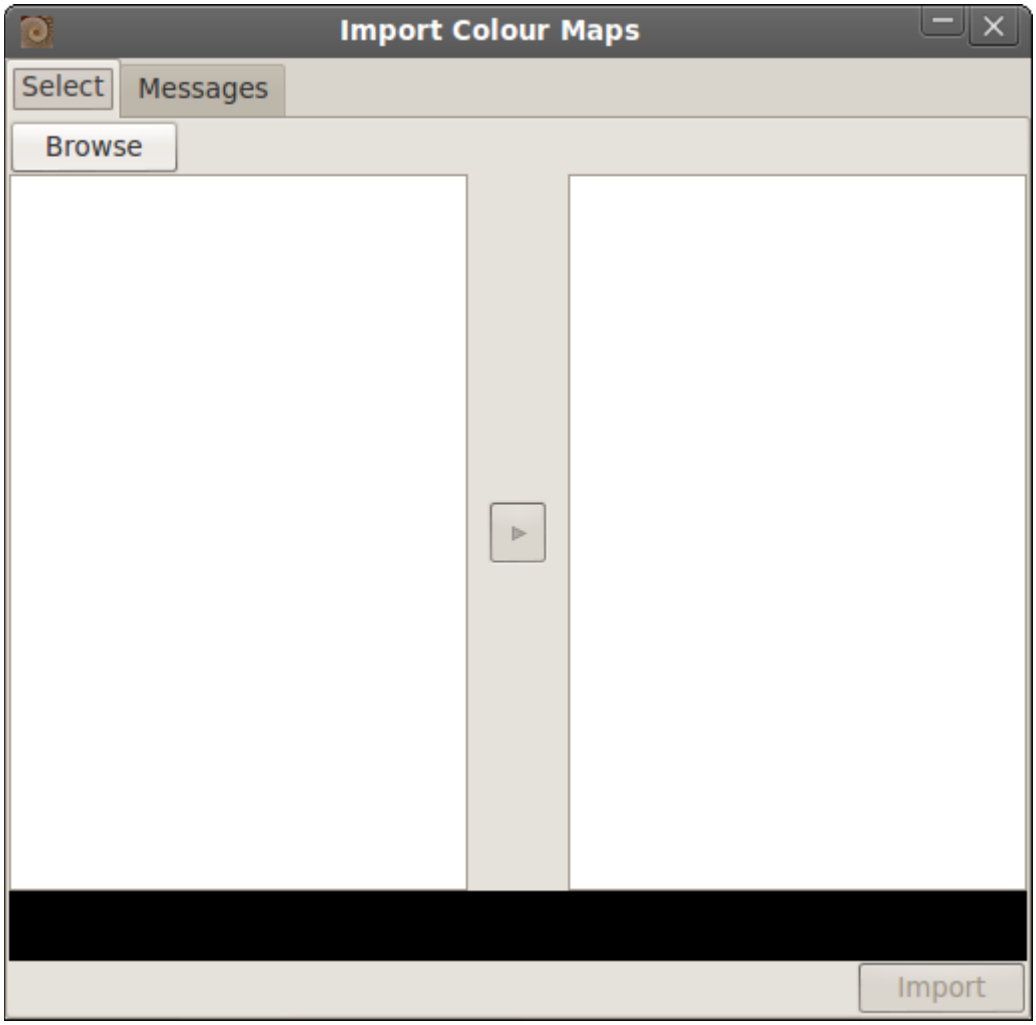


Component


Saturn - Settings
Blatte92
Blatte93
Blatte94
Blatte95
Blatte96
Blatte97
Blatte98
Blatte99
BlueSaw
Blues
BrasilianSunrise.by.heaven
BronzeAge.by.heavenriver
Carlson1
Carlson1 red blue reversed
Chimneytown.by.heavenriv
CloudySky.by.heavenriver
Cluedo.by.heavenriver
Components01
Confetti.by.heavenriver
DarkenedIndigo.by.heaven
DelicateSkin.by.heavenrive
Desaturate.by.heavenriver
Digiorg1
Digiorg1 red blue reversed
Digiorg2
Digiorg2 red blue reversed
EightCyclrGreyscale
Emboss
Exotic.by.heavenriver
Fractal
Image
Transforms
Colour
Colour Maps
Component
Map name
Components01
Red
Green
Blue
red
green
blue
colour
#000000
Index
Add
Set
Delete
Cancel
Swap
Above
Below
15 #ff0000
100 #000000
101 #ff0000
102 #800000
New
Delete
Revert
Save

3.3 Import Colour Maps Window

The initial import colour maps window is shown below:



Colour map files can be added to the tree on the left hand side (the source tree) using the browse button, three types of colour map can be imported, .map, .scm and .ugr. Several colour maps can be defined in .ugr and .scm files, only one colour map is defined in a map file.

All the colour maps in a file can be copied to the list on the right (importing list) using the arrow button or individual colour maps in a file can be copied. If the source tree cursor is on a file name it is disabled if all its colour maps have been copied to the importing list, if the cursor is on a colour map name the arrow button is disabled if the colour map is also in the importing list.

If the source tree cursor is on a file name there is no cursor displayed on the importing list, if the cursor is on a colour map name and that colour map is in the importing list the importing list cursor is displayed on the corresponding colour map name.

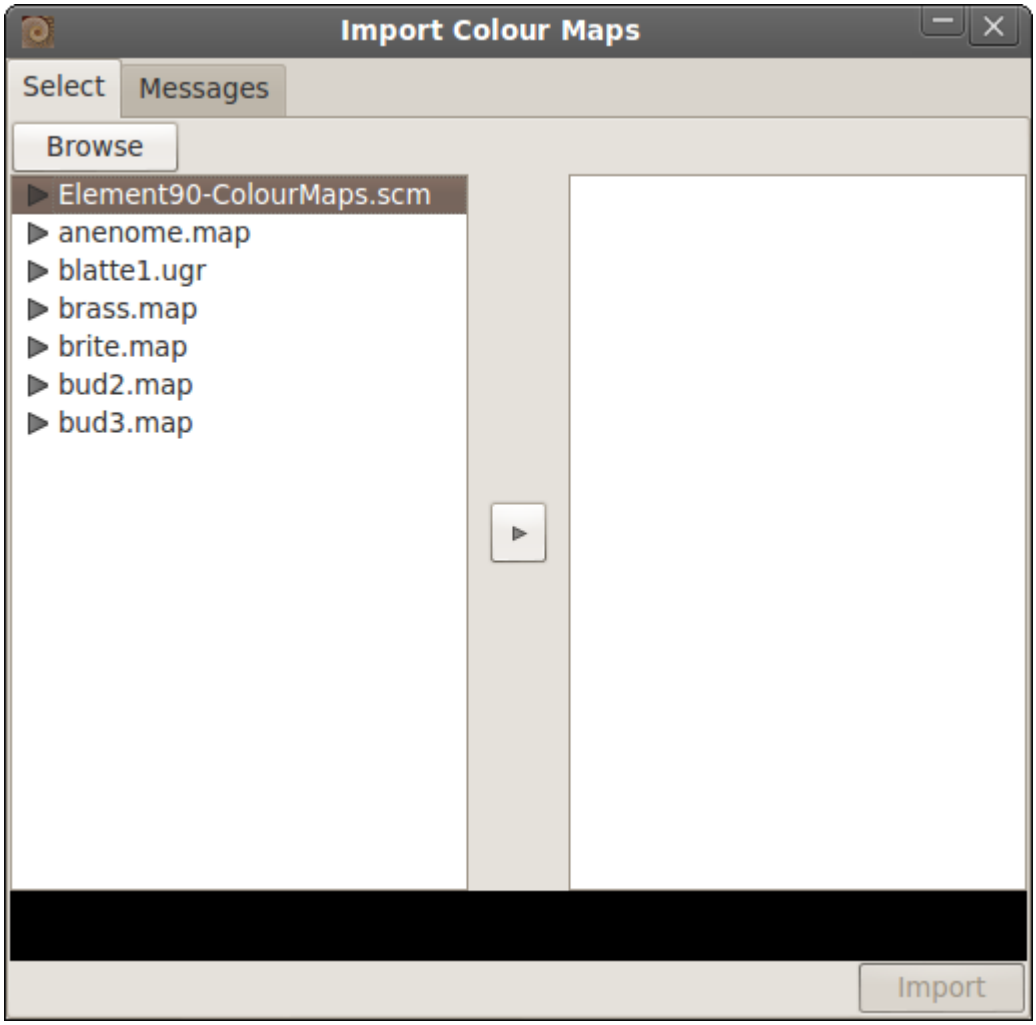
The names in the importing list can be different from the colour map names in the source tree, this is because the names in the import file may already exist in the Saturn's collection of colour maps, when the colour map is copied to the importing list the name will be altered if it already exists by the addition “~*n*” where *n* is a number, for example if a colour map is called “Reds” and “Reds” already exists the name become “Reds~0” if that name also already exists then “Reds~1” will be

tried and so on until an unused name is found. The altered names are intended to be temporary and can changed once the maps have been imported..

The black rectangle above the import button will display the current colour map only when the source tree cursor is on a colour map name (the cursor may be hidden), the rectangle will be black when the cursor is on a file name.

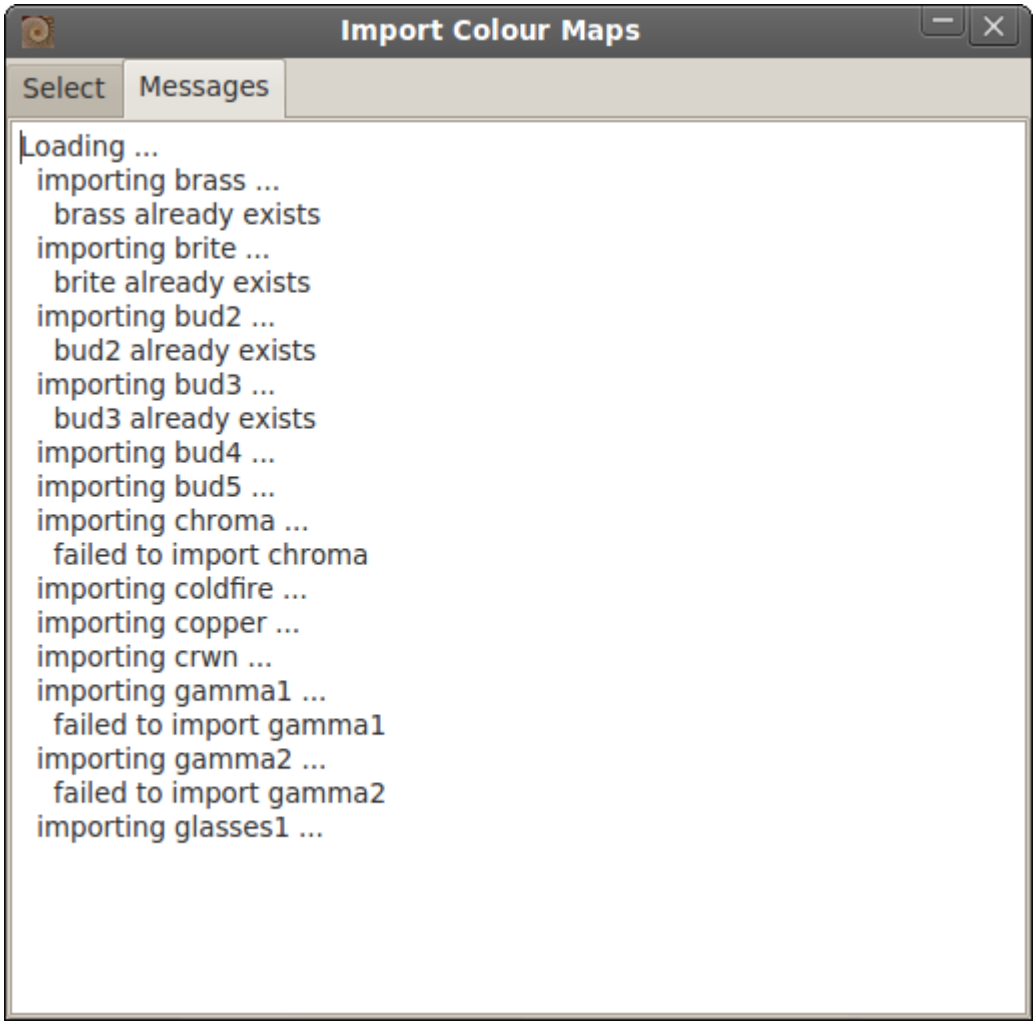
The import button is enabled when there are colour maps in the importing list, when the button is pressed the colour maps are added to Saturn's collection, the importing list is cleared and the imported colour maps are also removed from the source tree, if all the maps in a file are imported the file name is also removed. On completion of the import the import button is disabled and no source cursor is displayed.

The window below shows two files that have been added to the source tree.



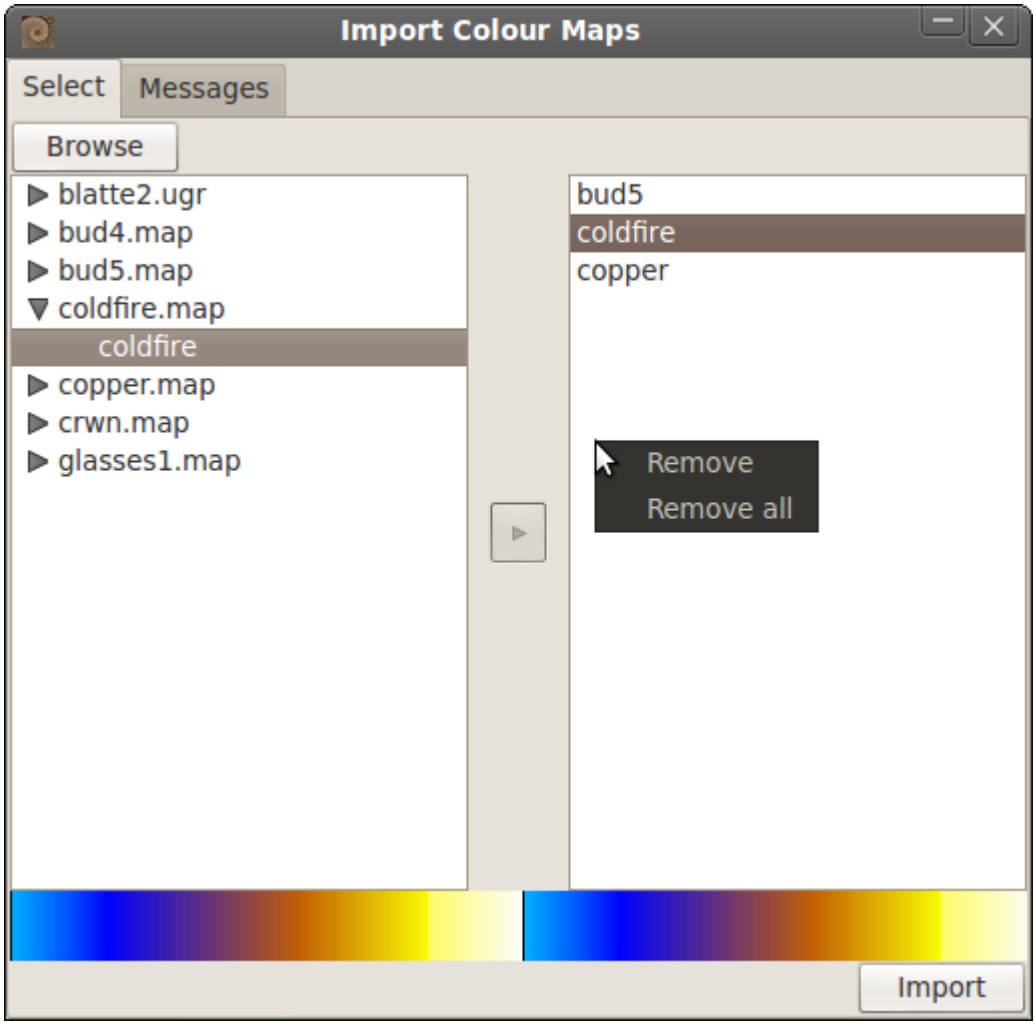
When files are added to the source tree a check is made for each map to determine whether it is already in Saturn's collection of colour maps if the map is already in the collection it is disregarded, the file name is only added to the list if it contains new colour maps. The message tab is used to report duplicate colour maps and any errors that occurred while adding an import file to the source tree. If Saturn fails to read a colour map file it is not added to the source tree.

An example of the messages tab:



The import of .map files can often fail because they contain text on lines after the triplet of numbers representing a colour, removing the text will allow the .map file to be imported.

The window below shows the colour maps defined in a file, colour maps in the importing list and a pop up menu.



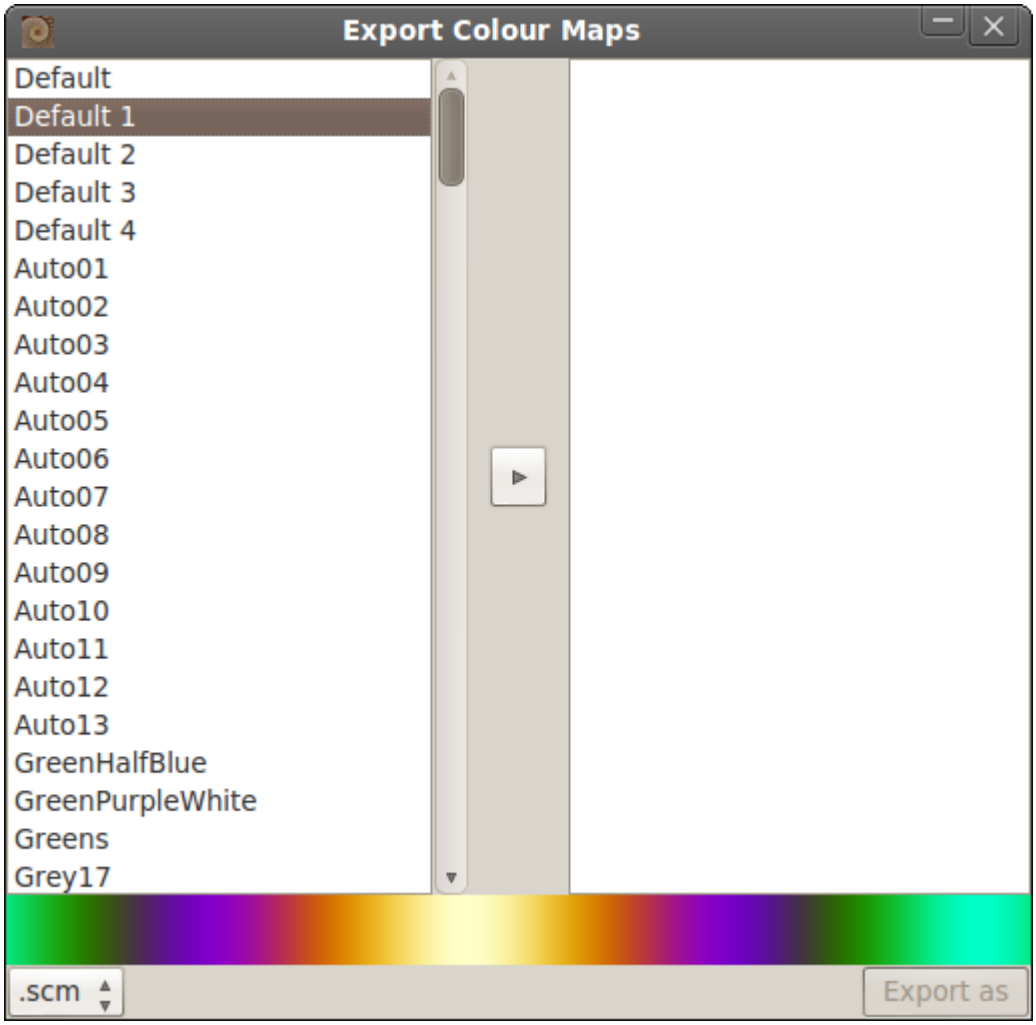
The pop up menu shown allows all the colour maps in the importing list or an individual colour map to be removed.

The pop up menu can also be used to delete all the source tree entries, individual colour maps or a file of colour maps.

The pop up menus are displayed when the right mouse button is clicked on the source tree or the importing list.

3.4 Export Colour Maps Window

The export colour maps window when it is first started:



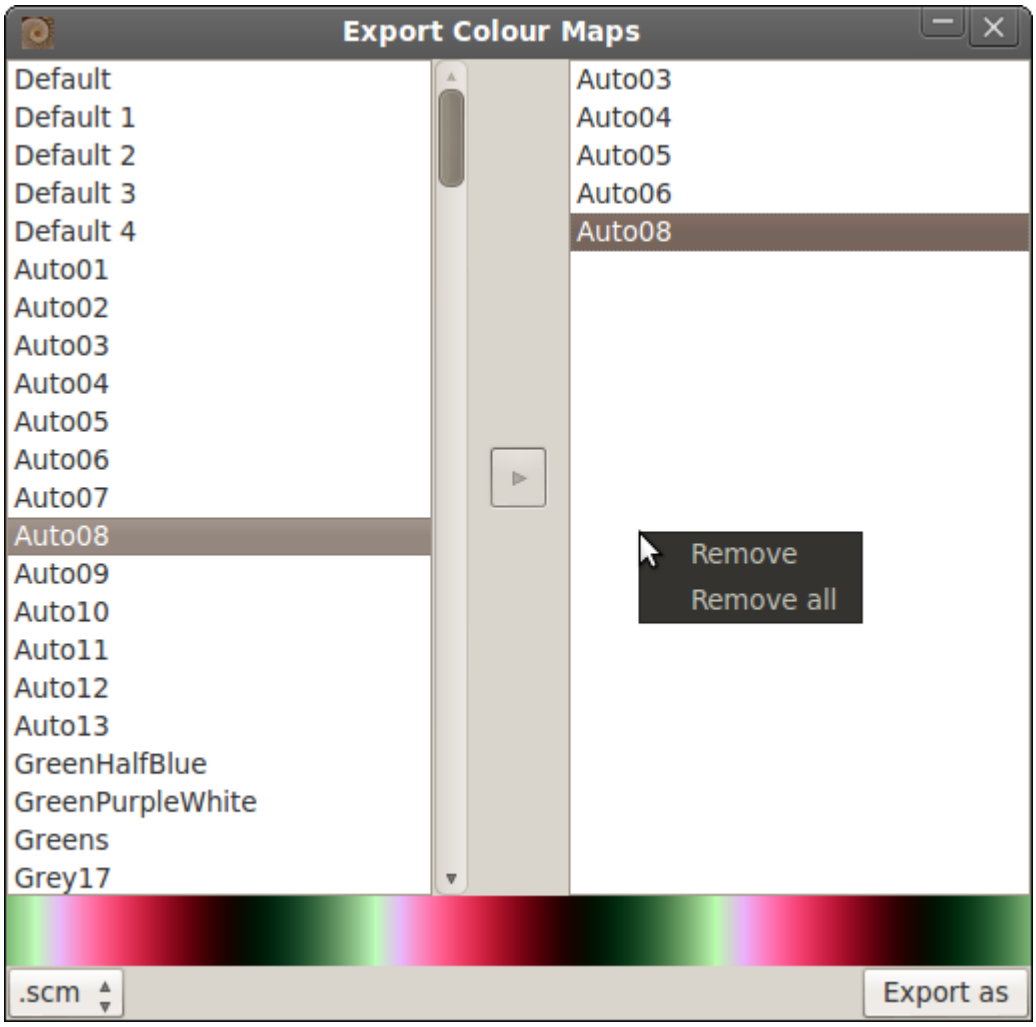
A list of all the colour maps defined in Saturn is on the left hand side, on the right hand side is a list of the colours to be exported, initially this empty and as there are no colour maps to export and the export as button is disabled.

Colour maps are copied into the right hand side (the exporting list) using the arrow button which copies the colour map at the current cursor, the cursor then moves down one position. The cursor on the left hand list (the source list) can be moved up and down, if the colour map at the cursor is also in the exporting list then the arrow button is disabled and a cursor is also displayed on the exporting list. The colours in the colour map are displayed at the bottom of the window.

The cursor can also be moved on the exporting list, the arrow button will be disabled and a cursor will be displayed on the corresponding colour map in the source list.

The “export as” button is enabled as soon as there is a colour map in the exporting list, when the “export as” button is pressed you will be asked for a file name, if the file extension is omitted the appropriate extension will be added. The are two two type of file that can be exported, .scm or .ugr. When the colour maps have been exported the exporting list is not cleared as a second export of the same colour maps in a different format may be required.

The exporting list can have individual colour map or all colour maps removed using a pop up menu. To remove an individual colour map select it using the cursor and then right click on the colour map, a pop up menu will appear with two items: Remove and Remove All.



4 Fractal Types

4.1 Escape Time

Most of the fractals implemented in Saturn and Titan are so called escape time fractals which are iterated a maximum number of times or until a bailout condition is met when processing for a given point stops. In some cases no bailout condition is specified in which case all points are iterated to the maximum number of iterations.

There are two basic types of algorithm, the Mandelbrot algorithm and the Julia algorithm.

4.1.1 Mandelbrot Algorithm

The Mandelbrot algorithm is used when at least one of the complex parameters is set to use the location in the complex plane.

The algorithm builds a picture by iterating the formula until the maximum number of iterations has been reached or the escape (or bailout) condition has been met. The value of the complex parameters set to use the complex plane is varied and is different for all locations calculated. Usually the initial value of z is a fixed value commonly zero except where the use of zero produces infinity or an undefined value (e.g. log of zero), Saturn also provides for the use of the location in the complex plane, any complex plane transforms can be ignored.

4.1.2 Julia Algorithm

The Julia algorithm is used when none of the complex parameters are set to use the location in the complex plane. It is essentially the same as the Mandelbrot algorithm except that only the starting value is varied based on the location in the complex plane being calculated.

4.2 Orbit Plotting

Orbit fractals consist of all the points calculated using the orbit fractal's formula. The initial value is a point on a plane when the formula is evaluated a new point on the plane is plotted which is then used to determine the next point to be plotted. The number of points plotted is specified by the orbit length, the set of plotted points being the orbit. A grid is overlaid on the plane with a number of columns and rows, the position on the plane of the starting value corresponds with the row and column position on the plane, the value calculated is a position on the plane and plotted position is found by converting to the nearest row and column on the grid.

The grid used to calculate the the fractals, the calculating area, is larger than the display area and only the points that are in the display area are displayed. The orbit fractals plotted by Saturn consist of the visible parts of the orbits plotted for every row and column in the calculating area grid. The size of the calculating area can be up to 1000 times the display area, the number of columns and rows in the calculating can also be increased by altering the density.

4.2.1 Single Orbit

Single orbit fractals plot a single orbit with a long orbit length. The appearance of the pictures is depending on the starting position often the result is just a blank image or just a few dots endlessly revisited. The fractals can not be used to produce very resolution imges as they fade away as the resolution is increased which is which they are supported by Titan.

4.3 Lyapunov

The area plotted for the Lyapunov fractal is a rectangle of length a and height b, it is divided up into a grid of columns and rows . The formula used for the Lyapunov fractal is:

$x \leftarrow rx(1-x)$

The initial of x is 0.5, the value of r is dependent on the position of in the grid and on the "sequence", the sequence is a string of As and Bs where there is a least one A and one B. For a given location (a,b) the first evaluation of the formula sets r to the location of the column on the a axis if the first letter in the sequence A or it sets r to the location of the row on the b axis if the letter is B, the next evaluation of the formula checks the next letter in the sequence and uses either the a axis or b axis location depending on the letter, calculation continues using the next letter in the sequence, if the end of the sequence has been reached then the sequence is restarted. This is known as a forced cycllic system.

A settling period is allowed for so that the behaviour of the system settles down to a particular value or becomes chaotic. The length of the settling period is determined in Saturn by the number of settling cycles, i.e. the number of times a sequence has been completed. Once the value has been allowed to settle the Lyapunov co-efficient can be calculated using the values calculated during the calculating cycles.

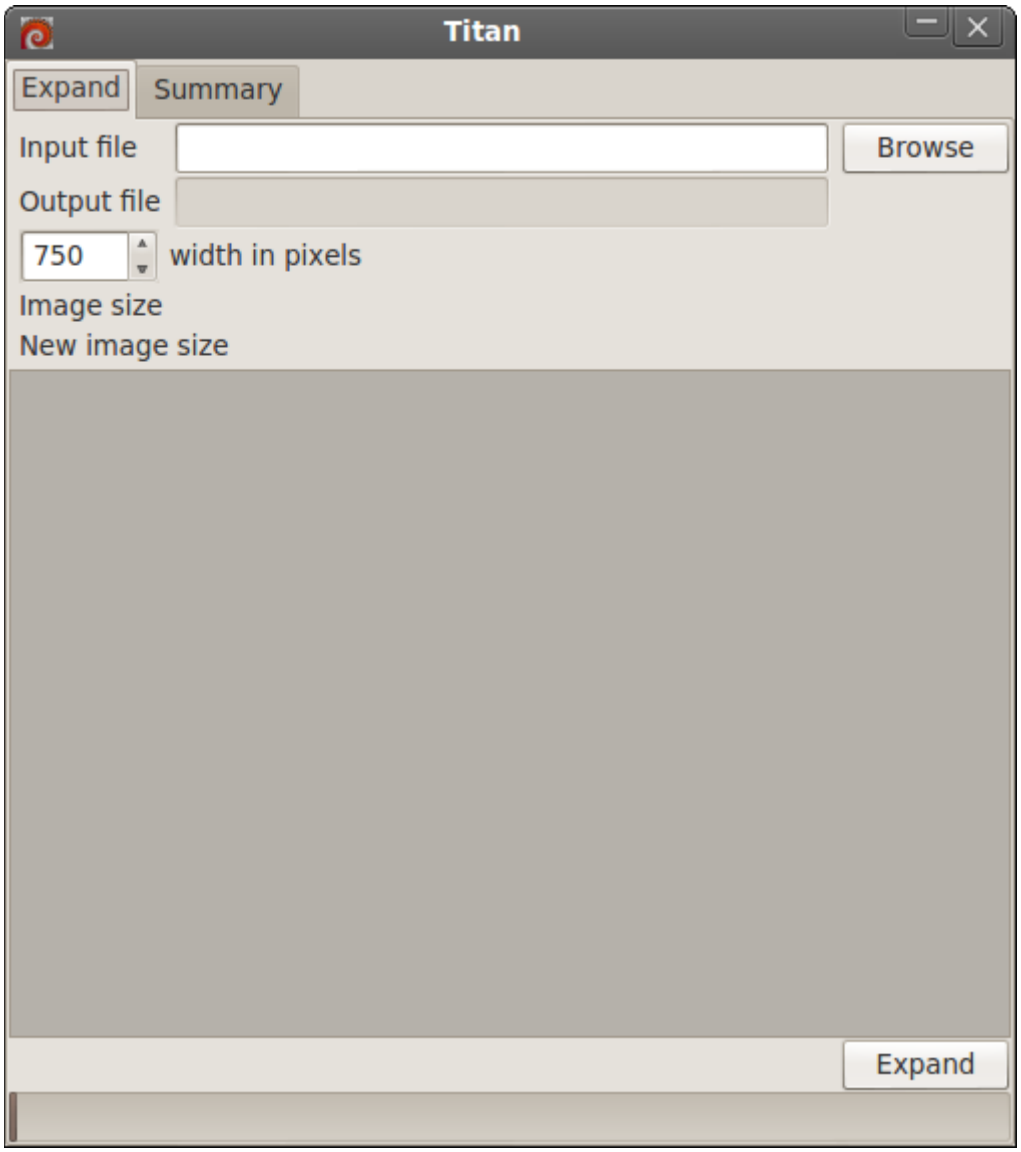
The Lyapunov co-effecient is calculated for every point in the grid and is coloured depending on whether the values are negative or positive. The system is stable if the value of the co-efficient is negative otherwise it is chaotic.

5 Titan

Titan is a fractal expander and is the sister program to Saturn. The size of the pictures that can be saved by Saturn is restricted, however, seed files have all the necessary parameters for recreating the image, these file can be read by Titan and much larger images can be created.

That's is Titan's job in a nutshell, without Saturn it's useless.

This is what Titan looks like when it is first started.



The area bounded by the the horizontal lines separating the width in pixels and the expand button is the display area, once a file has been loaded an image is displayed and it will scaled and scroll bars may appear.

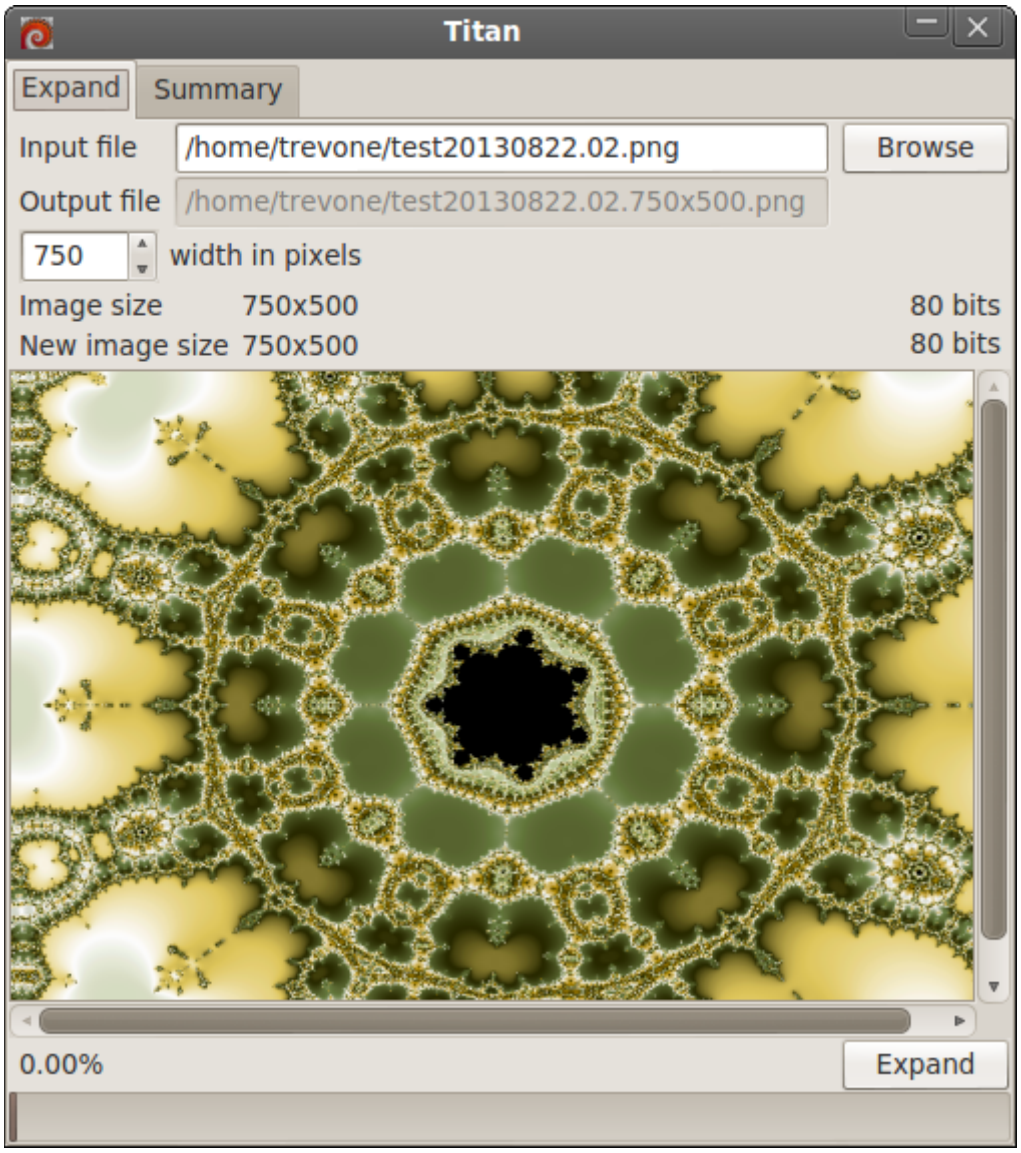
5.1 Loading a Seed File

A seed file can be loaded either by using the browse button or by dragging a seed file and dropping it on the display area. The input file entry box is used to display the input file name and can not be edited.

When a seed file has been successfully loaded its image will be displayed in the display area, its size and the expansion size are displayed along with input and output file names.

The output file name is automatically generated by inserting the size of the output image before the extension ".png".

Titan with a seed file loaded:



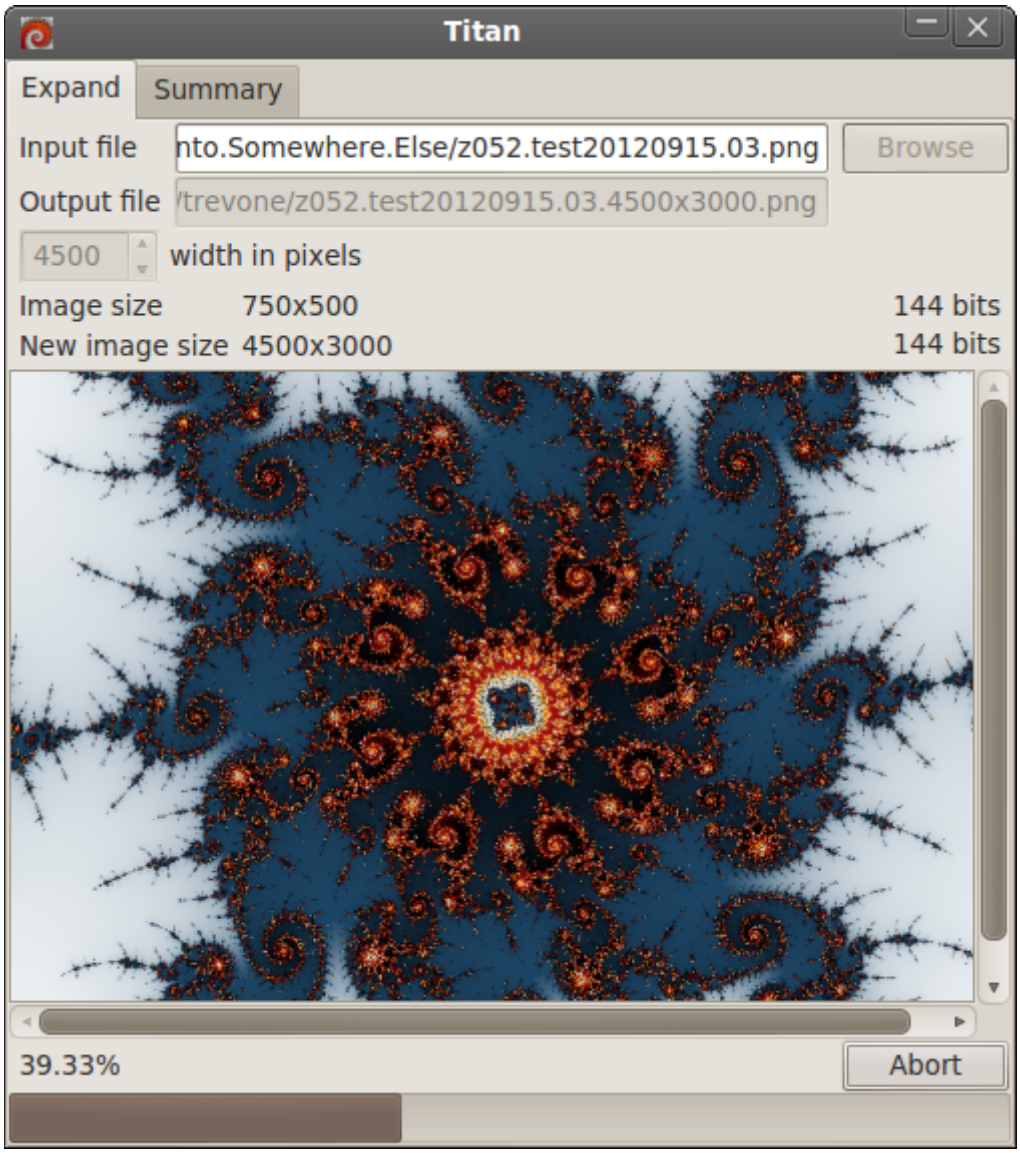
Once the seed file has been loaded the size of the required image can be set by setting the width in pixels. Number of precision bits required to calculate the image is display for the original sized image and for the new size.

Example seed files are available for download at http://element90.wordpress.com.

5.2 Expanding the Image

To expand to image press the “Expand” button. Expansion will be started and progress shown in the progress bar. The “Expand” button turns into the “Abort” button which can be used to abandon the expansion.

Titan expanding an image:



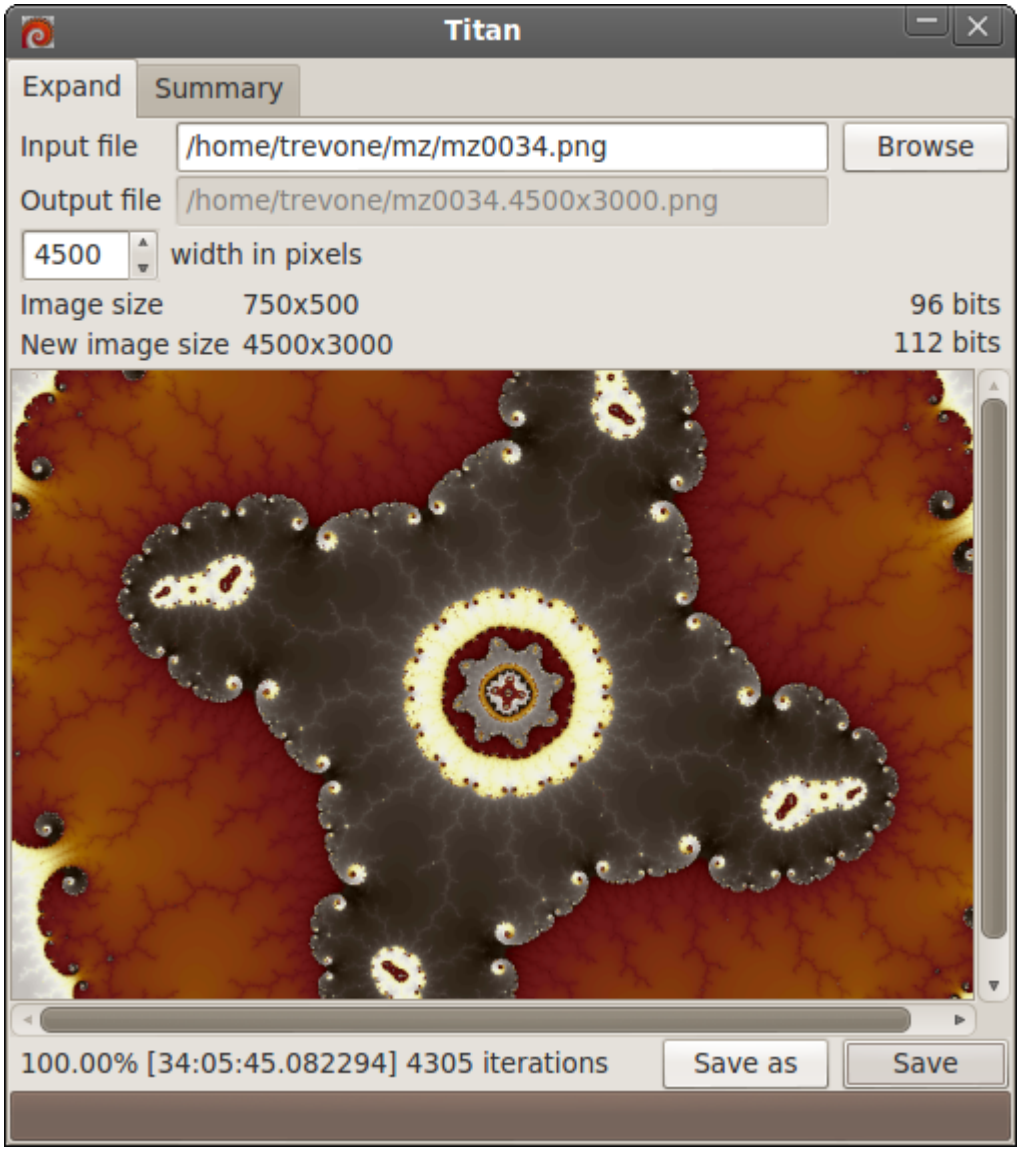
If the “Abort” button is pressed the button reverts to “Expand”, this doesn't happen immediately as the check for abort happens only after a calculation “work set” has completed.

While expansion is in progress loading seed files and changing the output image size are disabled.

5.3 Saving the Image

When the expansion has completed two buttons are displayed, “Save” and “Save as”. Pressing the “Save” will save the newly generated image using the automatically generated file name. Pressing the “Save as” leads to a dialogue where the location and file name of the newly generated image can be specified. If the file of the same name already exists a dialogue message will be displayed and you will not be able to save the image, use “Save as” again with the name of a file that doesn't already exist.

Titan ready to save the expanded image.



Once the file has been saved the “Save” and “Save as” buttons will be replaced with the “Expand” button.

When expansion has completed loading of seed files and changing the output image size are enabled. ***CAUTION - Loading a seed file or changing the output image before saving the image will cause the expansion to be lost as the buttons will be replaced with the “Expand” button.***

The final image should be examined at full resolution as it is possible that the required precision

wasn't correctly determined. If the image is clearly blocky or fuzzy then the only rememdy is to load the seed file back into Saturn so that the precision can be set manually, a new seed file should be saved which can then be used by Titan to produce a higher quality picture.

5.4 Expansion Time

The time taken to expand a fractal can be considerable, for a rough estimate of the time required to complete the expansion it is best to determine how long Titan takes to produces a fractal image that is the same size as the image produced by Saturn. When the input file has been loaded just press the expand button, when the fractal is complete the duration will be displayed above the progress bar. To find the rough duration for the expansion: find the expansion factor and multiply by the duration.

expansion factor = new image size / image size

So to expand a 750x500 image to 7500x5000 image the expansion factor is 100, to expand it to 12000x8000 the expansion factor is 250.

For a duration of 10 seconds the rough expansion times would be 16 minutes 40 seconds for a 7500x5000 and 42 minutes 40 seconds for a 12000x8000 image.

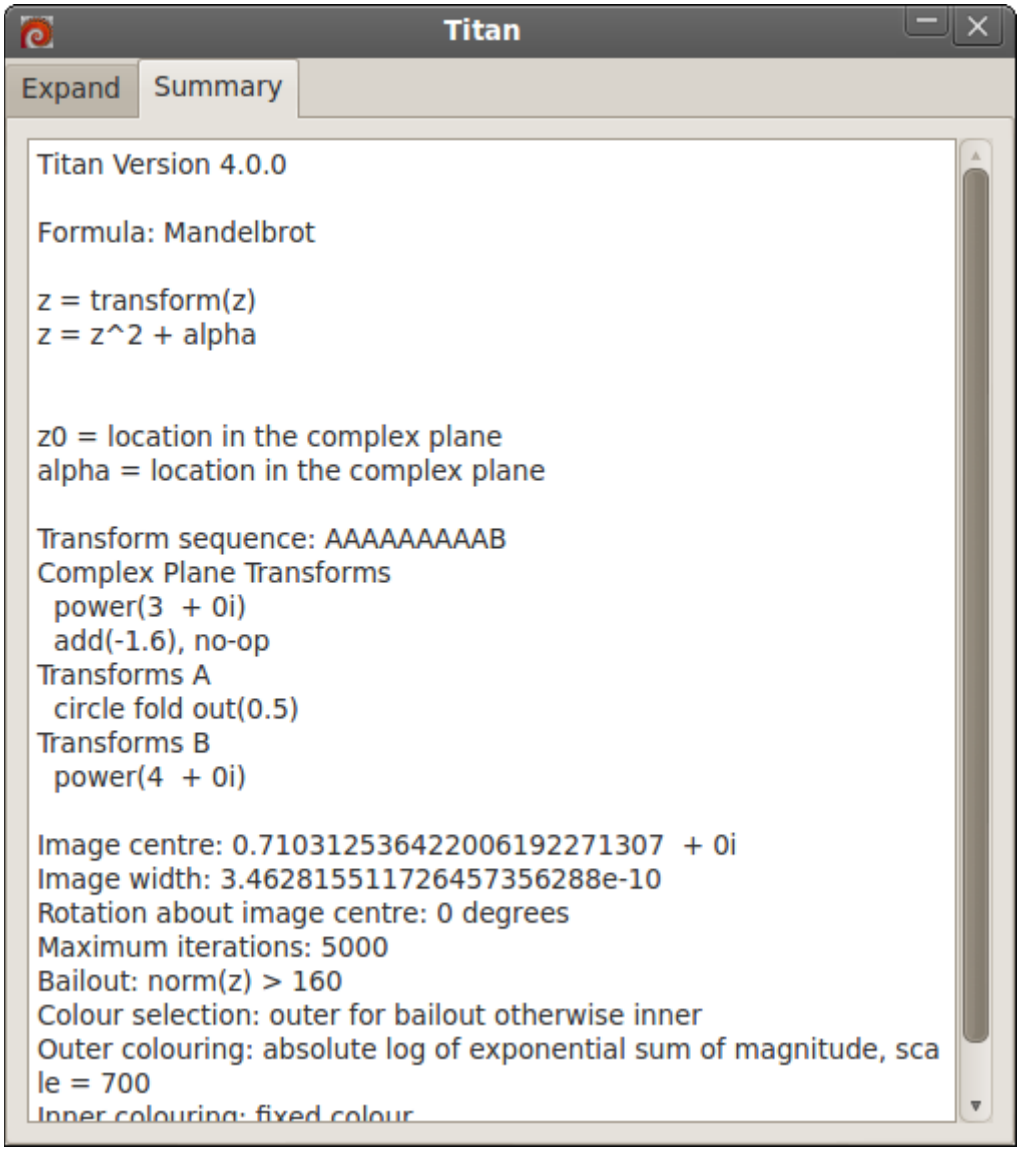
If the the number of bits for the new image is greater than that for the original image expansion will be greatly increased.

5.5 Memory Use

The maximum size of an expanded image is around 700 megapixels. Escape time fractals and the Lyapunov fractals can be calculated on a point by point basis so only memory for the picture buffer is of any great size. Orbit fractals require a working buffer with up to 16 bytes per pixel for each pixel in the final image, so the memory requirement can be very large. If the required memory can not be allocated a message will be displayed saying so.

5.6 The Summary Tab

The summary tab displays a summary of the parameters used to generate the fractal image. The summary is displayed in a text box so it can copied and pasted as required.



5.7 Post Titan Processing

The files produced by Titan are PNG files, they can easily be loaded into other image manipulation programs for adding:

1. Text such as a title, creator's name or copyright notice.
2. Rotation from landscape to portrait.
3. Conversion to a different format such as JPEG.

Neither Saturn or Titan provide for anti-aliasing: to produce a higher quality image for a given picture size, a much larger image can be produced and then scaled down to the required size using an image manipulation program, the program you use should have a quality option such as interpolation, GIMP provides, Linear, Cubic and Lancz03.

6 Appendix – Notes on Fractal Formulae

The fractal formula is displayed in the fractal tab of the fractal settings window, however for some of the fractals the actual formula is different.

6.1 Almost Cubic

The formula for Almost Cubic is an erroneous implementation of the formula:

$z \leftarrow \alpha z^3 + \beta z^2\, \gamma z + \delta$

implemented as

$z_r \leftarrow (z_r^3 - 3z_r z_i^2)\alpha_r - (3z_r^2\, z_i - z_i^3)\alpha_i + (z_r^2 - z_i^2)\beta_r - 2z_r z_i \beta_i + z_r \gamma_r - z_r \gamma_i + \delta_r$

$z_i \leftarrow (3z_r^3 - z_r z_i^2)\alpha_i + (3z_r^2 z_i - z_r 3)\alpha_r + (z_r^2 - z_i^2)\beta_i + 2z_r z_i \alpha_r + z_r \gamma_i + z_i \gamma_r + \delta_i$

The error is in the assignment of z_i the term

$2z_r z_i \alpha_r$

should be

$2z_r z_i \beta_r$

6.2 Bad Complex Power Functions

The calculation of the complex power of a complex number using pairs of real numbers instead of using C++ language's standard library complex type and pow function is a complicated affair involving the conversion of complex numbers in cartesian form into polar form calculating the complex power and converting back from polar form into cartesian form.

The reason this was done was an attempt to use the graphics processor to perform fractal calculation using OpenCL routines which do not support complex number types or functions. Although the results using OpenCL were impressive in terms of the speed of calculation, the only real number type available was single precision (float) which resticts the depth of zoom into a fractal picture. The resources on the graphics card used was restricted and there were insufficient resources to support the implementation of the more complex fractal types. This was attempted in Mars and subsequently abandoned, Mars has since been superceded by Saturn. It was during this development phase that the Bad Complex Power fractals were discovered.

6.3 Bad Complex Power Fractals

When implementing the calculation of the complex power of a complex number I made made a mistake in the conversion of the complex number into polar form before performing the power function.

This is a complex number in cartesian form:

$$z = a + bi$$

where a is the real component and b is the imaginary component.

And this is a complex number in polar form:

$$z = re^{\theta i}$$

where r is the magnitude and θ is the argument.

Conversion of z into polar form:

$$r = \sqrt{(a^2 + b^2)}$$
$$\theta = \arctan(b/a)$$

The value of θ is problematic as the value obtained when both a and b are negative is the same as when both a are positive when in fact it is pi radians away, similarly -a and +b gives the same as +a + -b and again there is pi radians difference. This is where the bad complex power functions come in:

bcp – if a is negative add pi to θ
bcp2 – no adjustment to θ

The correct adjustment would subtract pi radians if a and b are negative and add pi radians if a is negative and b is positive.

The formulae used for Bad Complex Power and Bad Complex Power Julia fractal are:

$$z = bcp(z, z) + c$$

$z = bcp(z, z) + \alpha$

The formulae used for Bad Complex Power 2 and Bad Complex Power Julia 2 fractal is:

$z = bcp2(z, z) + c$
$z = bcp2(z, z) + \alpha$

The steps to perform the calculation of a complex number to a complex power has been omitted as the error occurs at the first step, i.e. the conversion of the complex number into polar form.

Note: in C++ if the correct function had been used i.e. atan2(b, a) of atan(b/a) the bcp functions would never have existed.

6.4 Pickover Popcorn Formulae

The formulae for Pickover Popcorn are misleading, for example the four function Julia form:

$z.r = z.r - A*f1(z.i + f2(B*z.i)) + \alpha.r$
$z.i = z.i - C*f3(z.r + f4(D*z.r)) + \alpha.i$

imply that the z.r used in the second formula is the result of the first formula this is not the case. The formulae are more correctly written:

$z.r_{n+1} = z.r_n - A*f1(z.i_n + f2(B*z.i_n)) + \alpha.r$
$z.i_{n+1} = z.i_n - C*f3(z.r_n + f4(D*z.r_n)) + \alpha.i$

7 Appendix - Transforms

The basic transforms apply to the iterating variable in escape time fractals and is referred to in this section as z, many of the transforms are complex functions which may have parameters.

Note: old transforms used in seed files saved using versions of Saturn prior to version 3.0.0 are automatically converted to the sets od complex functions.

7.1 Old Transforms to Complex Functions

Prior to version 3.0.0 transforms and complex functions were defined separately. Transforms are now simply complex functions. Complex functions from version 3.0.0 were revised to support various some of the features provided by the old transforms such as circle fold out, complex function can now be treated as a pair of functions which allows the transforms such as top left, top right etc. to be defined.

7.1.1 Translation

This transform simply adds a complex number t to z. Can be replaced with “add(t.r), add(t.i)” where a component is zero “add(t.r), no-op” and “no-op, add(t.i)” can be used.

7.1.2 Rotation

This transform rotates the value z about the origin r degrees. Replaced with “rotation(r)”.

7.1.3 Scale

This transform multiples the component values of z by the component values of t, i.e.

$z \leftarrow z.r*t.r + (z.i*t.i)*i$

Replaced with “scale(t.r), scale(t.i)” where the scale factor for a component is 1 then “scale(t.r), no-op” and “no-op, scale(t.i)” can be used.

7.1.4 Unsign Real

This transform forces the real component of z to be positive. This is folds values on the left hand side of the vertical or imaginary axis over to the right hand side of the axis. The folding line can be moved from the imaginary axis using t.r and can be moved from the vertical by rotation r.

z ← z – t.r
rotate -r degrees
z ← abs(z.r) + z.i*i
rotate r degrees
z ← z + t.r

This becomes a set of up to 5 complex functions:

add(-t.r), no-op
rotation(-r)
abs, no-op
rotation(r)
add(t.r), no-op

If t.r is 0 then the translation functions can be omitted and if r is also zero the rotation functions can be omitted.

7.1.5 Sign Real

This transform forces the real component of z to be negative. This folds values on the right hand side of the vertical or imaginary axis over to the left hand side of the axis. The folding line can be moved from the imaginary axis using t.r and can be moved from the vertical by rotation r.

z ← z – t.r
rotate -r degrees
z ← -abs(z.r) + z.i*i
rotate r degrees
z ← z + t.r

This becomes a set of up to 5 complex functions:

add(-t.r), no-op
rotation(-r)
-abs, no-op
rotation(r)
add(t.r), no-op

If t.r is 0 then the translation functions can be omitted and if r is also zero the rotation functions can be omitted.

7.1.6 Reverse Sign Real

This transform reverses sign of the the real component. This reflects values on the right hand side of the vertical or imaginary axis over to the left hand side of the axis and vice versa. The reflection line can be moved from the imaginary axis using t.r and can be moved from the vertical by rotation.

z ← z – t
rotate -r degrees
z ← -z.r + z.i*i
rotate r degrees
z ← z + t

This becomes a set of up to 5 complex functions:

add(-t.r), no-op
rotation(-r)
-/+, no-op
rotation(r)
add(t.r), no-op

If t.r is 0 then the translation functions can be omitted and if r is also zero the rotation functions can be omitted.

7.1.7 Unsign Imaginary

This transform forces the imaginary component of z to be positive. This folds values below the horizontal or real axis above the axis. The folding line can be moved from the real axis using t.r and can be moved from the vertical by rotation.

z ← z – t.i
rotate -r degrees
z ← z.r + abs(z.i)*i
rotate r degrees
z ← z + t.i

This becomes a set of up to 5 complex functions:

no-op, add(-t.i)
rotation(-r)
no-op, abs
rotation(r)
no-op, add(t.i)

If t.i is 0 then the translation functions can be omitted and if r is also zero the rotation functions can be omitted.

7.1.8 Sign Imaginary

This transform forces the imaginary component of z to be negative. This folds values above the horizontal or real axis below the axis. The folding line can be moved from the real axis using t.r and can be moved from the vertical by rotation.

z ← z – t.i
rotate -r degrees
z ← z.r - abs(z.i)*i
rotate r degrees
z ← z + t.i

This becomes a set of up to 5 complex functions:

no-op, add(-t.i)
rotation(-r)
no-op, -abs
rotation(r)
no-op, add(t.i)

If t.i is 0 then the translation functions can be omitted and if r is also zero the rotation functions can be omitted.

7.1.9 Reverse Sign Imaginary

This transform reverses sign of the the real component. This reflects values above the horizontal or real axis below the axis and vice versa. The reflection line can be moved from the real axis using t.r and can be moved from the vertical by rotation.

z ← z – t
rotate -r degrees
z ← z.r - z.i*i
rotate r degrees
z ← z + t

This becomes a set of up to 5 complex functions:

no-op, add(-t.i)
rotation(-r)
no-op, +/-
rotation(r)
no-op, add(t.i)

If t.i is 0 then the translation functions can be omitted and if r is also zero the rotation functions can be omitted.

7.1.10 Circle Fold In

This transform maps points outside a circle to points inside inside the circle. The size of the circle is specified by the radius r and the position is specified by t.

z ← z – t
if abs(z) > r then z ← (r*r*z)/norm(z)
z ← z + t

This becomes a set of up to 3 complex functions:

add(-t.r), add(-t.i)
circle fold in(2r)
add(t.r), add(t.i)

If t is 0 then the translation functions can be omitted.

7.1.11 Circle Fold Out

This transform maps points inside a circle to points outside the circle. The size of the circle is specified by the radius r and the position is specified by t.

z ← z – t
if abs(z) < r then z ← (r*r*z)/norm(z)
z ← z + t

This becomes a set of up to 3 complex functions:

add(-t.r), add(-t.i)
circle fold out(2r)
add(t.r), add(t.i)

If t is 0 then the translation functions can be omitted.

7.1.12 Circle Reflect

This transform maps points outside a circle inside the circle and vice versa. The size of the circle is specified by the radius r and the position is specified by t.

z ← z – t
z ← (r*r*z)/norm(z)
z ← z + t

This becomes a set of up to 3 complex functions:

add(-t.r), add(-t.i)
circle fold reflect(2r)
add(t.r), add(t.i)

If t is 0 then the translation functions can be omitted.

7.1.13 Top Right

This transform maps points outside the top right quadrant into the top right quadrant, this is done by folding points to the left of the imaginary axis over to the right side of the axis and then folding the points below the real axis over to above the axis.

The position of these folds can be moved and rotated.

z ← z – t
rotate -r degrees
z ← abs(z.r) + abs(z.i)*i
rotate r degrees
z ← z + t

This becomes a set of up to 5 complex functions:

add(-t.r), add(-t.i)
rotation(-r)
abs, abs
rotation(r)
add(t.i), add(t.i)

If t is 0 then the translation functions can be omitted and if r is also zero the rotation functions can be omitted.

7.1.14 Bottom Right

This transform maps points outside the bottom right quadrant into the top right quadrant, this is done by folding points to the left of the imaginary axis over to the right side of the axis and then folding the points above the real axis over to below the axis.

The position of these folds can be moved and rotated.

z ← z – t
rotate -r degrees
z ← abs(z.r) - abs(z.i)*i
rotate r degrees
z ← z + t

This becomes a set of up to 5 complex functions:

add(-t.r), add(-t.i)
rotation(-r)
abs, -abs
rotation(r)
add(t.i), add(t.i)

If t is 0 then the translation functions can be omitted and if r is also zero the rotation functions can be omitted.

7.1.15 Top Left

This transform maps points outside the top left quadrant into the top right quadrant, this is done by folding points to the right of the imaginary axis over to the left side of the axis and then folding the points below the real axis over to above the axis.

The position of these folds can be moved and rotated.

z ← z – t
rotate -r degrees
z ← -abs(z.r) + abs(z.i)*i
rotate r degrees
z ← z + t

This becomes a set of up to 5 complex functions:

add(-t.r), add(-t.i)
rotation(-r)
-abs, abs
rotation(r)
add(t.i), add(t.i)

If t is 0 then the translation functions can be omitted and if r is also zero the rotation functions can be omitted.

7.1.16 Bottom Left

This transform maps points outside the bottom quadrant into the top right quadrant, this is done by folding points to the right of the imaginary axis over to the left side of the axis and then folding the points above the real axis over to below the axis.

The position of these folds can be moved and rotated.

z ← z – t
rotate -r degrees
z ← -abs(z.r) - abs(z.i)*i
rotate r degrees
z ← z + t

This becomes a set of up to 5 complex functions:

add(-t.r), add(-t.i)
rotation(-r)
-abs, -abs
rotation(r)
add(t.i), add(t.i)

If t is 0 then the translation functions can be omitted and if r is also zero the rotation functions can be omitted.

7.1.17 Inverse Fold In

This transform maps points outside a circle to points inside inside the circle. The size of the circle is specified by the radius r and the position is specified by t.

$z \leftarrow z - t$
if abs(z) > r then $z \leftarrow (r*r)/z$
$z \leftarrow z + t$

This becomes a set of up to 3 complex functions:

add(-t.r), add(-t.i)
inverse fold in(2r)
add(t.r), add(t.i)

If t is 0 then the translation functions can be omitted.

7.1.18 Inverse Fold Out

This transform maps points inside a circle to points inside outside the circle. The size of the circle is specified by the radius r and the position is specified by t.

$z \leftarrow z - t$
if abs(z) < r then $z \leftarrow (r*r)/z$
$z \leftarrow z + t$

This becomes a set of up to 3 complex functions:

add(-t.r), add(-t.i)
inverse fold out(2r)
add(t.r), add(t.i)

If t is 0 then the translation functions can be omitted.

7.1.19 Inverse Reflect

This transform maps points outside a circle to points inside inside the circle and vice versa. The size of the circle is specified by the radius r and the position is specified by t.

$z \leftarrow z - t$
$z \leftarrow (r*r)/z$
$z \leftarrow z + t$

This becomes a set of up to 3 complex functions:

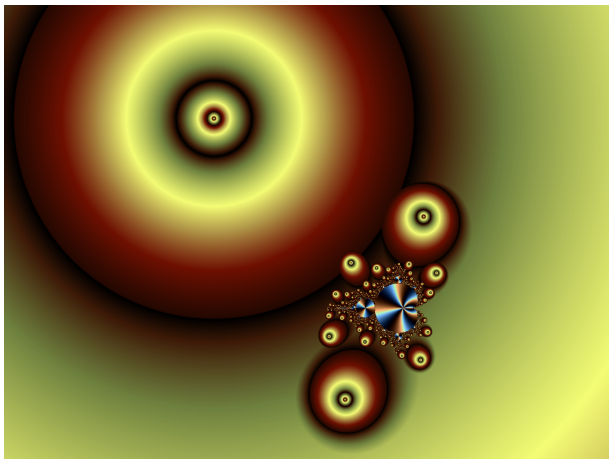
add(-t.r), add(-t.i)
inverse reflect(2r)
add(t.r), add(t.i)

If t is 0 then the translation functions can be omitted.

8 Appendix 11 – Orbit Trap Types

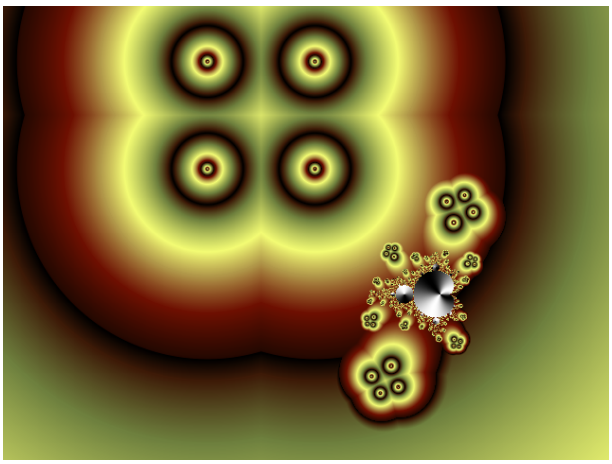
This appendix shows what the orbit trap shapes look like, some are obvious others are not for completeness pictures of all the orbit trap shapes are shown with any optional variations.

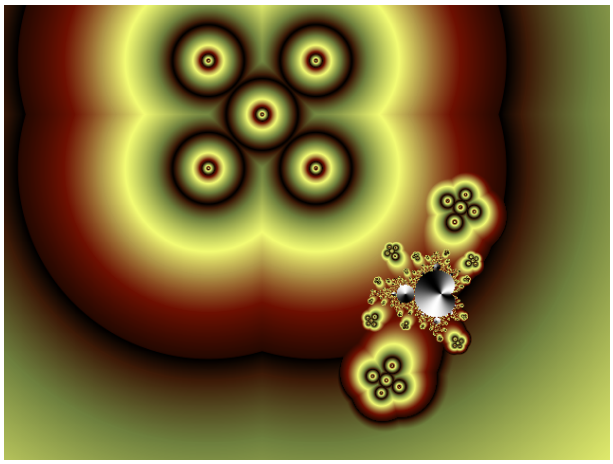
8.1 Point



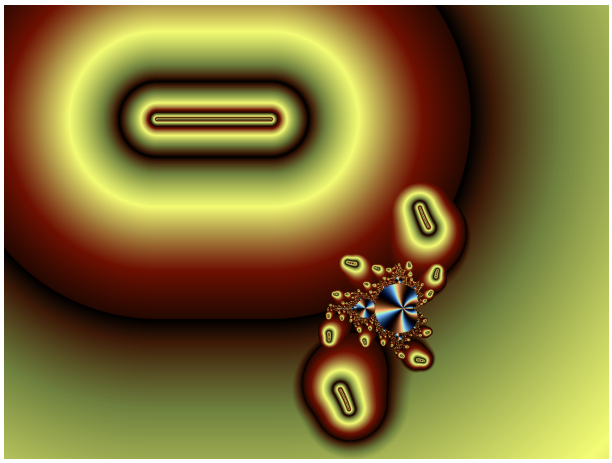
8.2 Four Points

Four point orbit trap, without and without central point.

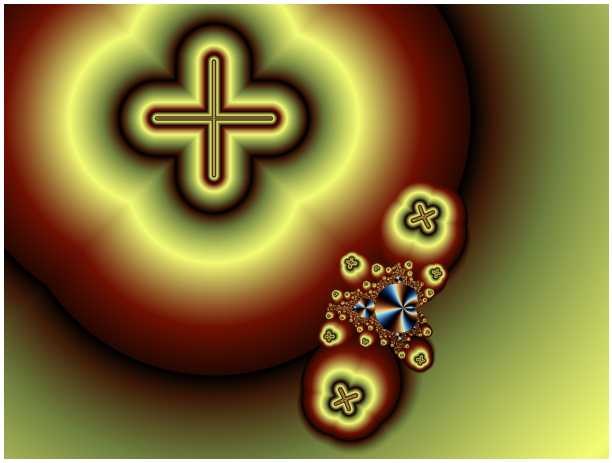




8.3 Line

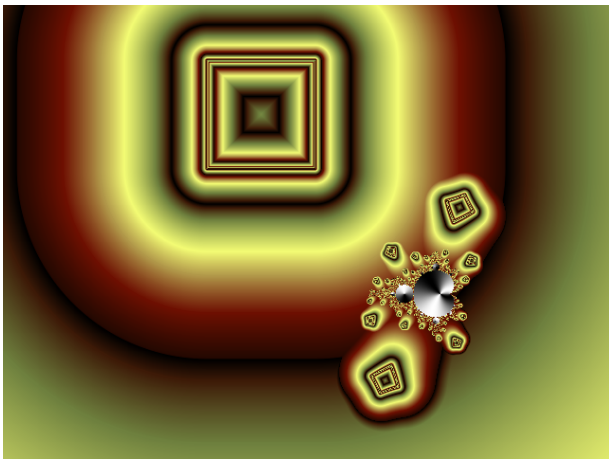


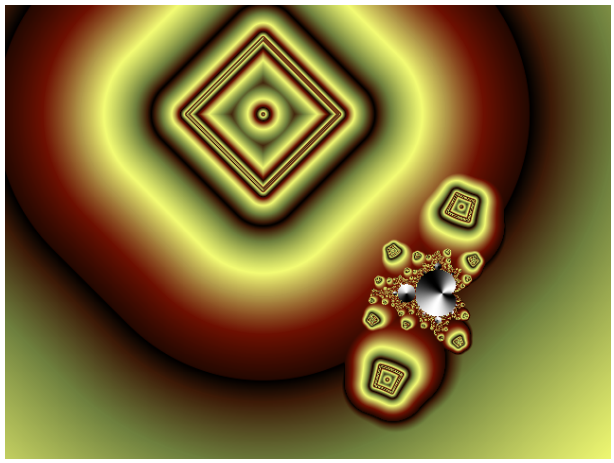
8.4 Cross



8.5 Square

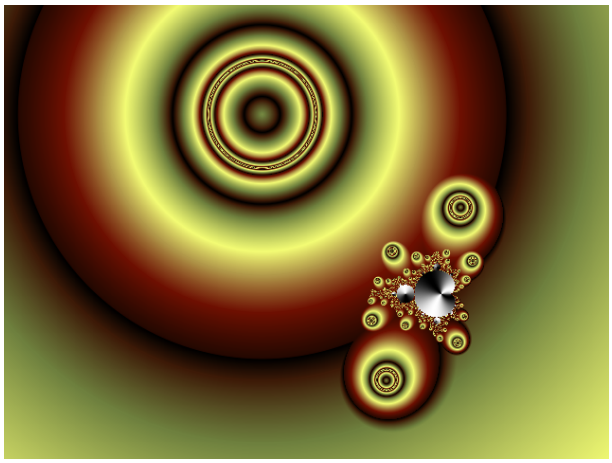
Square orbit trap, without the central point and with central point and rotated 45 degrees.

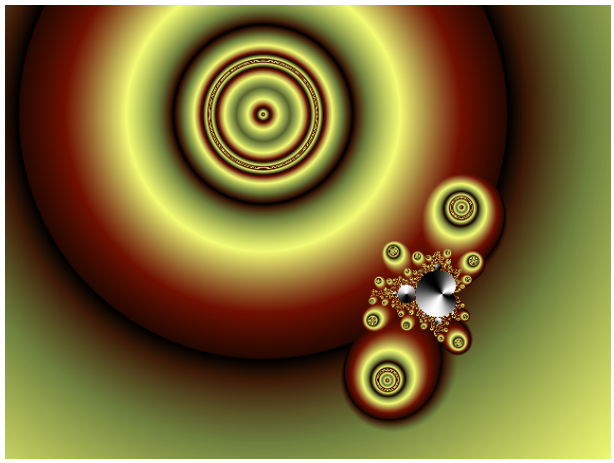




8.6 Circle

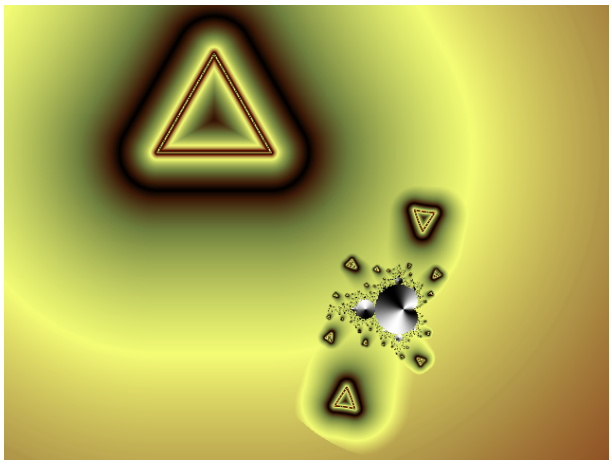
Circle orbit trap without and with central point.

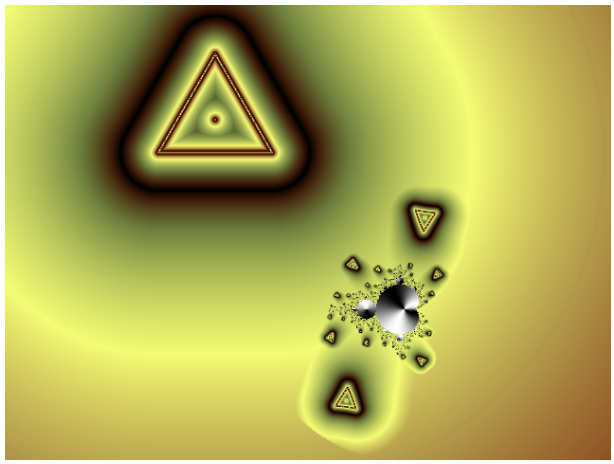




8.7 Triangle

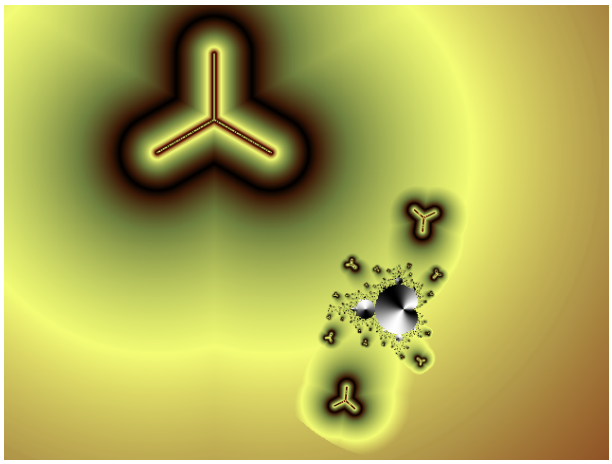
Triangle orbit trap without and with central point.





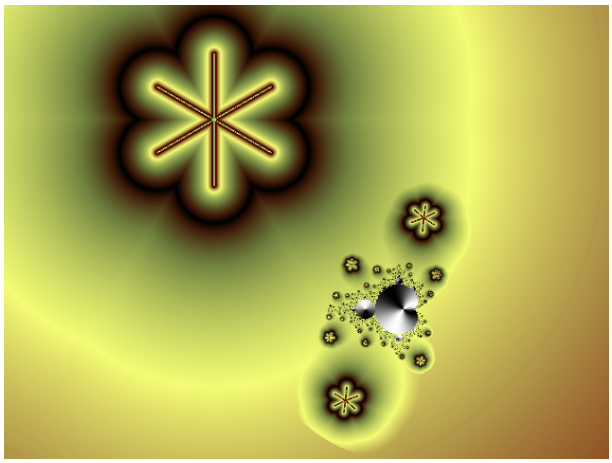
8.8 Triform

Triform orbit trap, no optional central point.

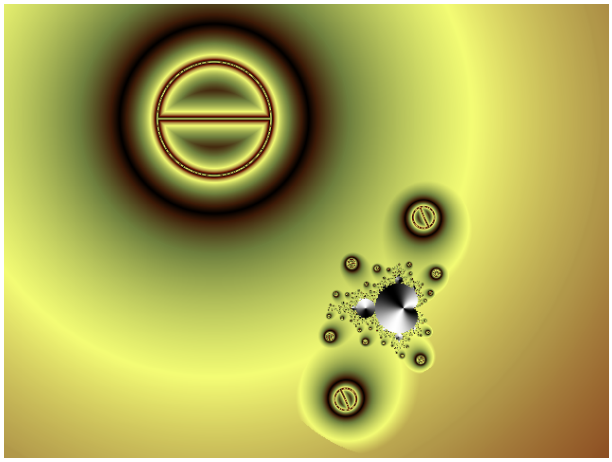


8.9 Asterisk

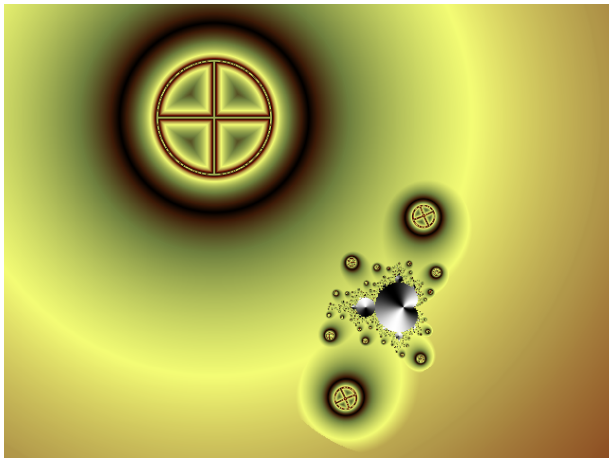
The asterisk orbit trap, again, no optional central point.



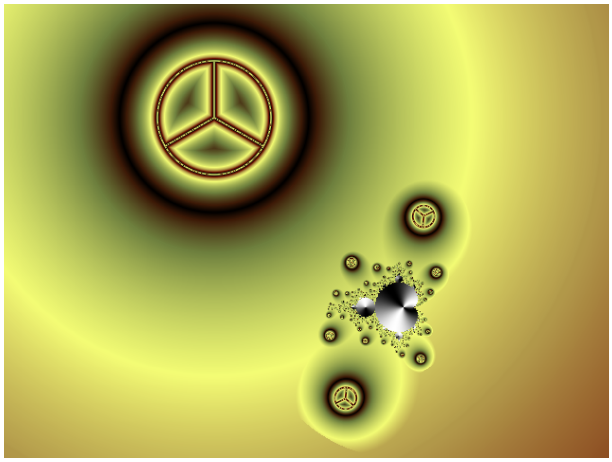
8.10 Circle Line



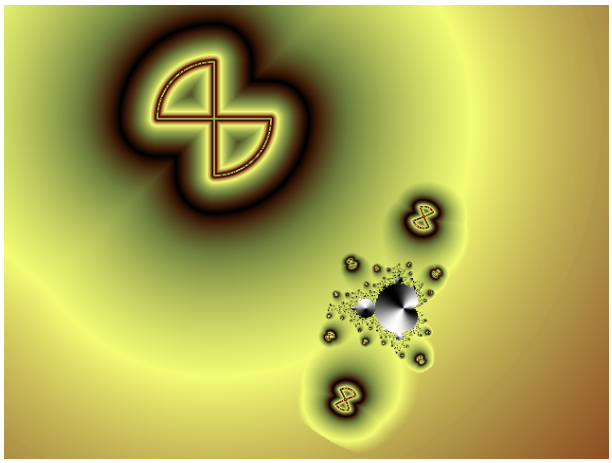
8.11 Circle Cross



8.12 Circle Triform

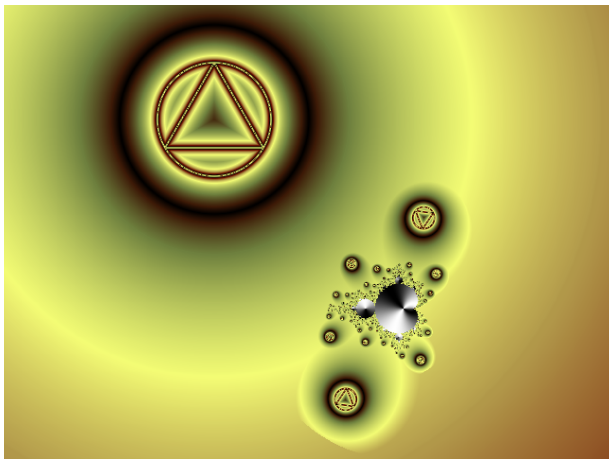


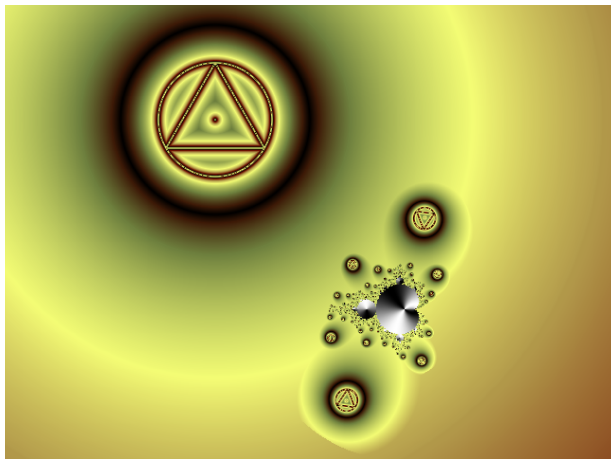
8.13 Two Quarter Circles



8.14 Circle Triangle

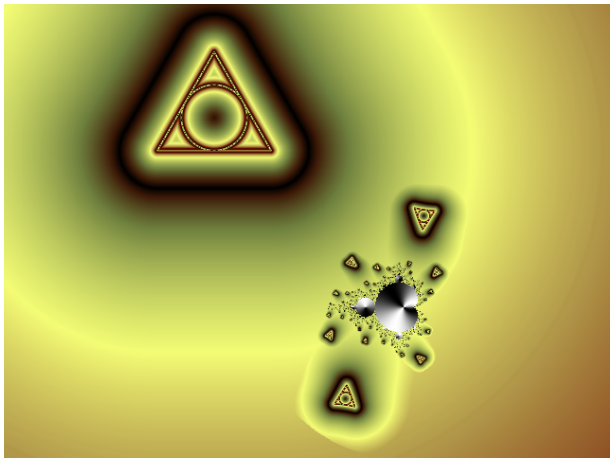
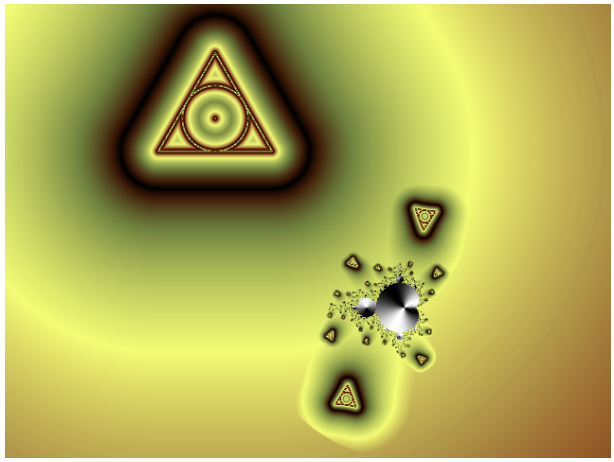
Circle Triangle orbit trap without and with optional central point.





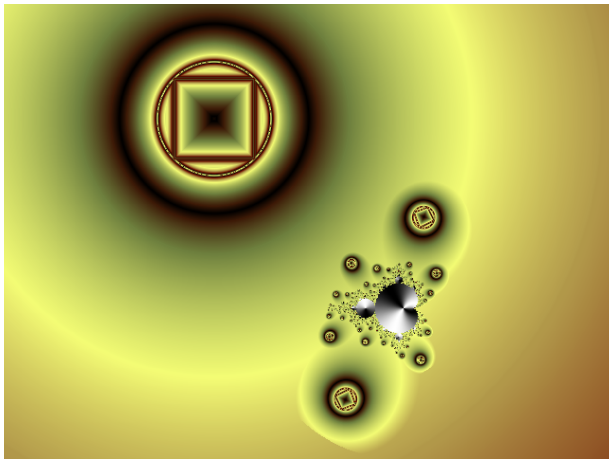
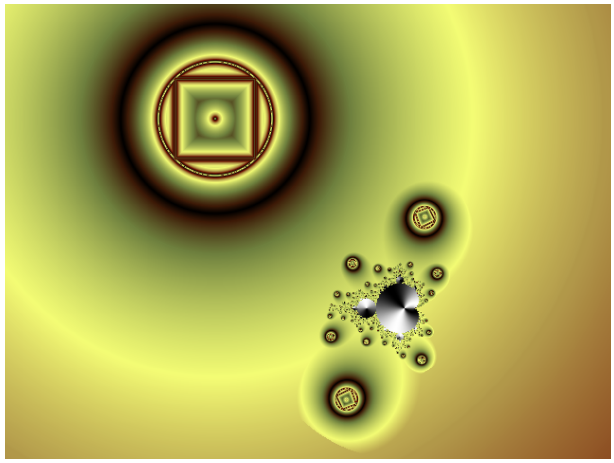
8.15 Triangle Circle

Triangle circle obit trap without and with central point.

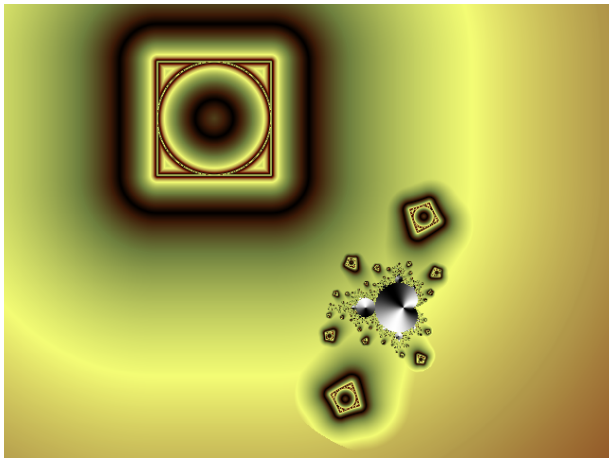
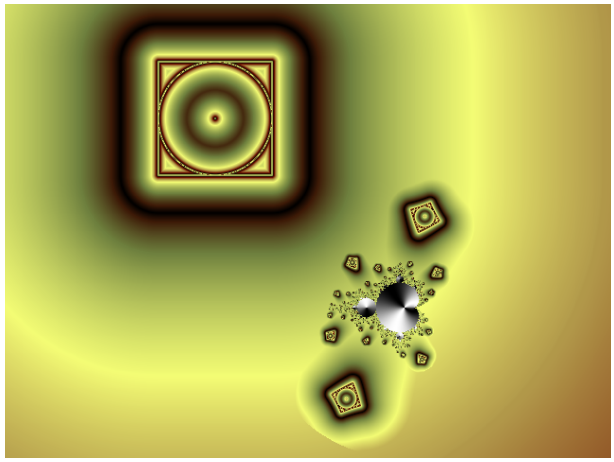
8.16 Circle Square

Circle square orbit trap without and with central point.

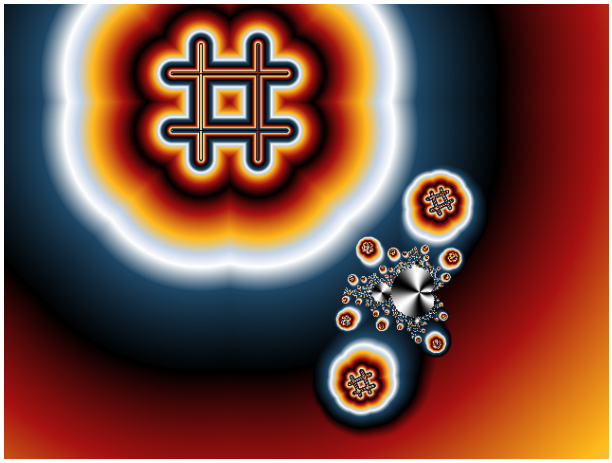
8.17 Square Circle

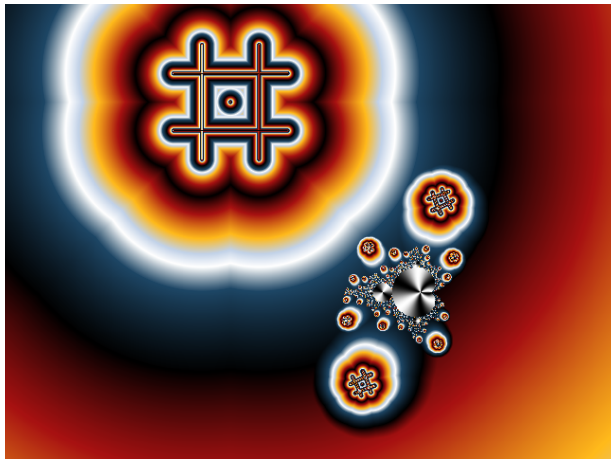
Square circle orbit without and with central point.

8.18 Octothorpe

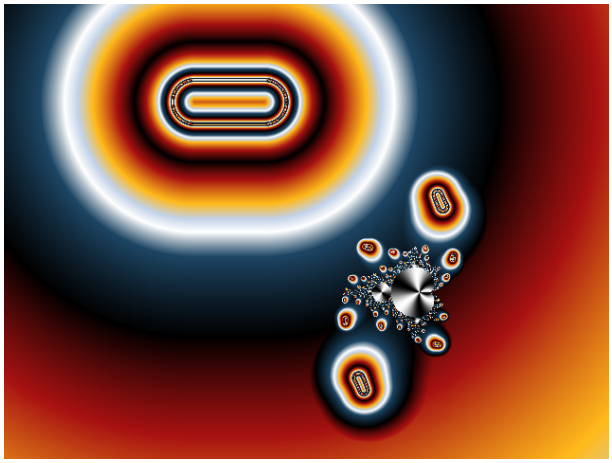
Octothorpe (or pound, hash, sharp) orbit trap without and with central point.

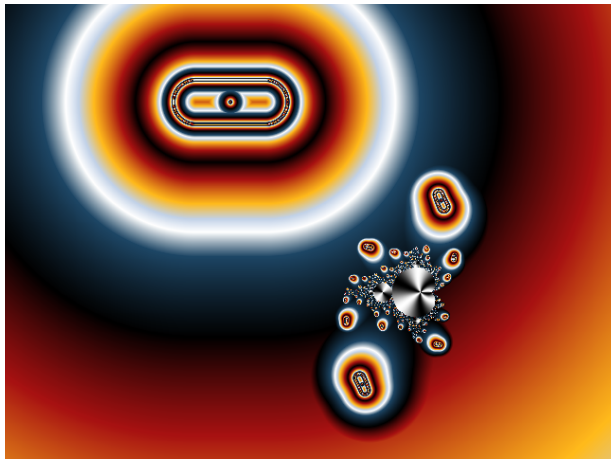




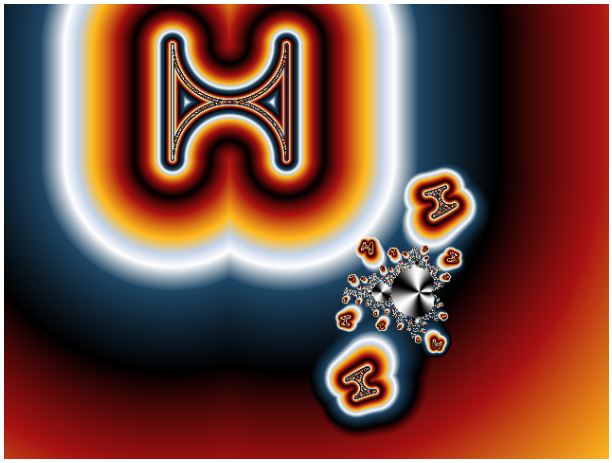
8.19 Running Track

Running track orbit trap without and with central point. The length of the straight lines and the semi-circles are the same.



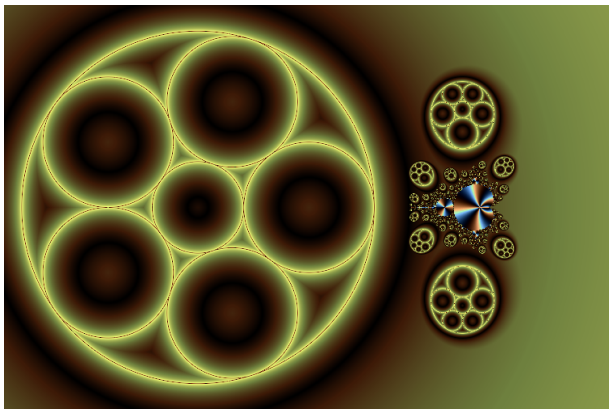
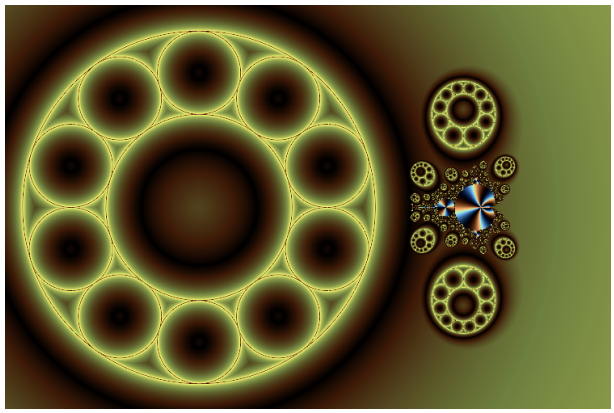


8.20 Pinch

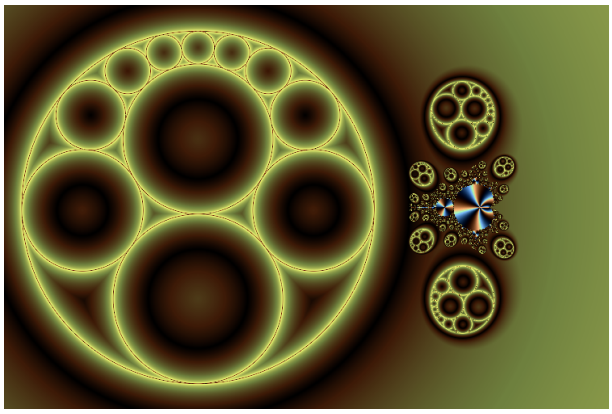
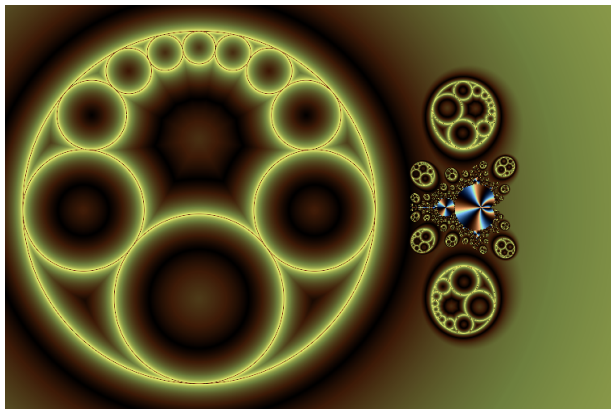


8.21 Steiner Chain

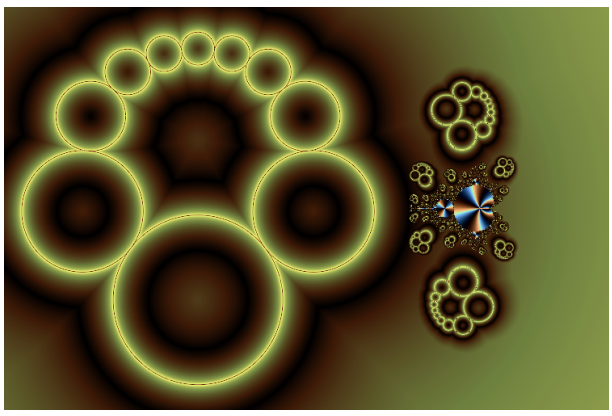
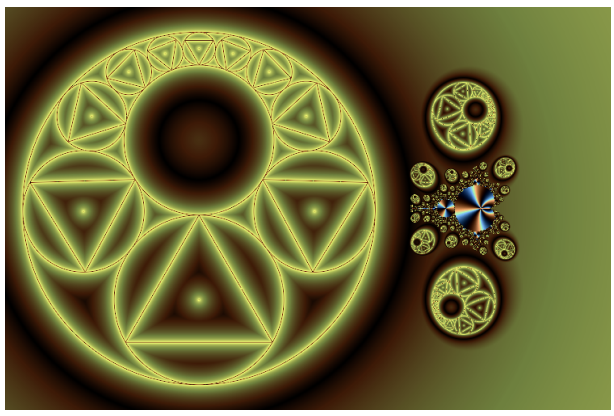
The Steiner Chain used an orbit can be configured so that its appearance can be varied widely.

Two annular Steiner Chains one with a chain length of 5 and the other with a chain length of 10.

Two Steiner Chains of length 10, inversion distance of -0.5 and rotation of 90 degrees, the second one has its inner circle omitted.

The same Steiner Chain tfirstly with both the inner and outer circles omitted and secondly with all options enabled.

9 Appendix - Statistics

Several of the colouring methods collect statistics for use in determining the value to use to select the colour. This appendix contains the statistic formulae for the non-obvious statistic values.

9.1 Range

Simple the difference between the maximum and minimum values.

range = maximum – minimum

9.2 Variance

The variance is calculated from the sum of the results of all the iterations (sum) and the sum of the squares of the results of all the iterations (sum2) and the number of iterations using this formula:

variance = (sum2 – ((sum*sum) ÷ iterations)) ÷ iterations

This is the final stage of the “naïve algorithm” which just requires that sums of the values and squares of the values are accumulated. I've checked the variance obtained using this algorithm on a small set of data and calculated the actual variance and the result using the “naïve algorithm” is just wrong. However as we using the values just to select colour it really doesn't matter.

9.3 Standard Deviation

The standard deviation is the square root of the variance. This uses the the variance obtained using the “naïve algorithm” so its values are also wrong.

9.4 Co-efficient of Variation

The co-efficient of variation is the standard deviation divided by the mean (average). This uses the the variance obtained using the “naïve algorithm” so its values are also wrong.

9.5 Fractal Dimension

The formula I'm using for this is:

fractal dimension = (sum2 – ((sum*sum) ÷ iterations)) ÷ range

I don't know whether this formula means any thing at all, although in all probability it does not calculate the fractal dimension. The sum2 and sum values are the same as for variance.

9.6 Exponential Sum

The exponential sum is an accumulation of the exponential of the result of each iteration, so e^z is added to the running total. The total is used to determine the colour.

9.7 Exponential Inverse Change Sum

The exponential inverse change sum is an accumulation of the negative inverse of the difference in magnitude between the result of the each iteration and the corresponding previous iteration, so if z is the result of an iteration and old_z is the result of the previous iteration then the following is added to the running total:

$$e^{(-1 \div abs(z - old\_z))}$$

the total is used to determine the colour.

10 Appendix – Saturn Memory Use

In order for Saturn to display a full fractal image every few iterations or so and to allow for colour map changes without re-calculation it reserves a significant amount of memory. The memory allocated is dependent on the size of the image and the type of fractal. The example memory allocated is for the largest screen size of 1280x1280.

10.1 Escape Time Fractal

737.5 Megabytes of working memory is allocated. This is very large so the more memory your computer has the better.

10.2 Orbit Fractals

1.5 Megabytes of working memory is allocated. There are three types of or orbit plots an orbit may not be plotted and that can only be determined after the orbit has been calculated so a buffer must be allocated for the orbit, (20 bytes times the orbit length). An orbit buffer is allocated for each thread used to calculate the fractal. On a single processor one thread is used, dual core uses 2, quad core uses 4 etc., the processors may also support 2 hardware threads per core so those vales will be doubled.

10.3 Lyapunov Fractal

About 130 Megabytes is allocated.