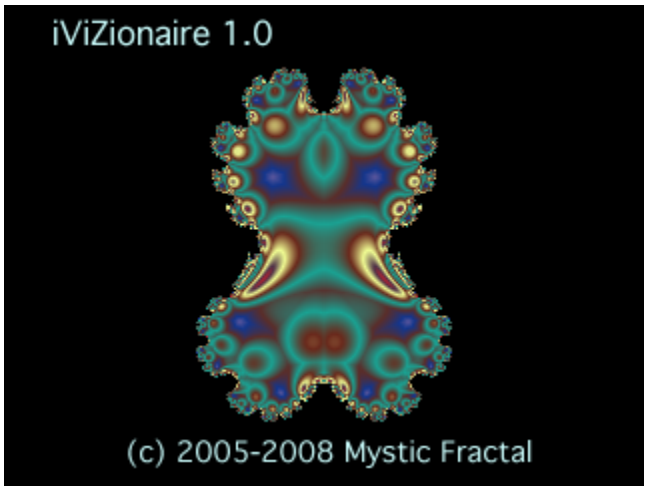
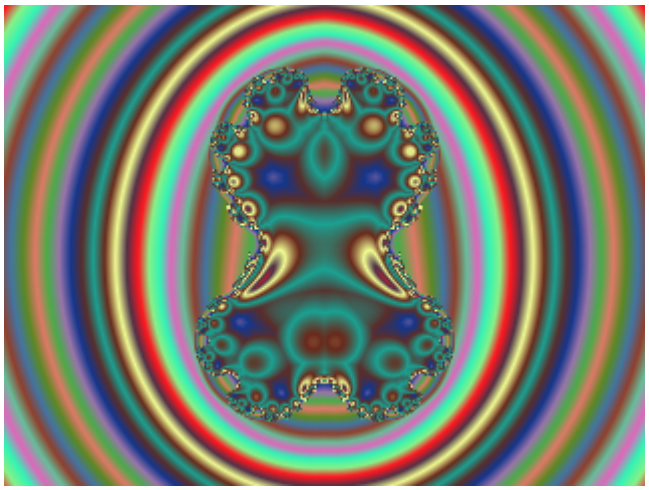


iViZionaire Application Help

Table of Contents

1 Main Index

iViZionaire Help Index



Getting Started

An Introduction To CQuat Fractals by Terry W. Gintz

Menus

Additional

2 File Menu

File menu commands

The File menu offers the following commands:

New	Creates a new drawing.
Open	Opens an existing drawing.
Close	Closes an opened drawing.
Save	Saves an opened drawing using the same file name.
Save As	Saves an opened drawing to a specified file name.
Save Quaternion to [OBJ]	Save polygonized quaternion as Wavefront object.
Save Quaternion to [POV]	Save polygonized quaternion as a POV-Ray triangle object.
Save Quaternion to [STL]	Save polygonized quaternion as STL solid file.

Save Quaternion to [DXF]	Save polygonized quaternion as AutoCad dxf file.
Export Setup	Set maximum number of faces and vertices allocated for quad export.
Load Texture [QTX]	Load texture variables.
Save Texture [QTX]	Save texture variables.
Load Text Format [VZT]	Load a data file in text (platform independent) format.
Save Text Format [VZT]	Save a data file in text (platform independent) format.

2.1 New command

New command (File menu)

Use this command to create a new drawing window in iViZionaire. The image and data file for the opening picture (title.png and title.vse) are used to create the new window.

You can open an existing data/image file with the Open command.

2.2 Open command

Open command (File menu)

Use this command to open an existing data/image file in a new window.

You can create new images with the New command.

2.2.1 File Open dialog box

File Open dialog box

The following options allow you to specify which file to open:

File Name
Type or select the filename you want to open.

Drives
Select the drive in which iViZionaire stores the file that you want to open.

Directories
Select the directory in which iViZionaire stores the file that you want to open.

Network...
Choose this button to connect to a network location, assigning it a new drive letter.

2.3 Close command

Close command (File menu)

Use this command to close the window containing the active image. If you close a window without saving, you lose all changes made since the last time you saved it.

2.4 Save command

Save command (File menu)

Use this command to save the active drawing to its current name and directory. When you save a drawing for the first time, iViZionaire displays the Save As dialog box so you can name your drawing. If you want to change the name and directory of an existing drawing before you save it, choose the Save As command.

2.5 Save As command

Save As command (File menu)

Use this command to save and name the active drawing. iViZionaire displays the Save As dialog box so you can name your drawing.

To save a drawing with its existing name and directory, use the Save command.

2.5.1 Save As dialog box

File Save As dialog box

The following options allow you to specify the name and location of the file you're about to save:

File Name
Type a new filename to save a drawing with a different name. iViZionaire adds the extension .vze.

Drives
Select the drive in which you want to store the drawing.

Directories
Select the directory in which you want to store the drawing.

Network...
Choose this button to connect to a network location, assigning it a new drive letter.

2.6 Save Quaternion to [OBJ] command

Save Quaternion to [OBJ] command (File menu)

Use this command to save a quaternion as a true 3-D object. This uses John C. Hart's Implicit code (Quaternion Julia Set server) to polygonize a quaternion formula, and then writes the triangles to a Wavefront object file. The higher the precision, the smoother the finished object becomes but the larger the object file is too. Precision is set with the Steps variable in the Parameters window, where precision=10/Steps.

Notes: some formulas produce asymmetrical object files with this routine, where one side of the quad polygon isn't resolved completely. Usually one side is markedly smoother in this case. If you enter a file name without an extension [obj] when saving an object the requester doesn't check for a file with the same extension before overwriting it. (The file extension is added later.) If you want to make sure you don't overwrite a file with an identical name, enter the filename with the applicable extension.

2.7 Save Quaternion to [POV] command

Save Quaternion to [POV] command (File menu)

Use this command to save a quaternion as a true 3-D object. This uses John C. Hart's Implicit code (Quaternion Julia Set server) to polygonize a quaternion formula, and then smooths and outputs the triangles to a POV-ray file. The file is written as a simple scene, the smooth triangles part of a "union" object, with camera and lighting elements compatible with POV-Ray 3.5. This can be used as a starting point for more complex compositions. The higher the precision, the smoother the finished object becomes but the larger the object file is too. Precision is set with the Steps variable in the Parameters window, where precision=10/Steps.

Notes: some formulas produce asymmetrical object files with this routine, where one side of the quad polygon isn't resolved completely. Usually one side is markedly smoother in this case. If you enter a file name without an extension [pov] when saving an object the requester doesn't check for a file with the same extension before overwriting it. (The file extension is added later.) If you want to make sure you don't overwrite a file with an identical name, enter the filename with the applicable extension.

2.8 Save Quaternion to [STL] command

Save Quaternion to [STL] command (File menu)

Use this command to save a quaternion as a true 3D object. This uses John C. Hart's Implicit code (Quaternion Julia Set server) to polygonize a quaternion formula, and then writes the triangles to a STL solid file. STL files are used with 3D printers and other machinery. The higher the precision, the smoother the finished object becomes but the larger the object file is too. Precision is set with the Steps variable in the Parameters window, where precision=10/Steps.

Notes: some formulas produce unsymmetrical object files with this routine, where one side of the quad polygon isn't resolved completely. Usually one side is markedly smoother in this case. If you enter a file name without an extension [stl] when saving an object the requester doesn't check for a file with the same extension before overwriting it. (The file extension is added later.) If you want to make sure you don't overwrite a file with an identical name, enter the filename with the applicable extension.

2.9 Save Quaternion to [DXF] command

Save Quaternion to [DXF] command (File menu)

Use this command to save a quaternion as a true 3-D object. This uses John C. Hart's Implicit code (Quaternion Julia Set server) to polygonize a quaternion formula, and then writes the triangles to an AutoCad dxf file. The higher the precision, the smoother the finished object becomes but the larger the object file is too. Precision is set with the Steps variable in the Parameters window, where precision=10/Steps.

Notes: some formulas produce asymmetrical object files with this routine, where one side of the quad polygon isn't resolved completely. Usually one side is markedly smoother in this case. If you enter a file name without an extension [dxf] when saving an object the requester doesn't check for a file with the same extension before overwriting it. (The file extension is added later.) If you want to make sure you don't overwrite a file with an identical name, enter the filename with the applicable extension.

2.10 Export Setup

Export Setup (File menu)

Use this command to set max faces, the target object size, and max vertices, the maximum number of indices that are allocated by the polygonizing routine. Defaults are 50,000 faces and 5,000,000 vertices. Use less to limit the size of the output file or the amount of memory used while polygonizing. Use more if necessary for higher resolution object files. With 50,000 triangle faces the output file is approximately 1.6MB for the [obj] format.

Note: the max faces variable has no effect on the size of [Dxf] exports. Dxf-formatted objects are saved without slimming.

2.11 Load Texture [QTX] command

Load Texture command (File menu)

Use this command to load variables that make up the texture and noise parameters. This also loads the palette, coloring filter, orbit trap and coloring options in a texture file [qtx].

2.12 Save Texture [QTX] command

Save Texture command (File menu)

Use this command to save the variables that make up the texture and noise parameters for the current figure. This also saves the palette, coloring filter, orbit trap and coloring options in the texture file [qtx].

2.13 Load Text Format [VZT]

Load a data file in text (platform independent) format.

2.14 Save Text Format [VZT]

Save a data file in text (platform independent) format.

3 Edit Menu

Edit menu commands

The Edit menu offers the following commands:

Undo	Undo last edit or action.
Functions and Type	Edit formula/type data.
Parameters	Edit initial parameters.
Size	Change size of image
Palette	Edit color palette
Lighting	Edit light point and Phong variables

3.1 Undo command

Undo command (Edit menu)

Use this command to undo the last action (except resizing.)

3.2 Functions and Type

Functions and Type Window

Fun #1 and Fun #2 are popup menu controls for selecting up to two built-in formulas. There are additionally a Type popup menu control, an Arg control, 'S' (real) and 'S' (imag) controls, and an Arg Limit control that are used in various fractal types and formulas. User-defined functions are supplied through popups controls fn1-fn4.

The Type control selects a value of 0 to 9:

For a value of 0, the first formula is always used and the second formula is ignored.

For a value of 1, the second formula is processed and the first formula is ignored. Type 1 is of use only if you are switching between two functions and don't want to reenter them each time you plot the other one.

For a value of 2, the first formula is processed if the real component of Z is greater than 0; else the second formula is used.

For value of 3, the first formula is processed and its output becomes the input of the second formula, which is then processed.

Type 4 takes the average of Fun#1 and Fun#2.

Type 5 alternates between the two functions while iterating.

Type 6 takes the quotient of both functions.

Type 7 uses the lowest iterative results of both Fun#1 and Fun#2

Type 8 uses the highest iterative results of both Fun#1 and Fun#2

Type 9 uses the Formula box to enter up to 150 characters per formula.
Text can be pasted from the clipboard to the formula box by using the keystroke apple-v.

About formula syntax: This applies if you elect to enter your own formula into one of the function boxes and use the parser to generate the plot. The use of parenthesis is necessary around complex exponents or variables. E.g.: '(z-i)^(1.5e)'. For a complete list of variables, operators and functions recognized by the parser, see Parser Information.
User-named-complex variables and constants may be included in a formula. A variable must begin with a letter and may contain numbers and letters only. A variable may be up to 9 characters long. A constant may be up to 20 digits long, including the decimal point.
iViZionaire uses syntax similar to Fractint's formula style with an initialization section, followed by the main formula, and an optional bailout routine. Comments may be entered on the same line with a proceeding ';'. These are provided to allow iViZionaire users to more easily convert Fractint formula types to iViZionaire use. A ':' terminates the initialization section. Multiple phrases may be entered in the main formula or initialization sections on the same line by using the terminator ',' between phrases. Note: only a small subset of Fractint variables are supported by iViZionaire, so complicated Fractint formulas may not work, or could give misleading results.

Click on Draw to apply new formula options without closing the window. Click on Stop to halt the drawing. Click on the Okay button to use the formulas currently displayed in the window and redraw the figure, or Close to exit the window without making any further changes.

3.3 Parameters

Parameters

This command opens the formatting parameters window for the 2D/3D generator.

For 2D images only the bailout and iterations variables are used here.

Quad complex constants are cr, ci, cj and ck.

Steps and Fine Tune are pitch adjustments that bear on the quality of the figure or model at the expense of lengthier calculations and larger object files.

A larger bailout can increase the fullness of a quad figure. For 2D images reducing the bailout below its standard value of 4 may help to eliminate excess detail. Iterations controls the graininess and esthetics of the quad object. Higher iterations are generally not needed for most quaternion images, but may be useful for increasing definition in 2D Julia and cquat fractals.

Three rotate variables determine the 3D angle of rotation.

Click on Draw to apply new parameters without closing the window. Click on Stop to halt the drawing. Click on the Okay button to use the parameters currently displayed in the window and redraw the figure, or Close to exit the window without making any further changes.

3.4 Size command

Size (Edit menu)

This allows you to set the drawing area for a picture, independent of the screen size. It also shows which size is currently in use. The aspect for the drawing is based on the ratio of a drawing window's width to vertical height. The size of an image can have a standard 4/3 or 1/1 aspect from 240X180 to 800X800, or a custom aspect set by the W(idth) and H(eight) boxes. Larger bitmap sizes can be created using the Image/Plot to File command

3.5 Palette command

Palette command (Edit menu)

Use the palette editor to modify the palette(s) in use. If the image has been drawn recently, changes to the palette will show up immediately (colors are mapped to the active image.) Otherwise the modified palette will take effect when you close the palette dialog and redraw the image.

There are copy and spread options to smooth or customize the active image palette in iViZionaire.

Colors are shown in 8 groups of 32 colors, or 256 color indexes total. Colors between the indexes are interpolated so there are actually 256X255 or 65000+ possible colors in the iViZionaire palette.

Use the RGB-slider controls to edit any color in the palette. Select Copy to copy any color to another spot in the palette. Select Spread to define a smooth spread of colors from the current spot to another spot in the palette. Copy and Spread take effect immediately when you select another spot with the mouse button. You can cancel the operation with the Cancel button. In iViZionaire, colors do not cycle smoothly when you adjust the RGB sliders. This would be

too slow with true color. The Map button is used to map color changes to an image after you are done adjusting the sliders.

Use Reset to reset the colors of the palette in use, to where it was before it was modified.

Use Reverse to reverse the order of the colors in the palette, excluding color index 1, the background color.

Use Neg to create a palette that is the complement of the current palette, excluding color index 1.

Use SRG to switch the red and green components of all palette colors except color index 1. Use SRB to switch the red and blue components of all palette colors except color index 1. You can use these buttons to form eight different palettes by repeatedly switching red, green and blue components.

Use the Random palette button to randomize the current palette.

Note: unless you click on Reset before exiting the editor, changes are permanent to the palette edited, no matter which way you close the editor (Okay button or Close box.)

3.6 Lighting

Lighting Window

The LightPoint variables (lightx thru lightz) determine the direction of the light source used in the ray-tracing algorithm. The ViewPoint represents the angle, at which the object is ray-traced, which can affect Phong highlights greatly. This has no effect on the camera view.

The lighting variables Shininess, Highlight, Gamma, Diffuse and Ambient are used to adjust ambient light and highlights. The ranges for these variables appear beside their label. Decreasing the Shininess value increases light reflected by the quaternion and the apparent sheen on the quaternion's surface. The Ambient value controls the amount of ambient light that illuminates the quaternion. The Highlight value increases or decreases the specular (Phong) highlighting, while the Gamma value increases or decreases the intensity of the light source's illumination. Once a plot is drawn (in the current session), the lighting variables and light point can be changed without redrawing the quaternion.

Click the Apply button to redisplay a plot after changing the lighting variables or light point. Click the Okay button to close the lighting window, and apply new settings, if the variables were modified. Click on Close to close the lighting window, keeping the current settings.

4 Image Menu

Image menu commands

The Image menu offers the following commands:

Draw	Draw the figure.
Scan	Scan Mandelbrot border for quaternion Julia set.
Zoom	Zoom into or out of current image.
Plot to File	Plot image to png file.
Reset Ranges	Reset Z ranges.
Symmetry	Horizontal, Vertical or XY Symmetry
Mapping	Mapping controls how the bailout is processed for each pixel.

4.1 Draw command

Draw command (Image menu)

Use this command to draw or redraw the image for the current fractal variables. Clicking inside the draw window with the left-mouse button stops all plotting.

4.2 Scan command

Scan (Image menu)

Enabled when the Type is Mandelbrot. After executing this command, a small quaternion Julia set is generated in the left upper corner, for the current quad constants and formula, and each time you click in the draw window. Use the Mandelbrot border as a guide to finding Julia sets. Generally if you click too far outside the Mandebrot borders the space occupied by the Julia quaternion set vanishes. Most interesting are the Julia sets found right on the M-border or just outside it.

Quaternion math is used where possible if the Type is set to Quaternion; else hypercomplex math is used for the Hypernion type, etc.

Once you find an interesting quaternion set press the space bar and a new window is opened that sets the fractal parameters to those in the exploratory qjulia window. (The parameters in the scan window revert to their original Mandelbrot settings.)

Click on the status_bar area of the draw window to exit this command without generating a new Julia set. (Parameters and bitmap revert to the previous state of the image.)

4.3 Zoom command

Zoom (Image menu)

Turns on zoom mode, so that detail of the current plot may be magnified or centered. Alternatively, just click inside any drawing window, move the mouse, and the zoom box will appear. Using the mouse, move the zoom box over the portion of the plot you wish to magnify. Hold the left mouse button (or left arrow key) to shrink the box or the right button (or right arrow key) to enlarge it. You may zoom in by defining a box inside the current drawing area. You zoom out by drawing a box outside the current drawing area. Use the up arrow key to enlarge the zoom box X4 for quickly zooming outward or the down arrow key to shrink the zoom box by 4. You start a zoom by pressing the space bar. The program opens a

new window and begins a new plot at the zoom coordinates. You abort a zoom by pressing the esc key (shift-esc with OS X 10.4) or click inside the status bar (bottom of draw window.)

For 2D plots you can also rotate the zoom box using the Tab and Enter keys. Holding the Enter key down rotates the box clockwise and depressing the Tab key rotates the box counter-clockwise. This is useful to allign some fractals horizontally, vertically or at any angle.

4.4 Plot to file

Plot to File (Image menu)

This allows you to plot a large bitmap directly to a .png file without the added system requirements of keeping the whole bitmap in memory. The target bitmapWidth can be 1024 to 14400. Target Height is set by the drawing window aspect, or Target_Height=draw_height/draw_width*Target_Width. Click on Okay to set the target file name and start a new plot to file, or Cancel to exit the window without plotting.

4.5 Reset Ranges

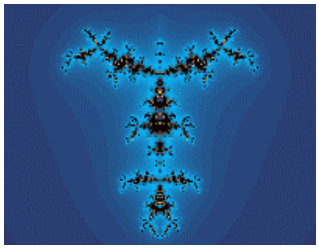
Reset Ranges (Image menu)

The Reset Ranges command resets the real Z and imaginary Z ranges of an image to fit the current drawing window's aspect. This is necessary when changing image size and the drawing window aspect changes also, since a different aspect will otherwise stretch or condense a figure. After reseting the ranges you may need to recenter or rezoom a figure.

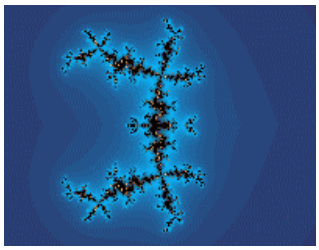
4.6 Symmetry

Symmetry (Image menu)

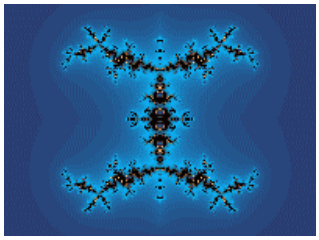
This produces a mirror image from left to right (vertical) or top to bottom (horizontal) or both (xy). You can zoom with symmetry, but the results will be uncertain if the zoom box is off-center on the window.



Vertical symmetry



Horizontal symmetry



XY symmetry

4.7 Mapping

Mapping

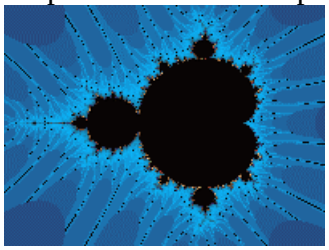
Mapping controls how the bailout is processed for each pixel.

Z-Real	Mapping based on real part of z only.
Z-Imag	Mapping based on imaginary part of z only.
Abs(Z-Real)	Mapping based on absolute value of real part of z.
Abs(Z-Imag)	Mapping based on absolute value of imaginary part of z.
Z-Real + Z-Imag	Mapping based on sum of parts of z.
Abs(Z-Real)+Abs(Z-Imag)	Mapping based on absolute value of parts of z.
>Abs(Z-Real) or Abs(Z-Imag)	Mapping based on highest absolute value of parts of z.
<Abs(Z-Real) or Abs(Z-Imag	Mapping based on lowest absolute value of parts of z.
Abs(Z)	Mapping based on absolute value of z.

4.7.1 Z-Real

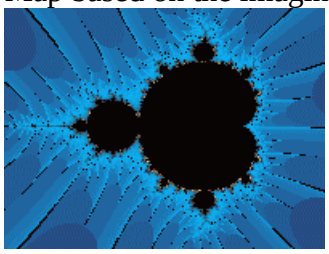
Z-Real

Map based on the real part of the complex number Z; used to map exponential Julia sets, etc.



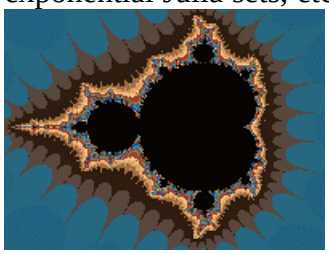
4.7.2 Z-Imag

Z-Imag
Map based on the imaginary part of the complex number Z.



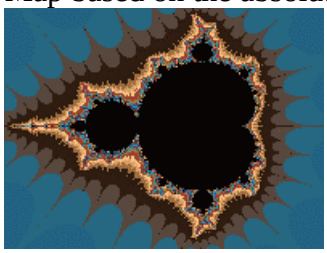
4.7.3 Abs(Z-Real)

Abs(Z-Real)
Map based on the absolute value of the real part of the complex number Z; used to map exponential Julia sets, etc.



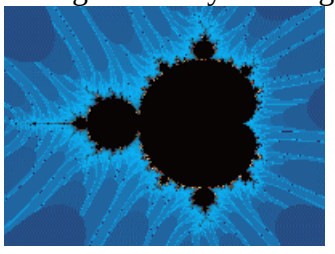
4.7.4 Abs(Z-Imag)

Abs(Z-Imag)
Map based on the absolute value of the imaginary part of the complex number Z.



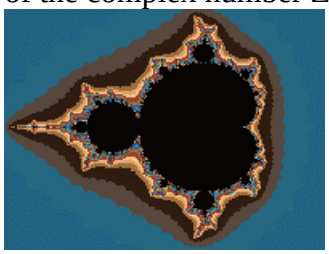
4.7.5 Z-Real+Z-Imag

Z-Real + Z-Imag
Map based on the sum of the real part and the imaginary part of the complex number Z. Changes the way banding appears in complex mappings.



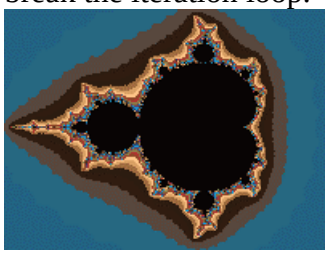
4.7.6 Abs(Z-Real)+Abs(Z-Imag)

Abs(Z-Real) + Abs(Z-Imag)
Map based on the absolute value of the real part plus the absolute value of the imaginary part of the complex number Z. Changes the way banding appears in complex mappings.



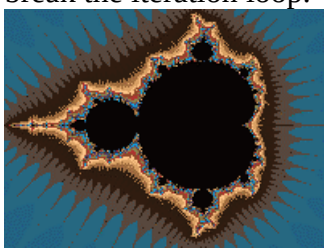
4.7.7 >Abs(Z-Real) or Abs(Z-Imag)

>Abs(Z-Real) or Abs(Z-Imag)
Map based on the greater of the absolute value of the real part or the imaginary part of the complex number Z. Works like a logical 'or', where either part of z must exceed zlimit to break the iteration loop. Changes the way banding appears in complex mappings.



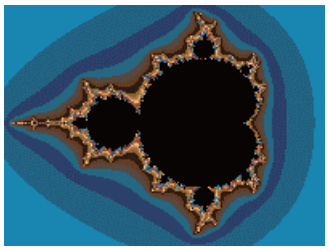
4.7.8 <Abs(Z-Real) or Abs(Z-Imag)

<Abs(Z-Real) or Abs(Z-Imag)
Map based on the lesser of the absolute value of the real part or the imaginary part of the complex number Z. Works like a logical 'and', where both parts of z must exceed zlimit to break the iteration loop. Changes the way banding appears in complex mappings.



4.7.9 Abs(Z)

Abs(Z)
Map based on the absolute value of the complex number Z (traditionally calculated by taking the square root of the sum of the squares of the real and imaginary parts of Z, but Fractal Zplot uses only the 'sum'(modulus of z) for break-point tests.) The standard method of mapping Julia and Mandelbrot sets.



5 Type Menu

Type menu commands

The Type menu offers the following commands:

Mandelbrot0	Mandelbrot set (orbit starts at zero.)
MandelbrotP	Mandelbrot set (orbit starts at pixel.)
Julia	Julia set.
3D Quaternion	Set fractal type to 3D quaternion.
3D Hypernion	Set fractal type to 3D hypercomplex quaternion.
3D Cquat	Set fractal type to 3D complexified quaternion.
2D Quaternion	Set fractal type to 2D quaternion.
2D Hypernion	Set fractal type to 2D hypercomplex quaternion.
2D Cquat	Set fractal type to 2D complexified quaternion.

5.1 Mandelbrot0

Mandelbrot0 (Type menu)

Mandelbrots base their mapping on varying inputs of complex C, which corresponds to the min/max values set in the Parameters window. With Mandelbrot0, the initial value of Z is set to zero.

5.2 MandelbrotP

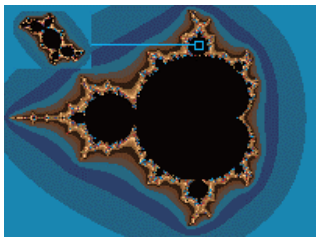
MandelbrotP (Type menu)

Mandelbrots base their mapping on varying inputs of complex C, which corresponds to the min/max values set in the Parameters window. With MandelbrotP, the initial value of Z is set to the value of the pixel being iterated. Usually used with cubic Mandelbrot formulas.

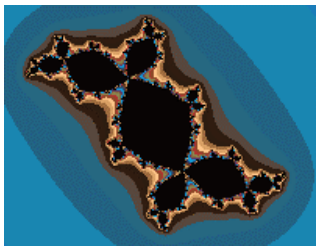
5.3 Julia

Julia (Type menu)

Julia sets normally have a fixed complex C, with varying inputs of Z, which corresponds to the min/max values set in the Parameters window.



Julia from Mandelbrot



Julia set

5.4 3D Quaternion command

3D Quaternion (Type menu)

Use this command to set the fractal type to quaternion. Quaternion math is used to calculate the 3D figure where possible, else hypercomplex math is used.

5.5 3D Hypernion command

Hypernion (Type menu)

Use this command to set the fractal type to a hypercomplex 3-D quaternion. A hypernion uses hypercomplex math to shape the 3-D object, which usually results in a squared-off shape, rather than the rounded shape of the typical quaternion.

5.6 3D Cquat command

3D Complexified Quaternion (Type menu)

Use this command to set the fractal type to a complexified quaternion. A cquat uses another variation of quad math to shape the 3-D object, which usually results in a more chaotic shape than the rounded lines of the typical quaternion. Not all formulas support the Complexified Quaternion type. In that case, the formula will default to hypercomplex algebra when this type is selected.

5.7 2D Quaternion command

2D Quaternion (Type menu)

Use this command to set the fractal type to 2D quaternion. Quaternion math is used to calculate the 2D image where possible, else hypercomplex math is used.

5.8 2D Hypernion command

2D Hypernion (Type menu)

Use this command to set the fractal type to a hypercomplex 2D quaternion. A hypernion uses hypercomplex math to generate the 2D image.

5.9 2D Cquat command

2D Complexified Quaternion (Type menu)

Use this command to set the fractal type to a complexified quaternion. A cquat uses another variation of quad math to generate the 2D image, which usually results in a more chaotic picture. Not all formulas support the Complexified Quaternion type. In that case, the formula will default to hypercomplex algebra when this type is selected.

6 Render Menu

Render menu commands

The Render menu offers the following commands:

Coloring Filter	Define coloring filter.
Palette Coloring	Palette-based coloring options.
Noise	Edit noise factors.
Texture Scale	Set scaling factor for texture.
Generalized Coloring	Generalized RGB-Coloring options.

6.1 Coloring Filter command

Coloring Filter

Here you define a coloring filter based on a real function. A generalization of Earl Hinrichs' sine-wave coloring method, the function can be any formula, up to 80 characters, that uses the z-buffer variable and framing variables x and y. Sample function: .1*sin(z)+cos(x*x). The Magnify slider is used to control the intensity of the filter. Click on Apply to apply a new coloring formula without closing the window. Click on Okay to close the window and apply

changes. Click on Cancel to close the window without applying changes. Use the Random Filter button to generate a random coloring filter. The best filters will use the z value and one of the other variables (x or y.)

Quaternions normally use palette index one (the second index, zero being reserved for the background color) for their predominant color, with pixel intensities/colors affected by the lighting variables. When the coloring filter formula is defined, up to 235 colors can be used (the full palette) to create mixed textures.

The trig and exponential functions translated include sine (sin), arc sine (asn), cosine (cos), arc cosine(acs), tangent (tan), hyperbolic tangent (th), hyperbolic sine (sh), hyperbolic cosine (ch), log (log), natural log (ln), power (pow), arc tangent (atn), absolute value (abs), exponential (exp) and square root (sqr.)

The math functions are *(multiply),-(subtract),/(divide), and +(add).

The constants are PI and E (ln (1)), plus any floating-point number up to 9 digits (including the decimal point).

The power function (x to the y power) is entered in standard notation: x^y', with optional parenthesis necessary around complex exponents or variables.

Note: Range limits exist for arguments to these functions: exp, arc sine, hyperbolic sine, arc cosine, hyperbolic cosine, arc tangent, and hyperbolic tangent (+/-100.0 for the exponential, +/-200.0 for hyperbolic functions, +/-1.0 for the arc functions), the log functions (must be >0) and the power function (x must be integral and non-zero when y<0, and 0^0 is undefined). Square root is undefined for x<0. No filtering is done when these limits are exceeded.

Syntax for an acceptable formula is AS([XY])+bs([xy])...
.up to 80 characters per formula. Algebraic notation is supported to a limited degree. E.G. you can enter a variable as 2x^2, instead of 2*x*x.
A and B are optional constants.

S is an optional trig function (1 to three letters: 1 will work for sine, cosine and tangent, but use the above abbreviations for the other functions. X and Y are the standard variables. The '+' could be any of the math functions.

The parser interprets up to 10 levels of parenthesis. Use parenthesis to separate complex expressions. Use parenthesis to embed trig functions within other trig functions, etc.

6.2 (3D) Palette Coloring

(3D) Palette Coloring

Atan Coloring
Uses an algorithm by David Makin to color a quaternion image. The angle of each point (as it escapes the quaternion border) is used to color the image.

Bof60 Coloring
A variation of the Bof60 algorithm found in the classic Pietgen/Ricter text, The Beauty of Fractals, adapted by David Makin, is used to color a quaternion image. The smallest magnitude of z (found while calculating the quaternion border) is used to render the image.

Potential Coloring
The escape value of z (at the quaternion border) is used to color the image.

Distance Coloring
The distance of z from zero (at the quaternion border) is used to color the image.

Orbit Traps
These includes methods that trap the orbit of a point if it comes in range of a pre-specified area or areas.

The Epsilon-Cross method colors points only if the absolute value of Z-real or Z-imaginary is less than or equal to Epsilon (a small value.) Other points are mapped at the time they blow up (exceed the zlimit.) This produces hair-like structures that branch wildly from the complex set boundaries.

The Globe method uses a circular area around the origin to map a point's orbits. This produces sphere-like structures.

The Ring method uses an area formed by two circles around the origin to map a point's orbits. This produces ring-like structures.

The Four-Circles method (Paul Carlson) uses four circular areas to map a point's orbit. This produces sphere-like structures.

The Square method uses an area formed by two squares around the origin to map a point's orbits. This produces ring-like structures with right angles.

The Petal method (Paul Carlson) also uses four trap areas to form flower-like patterns.

Enter a value for Epsilon and Epsilon2, which are used to define the size of the orbit trap areas (.001-2.0 and 0.0-epsilon.) The exclude box is used to exclude the first # iterations from orbit trapping.

Epsilon2 is used to create windows into the stalks. The default value is 0.0, which produces solid stalks. Epsilon2 has no effect on the Petal method.

Filter
Based on Stephen C. Ferguson's filter algorithms in his program Iterations, this control allows you to choose one of 26 tail-end filters for surface rendering. Corresponds roughly to its effect on the basic Mandelbrot-squared set. The effect will vary with the formula and fractal type chosen.

The Magnify variable is used to intensify or de-intensify the effect of the filter. This value can range from 1-500 nominally.

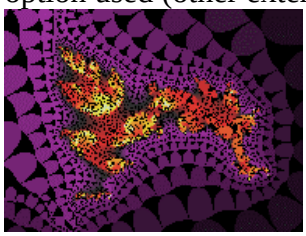
Click on Draw to apply a new palette-coloring option without closing the window. Click on Stop to halt the drawing.
Click on Okay to close the window and apply changes. Click on Close to close the window without applying any additional changes.

6.3 (2D) Pallete Coloring

(2D) Palette Coloring

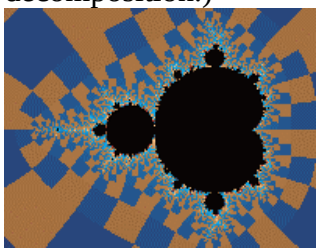
In this window the following rendering methods are available (for 2D fractals only)

Biomorph
Biomorphs test the real Z and imaginary Z values after breaking the iteration loop. If the absolute value of either is less than the preset zlimit, the point is mapped as part of the set. This method produces biological-like structures in the complex plane. Normally the biomorph tendrils are colored in the set color (the color reserved for non-divergent or inner points.) With the Set Only flag on, the tendrils are colored according to the color-scaling option used (other external points are colored in the background color.)



Decomposition
Toggle the External/Internal Decomposition options for either an external or internal decomposition. An external decomposition decomposes points that are outside the complex set. An internal decomposition decomposes the complex set. For Mandelbrot/Julia curves, z-arg is broken into two parts for a binary decomposition.

(Consult The Beauty of Fractals by Peitgen & Richter for a mathematical explanation of decomposition.)



Binary Decomposition

Orbit Traps

This includes methods that trap the orbit of a point if it comes in range of a pre-specified area or areas.

The Epsilon-Cross method colors points only if the absolute value of Z-real or Z-imaginary is less than or equal to Epsilon (a small value.) Other points are mapped at the time they blow up (exceed the zlimit.) This produces hair-like structures that branch wildly from the complex set boundaries. Note: Epsilon pictures use palette index 255 for the background color.

The Globe method uses a circular area around the origin to map a point's orbits. This produces sphere-like structures.

The Ring method uses an area formed by two circles around the origin to map a point's orbits. This produces ring-like structures.

The Four-Circles method (Paul Carlson) uses four circular areas to map a point's orbit. This produces sphere-like structures.

The Square method uses an area formed by two squares around the origin to map a point's orbits. This produces ring-like structures with right angles.

The Petal method (Paul Carlson) also uses four trap areas to form flower-like patterns.

The Outside Radius variable is used to define the size of the trap areas. Start with .02 and increase or decrease to taste. The Inner Radius variable opens a window inside the "stalks" or trap areas. Use Exclude to skip the first n# iterations from the trap zone. This is sometimes necessary with the Epsilon orbit method, to eliminate stray stalks and clarify the image.

To produce the maximum 3-D effects (as Phil Pickard and Paul Carlson do) with these options, the inside coloring method Linear Map must be set, and the Color Scaling variable should be equal to –1.2*Radius. Also the palette should be changed to one continuous spread, the brightest color in palette index #2 and the darkest color in palette index # 256. This style of palette is automatically and randomly generated by the Smooth button in the Colors editor when the Smooth box is set to zero and an orbit-trap method is selected.

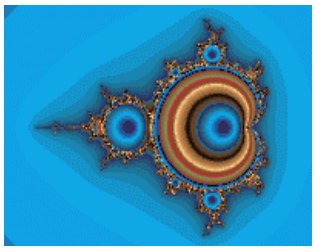
6.3.1 Optional Coloring Methods

Optional Coloring Options

Inside Coloring Options

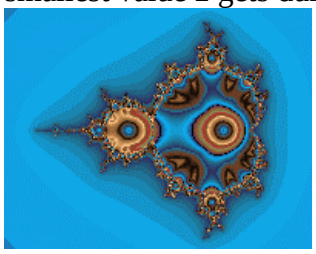
Level Curves
Level-curves map the set points based on the value of Z. This allows the inside of the complex set to be color-scaled.

Log Map
produces colored bands on the inside of the complex set. Points are mapped according to what the value of z is at final iteration.



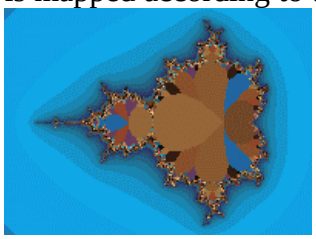
Log Map

Small Log
produces circular patterns inside the complex set. Points are mapped according to the smallest value z gets during iteration.



Small Log

Indexed Log
is mapped according to the escape time it takes z to reach its smallest value.

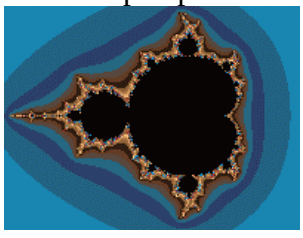


Indexed Log

Linear Map is mapped like Log Map (with the mapped value of the function at its final iteration applied to the color palette) and produces 3D-like effects with orbit-trap renderings. Period uses periodicity to color points inside the MS.

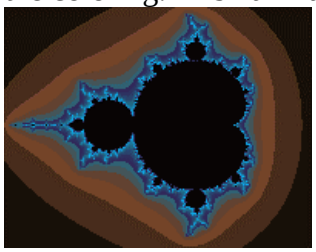
Outside Coloring Options

Escape
The Escape option uses the point's escape time.



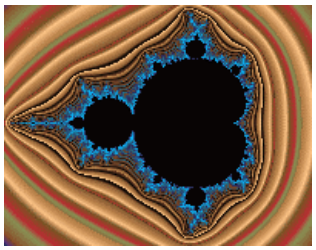
Escape-time coloring.

Level
A point is colored based on its logarithmic escape. The QFactor controls the smoothness of the coloring. A small number from 10-20 works best here..



Log coloring.

Continuous Potential
A point is colored based on its continuous potential (when it blows up.) The QFactor controls the smoothness of the coloring. A higher number from 2000-20000 or more works best here.



Continuous-Potential coloring.

Angle uses the absolute value of a point's exit angle (theta.) This is the atan method in Fractint. Angle-Iteration uses the angle formed by the difference between a point's last two exit values and subtracts the point's escape time. This is Paul Carlson's atan method.

Set Only
The Set Only flag plots all outside points in the background color.

Cutoff
This variable acts as a palette multiplier or divider, depending on whether the value entered is less than or greater than 1.0. The palette color is divided by the scaling factor to speed up or slow down color changes. For level curves and orbit-trap pictures, use a negative cutoff value to maintain a smooth palette. This ensures that the multiplier is used before the (floating-point) palette values are converted to (integer) palette indexes. For pictures that use a log palette (excluding orbit-trap images) a value of .01 is usually optimal for color scaling. Any positive value will work for the Continuous Potential method, though a small negative value will also produce interesting results. The Escape method can use a small value of .0039 to .00039 to cover the complete palette. A value of 2.0 to 4.0 works well with the Atan coloring method.

The **Carlson Extensions button** selects the Set Only flag as well as the Linear map coloring method, and sets the Color Scaling variable to –(Orbit-trap) Radius*1.2. This automates the process of producing Paul Carlson's 3-D like fractals, when used in conjunction with an orbit-trap method (see MS Rendering dialog.)

The **Bubble Extensions** button selects the Bubble map and Use Level coloring methods, and sets the Cutoff variable to a small negative value. It also selects a Divide-by-two palette if a split palette has not already been chosen. (The Palette may need to be redesigned in the Palette editor to match the split factor.) This automates the process of producing Paul Carlson's 3-D bubble-like fractals.

The Divide by (1-16) radio buttons are used to assign different colors to the orbit-trap tendrils. A value of 4 will split the palette into four-color spreads so that tendrils will alternate through these four color ranges. With a Smooth setting of 0 in the Colors editor, the Random palette generator will automatically produce a palette suitable for the number of splits used. Also useful with bubble and atan pictures. Note: when you choose a different split factor the color palette needs to be redesigned in the Palette editor. This is easily automated by using the Random palette button which converts the current palette into one that is optimal for the split factor.

6.3.1.1 Potential Log Map

Log Map produces colored bands on the inside of the complex set. Points are mapped according to what the value of z is at final iteration.

6.3.1.2 Small Log

Small Log produces circular patterns inside the complex set. Points are mapped according to the smallest value z gets during iteration.

6.3.1.3 Indexed Log

Indexed Log is mapped according to the escape time it takes z to reach its smallest value.

6.3.1.4 Potential Linear Map

Linear Map is mapped like Log Map (with the mapped value of the function at its final iteration applied to the color palette) and produces 3D-like effects with orbit-trap renderings.

6.3.1.5 Escape

The **Escape** option uses the point's escape time to color points outside the Mandelbrot set.

6.3.1.6 Use Level

Use Level -- All points (inside and outside) are colored according to the Level curve selected or Potential Linear if no level curve is selected.

6.3.1.7 Level

Level -- A point is colored based on its logarithmic escape. The QFactor controls the smoothness of the coloring. A small number from 10-20 works best here..

6.3.1.8 Continuous Potential

Continuous Potential -- A point is colored based on its continuous potential (when it blows up.) The QFactor controls the smoothness of the coloring. A higher number from 2000-20000 or more works best here. Deeper zooms require larger q-factor.

6.3.1.9 Atan

Atan uses the angle formed by the difference between a point's last two exit values and subtracts the point's escape time. This is Paul Carlson's atan method.

6.3.1.10 QFactor

QFactor

The QFactor controls the smoothness of the coloring. A small number from 10-20 works best for Log Palettes, while higher number 2000-20000 or more is best for Continuous Potential.

6.3.1.11 Set Only

Set Only plots all outside points in the background color.

6.3.1.12 Bubble Extensions

The **Bubble Extensions** button selects the Bubble map and Use Level coloring methods, and sets the Cutoff variable to a small negative value. It also selects a Divide-by-two palette if a split palette has not already been chosen. (The Palette may need to be redesigned in the Palette editor to match the split factor.) This automates the process of producing Paul Carlson's 3-D bubble-like fractals.

6.3.1.13 Carlson Extensions

The **Carlson Extensions** button selects the Set Only flag as well as the Linear map coloring method, and sets the Color Scaling variable to –(Orbit-trap) Radius*1.2. This automates the process of producing Paul Carlson's 3-D like fractals, when used in conjunction with an orbit-trap method (see MS Rendering dialog.)

6.3.1.14 Cutoff

The **Cutoff or color-scaling** variable acts as a palette multiplier or divider, depending on whether the value entered is less than or greater than 1.0. The palette color is divided by the scaling factor to speed up or slow down color changes. For level curves and orbit-trap pictures, use a negative cutoff value to maintain a smooth palette. This ensures that the multiplier is used before the (floating-point) palette values are converted to (integer) palette indexes. For pictures that use a log palette (excluding orbit-trap images) a value of .01 is usually optimal for color scaling. Any positive value will work for the Continuous Potential method, though a small negative value will also produce interesting results. The Iteration method can use a small value of .0039 to .00039 to cover the complete palette. A value of 2.0 to 4.0 works well with the Atan coloring method.

6.3.1.15 Divide by

The Divide by (1-16) radio buttons are used to assign different colors to the orbit-trap tendrils. A value of 4 will split the palette into four-color spreads so that tendrils will alternate through these four color ranges. With a Smooth setting of 0 in the Colors editor, the Random palette generator will automatically produce a palette suitable for the number of splits used. Also useful with bubble and atan pictures. Note: when you choose a different split factor the color palette needs to be redesigned in the Palette editor. This is easily automated by using the Random palette button which converts the current palette into one

that is optimal for the split factor.

6.3.2 Binary

Binary Decomposition

For Mandelbrot/Julia curves, z-arg is broken into two parts for a binary decomposition.

6.3.3 External

External Decomposition

An external decomposition decomposes points that are outside the complex set.

6.3.4 Internal

Internal Decomposition

An internal decomposition decomposes the complex set.

6.3.5 Biomorph

Biomorph

Biomorphs test the real Z and imaginary Z values after breaking the iteration loop. If the absolute value of either is less than the preset zlimit, the point is mapped as part of the set. This method produces biological-like structures in the complex plane. Normally the biomorph tendrils are colored in the set color (the color reserved for non-divergent or inner points.) With the Set Only flag on, the tendrils are colored according to the color-scaling option used (other external points are colored in the background color.)

6.3.6 Newton

Newton

The Newton flag is used to map the zeros of a particular function after the Newton transformation has been applied to the function. The program doesn't make the transformation (z-(f(z)/f'(z)), where ' stands for d/dx). When a built-in formula (or custom formula) is chosen that doesn't contain this transformation, the bailout is checked for convergence only.

This flag is mutually exclusive with the Renormalization flag, and automatically excludes all points that don't converge to one of the attractors set, within the preset number of iterations. The points that converge are colored according to the root they converge to and the time it takes to converge. The non-converging points are mapped with the set color or their level set color (with a level flag set) after the maximum number of iterations. Non-converging points show up typically as round areas or spots.

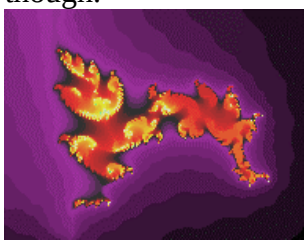
Generally, a limit of 50 iterations gives optimum results. The Newton transformation is normally used with Julia sets, as the attractors (solutions of the formula) can be calculated beforehand. It's also possible to explore the Mandelbrot set applied to Newton's method, but only with some of the built-in formulas mentioned above. It this case, the solutions of the

formula for every point on the screen have to be calculated separately, which the program does in a dedicated routine.

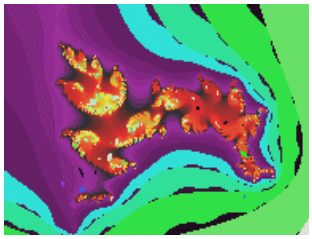
6.3.7 Renormalization

Renormalization

The Renormalization flag uses a hierarchical lattice transformation to map magnetic phases, with either the Julia set or Mandelbrot set as the iterated function. (Consult The Beauty of Fractals by Pietgen and Richter for appropriate formulas to use.) Basically, the default-mapping algorithm checks orbits for convergence to 1 or infinity, and scales these points in different colors. This flag is mutually exclusive with the Newton flag. The default method of display actually only checks if z passes through 1. This is similar to the epsilon-cross mapping method. For the original renormalization formulas, there is a strong orbital attraction to 1. For other functions, this mapping produces unusual effects (with obscure mathematical foundations.) For the built-in functions, the convergence tests (type 2 or 3 and 6-8 display types) are the same as the ones used with the Newton flag. So either method produces similar results with the same formula. The differences are worth playing with, though.



Original picture



With renormalization

6.3.8 Epsilon

Epsilon

The Epsilon-Cross method colors points only if the absolute value of Z-real or Z-imaginary is less than or equal to the Outer Radius (a small value.) Other points are mapped at the time they blow up (exceed the zlimit.) This produces hair-like structures that branch wildly from the complex set boundaries. Note: Epsilon pictures use palette index 255 for the background color.

6.3.9 Globe

Globe

The Globe method uses a circular area around the origin to map a point's orbits. This produces sphere-like structures

6.3.10 Ring

Ring

The Ring method uses an area formed by two circles around the origin to map a point's orbits. This produces ring-like structures.

6.3.11 Four Circles

Four Circles

The Four-Circles method (Paul Carlson) uses four circular areas to map a point's orbit. This produces sphere-like structures. .

6.3.12 Square

Square

The Square method uses an area formed by two squares around the origin to map a point's orbits. This produces ring-like structures with right angles.

6.3.13 Petal

Petal

The Petal method (Paul Carlson) also uses four trap areas to form flower-like patterns.

6.3.14 Epsilon

Outside Radius

The Outside Radius variable is used to define the size of the trap areas. Start with .02 and increase or decrease to taste.

6.3.15 Epsilon2

Inner Radius

The Inner Radius variable opens a window inside the "stalks" or trap areas.

6.3.16 Exclude

Exclude

Use Exclude to skip the first n# iterations from the trap zone. This is sometimes necessary with the Epsilon orbit method, to eliminate stray stalks and clarify the image.

6.4 Noise command

Noise (Render menu)

Add and/or edit noise factors. The Blend variable determines how much noise is added to an image. The higher the blend, the more pronounced the noise appears. This also tends to darken an image, which can be compensated for by decreasing Gamma. The Grain variable determines the frequency of the noise. The higher the grain, the noisier the image appears. You can adjust how the noise maps to an image by changing the scale factors. Higher scale factors make the image noisier on the respective axis (x, y and z.) Additional variables affect the type and shaping of the noise data: Gaussian is an alternate form of noise, while Planet,

Check, Tooth, Barber and Wood apply a specific envelope to the noise. The Marble variable is used to introduce a low frequency or high frequency modulation on top of the noise. You can achieve marble-like textures by combining a high frequency marble value with a low frequency Blend value. The marble variable also adds a high-frequency bump map to the wood envelope.

The Surface Warp variable allows you to apply the same noise to a (quaternion) figure's shape also. Small values are best for creating realistic surface variations, like stone and wood grain.

To turn off noise, enter '0' for Blend. Use the Apply button to change the current noise factors for the active figure, if the figure has been drawn recently, or Okay to apply noise factors and redraw the active figure.

6.5 Texture Scale command

Texture Scale (Render menu)

Opens a window to edit texture scale factors. The higher the scale factors, the more repetitive the texture becomes. You can adjust the factors to make the texture asymmetrical on the x, y or z-axis. Scale A is used to adjust the texture scale for the Atan, Potential and Bof60 coloring options. Click on Apply to apply changes without closing the window, if the image has been drawn recently. Click on Okay to close the window and apply changes. Click on Close to close the window, keeping the current changes without redrawing the image.

6.6 Generalized Coloring

Use this command to switch to Steven Ferguson's generalized coloring mode. Images are colored using algorithmic methods that address the whole rgb spectrum, instead of the palette-based coloring methods. To switch back to palette mode, apply a coloring filter or one of the palette-based methods, such as Atan, Potential or one of the orbit traps or filters in the Palette Coloring window. Use the Apply button to change the current coloring options for the active figure, if the figure has been drawn recently, without closing the window. Click on Okay to apply new generalized options and redraw the active figure, or Close to close the window without applying any additional changes.

6.6.1 Blend buttons

Various formulas are applied while mapping colors to pixels.

6.6.2 Red/Grn/Blu controls

Red/Grn/Blu controls

Use these sliders and edit boxes adjust the color emphasis or red/green/blue mixture of an image.

6.6.3 RGB button

RGB button

Use red/green/blue mapping for pixels.

6.6.4 RBG button

RBG button

Use red/blue/green mapping for pixels.

6.6.5 GRB button

GRB command (Render menu)

Use this command to use green/red/blue mapping, if in the generalized coloring mode.

6.6.6 GBR button

GBR command (Render menu)

Use green/blue/red mapping for pixels.

6.6.7 BRG button

BRG command (Render menu)

Use blue/red/green mapping for pixels.

6.6.8 BGR button

BGR command (Render menu)

Use blue/green/red mapping for pixels.

6.6.9 Sine algorithm button

Sine Algorithm command (Render menu)

When color values exceed the range of rgb components, the values are scaled with Steven C. Ferguson's sine algorithm.

6.6.10 Sawtooth algorithm button

Triangle Algorithm command (Render menu)

When color values exceed the range of rgb components or palette indexes, the values are scaled with a triangle algorithm, or linear ramp.

6.6.11 Gray Scale button

Gray Scale button

Color the active image with gray tones.

6.6.12 Invert button

Invert command (Render menu)

Invert image colors.

6.6.13 Fractal Dimension button

Fractal Dimension command (Render menu)

Generalized fractal dimension algorithm (S. Ferguson), for use with any blend option.

7 Demo Menu

Demo menu commands

The Demo menu offers the following commands, which illustrate various features of iViZionaire:

Random Quaternion	Generate random quad fractal.
Random Julia	Generate random Julia fractal.
Random Newton	Generate random Julia fractal using Newton method.
Random Halley	Generate random Julia fractal using Halley's method.
Random Stalks	Generate random orbit-trap fractal.
Random Bubbles	Generate random bubble fractal.
Random Composite	Generate random composite Julia fractal.
Random Render	Select a random rendering.
Random Setup	Set preferences for random commands.

7.1 Random Quaternion

Random Quaternion (Demo menu)

A random quad fractal is generated. A set of formulas appropriate for quaternions is scanned to find an interesting Julia quad set, and then the parameters are adjusted to produce the image. The quad type generated is the Type selected from the Type menu.

7.2 Random Julia

Random Julia (Demo menu)

A random Julia fractal is generated. Many of the built-in options of iViZionaire are selected

on a random basis, and the Mandelbrot space for one of ninety built-in formulas is scanned for an interesting Julia set. Tip: some things remain to be done after the Julia set is drawn. Feel free to experiment with all the parameters, reframe the image, change palettes etc. This routine provides a fast intro to many options in iViZionaire that the user may be unfamiliar with: no knowledge of fractal science/math required!

7.3 Random Stalks

Random Stalks (Demo menu)

A random Julia fractal is generated using one of Paul Carlson's orbit traps.

7.4 Random Bubbles

Random Bubbles (Demo menu)

A random Julia fractal is generated using one of Paul Carlson's bubble method.

7.5 Random Newton

Random Newton (Demo menu)

A random Julia fractal is generated using Newton method.

7.6 Random Halley

Random Halley (Demo menu)

A random Julia fractal is generated using Halley's method.

7.7 Random Composite

Random Composite (Demo menu)

A random Julia quad fractal is generated using one of the composite Types. Two formulas are selected at random and mixed using one of Types 2-8. This option can be applied to all built-in formulas, except the Gallet formulas (90-99.)

7.8 Random Render

Random Render (Demo menu)

The rendering options for the current fractal are randomized. Does not affect formula or range variables. For quaternion images, a random coloring filter is applied, if palette coloring is selected, else the generalized coloring parameters are randomized..

7.9 Random Setup

Random Setup (Demo menu)

Here you customize random variables to direct how the random scanning process works.

There are radio boxes that allow you to customize how random variables are processed to create new fractals:

Formula -- (default on) check to randomize built-in formula used
Z-Space -- (default on) check to set default z-space
Constants -- (default on) check to randomize the complex constants cj-ck
Custom Formula -- (default off) check to generate a random formula -- n/a with the Random Newton and Random Halley options

These settings are saved in a "prefs" file in the startup directory.

8 Video Menu

The Video Menu offers the following commands:

Write Movie	Write frames to QuickTime movie file.
Add Frame	Add current image to frame buffer.
Edit Frames	Edit frame buffer.
Load Frames [QFRM]	Load frame buffer.
Save Frames [QFRM]	Save frame buffer.

8.1 Write Movie

With this command the video frame buffer is written to a QuickTime movie. First you choose the width of the video, up to 2048 (height is determined by the current image aspect or height=width*aspect.) A file requester is then opened to choose the name and location of the movie, then the frames are written sequentially in the mov format. Variables are scaled between buffer frames to create the illusion of motion or morphing. The movie is written in the highest quality possible, so there are minimal compression artifacts. (The movie can be compressed later in an external QuickTime program to reduce file size, if necessary.) Most variables that have a numerical value can be scaled between frames.

8.2 Add Frame

iViZionaire uses a frame buffer to compose an animation. You add key frames to the buffer with this command. Each key frame is identical to the active image. Change variables between key frames to create the illusion of motion or morphing. You can edit the frames with the frame editor.

8.3 Edit Frames

Here you can edit any frames in the video buffer. Buttons are supplied to access all the image parameter editors, such as Function, Lighting and Parameters. The Move button allows you to move a frame from one spot in the buffer to another. You can change the frame image being edited by using the Frame slider or Edit box. After changing frames, use the Preview button to display the current frame being edited. In most cases the frame preview is automatically updated when you

change an image parameter using the editor or type buttons. The Delete button allows you to delete all but two of the frames, the minimum number of frames to create a movie. (If you want to delete all the frames, use the Video/Reset Frames command.)

8.4 Reset Frames

Delete the current frame buffer. The number of video frames is reset to zero.

8.5 Load Frames [VFRM]

Load a frame buffer that has been previously saved by iViZionaire. The buffer replaces any existing frame buffer.

8.6 Save Frames [VFRM]

This command saves the current frame buffer in a [vfrm] file. A file requester is opened that allows you to choose the location and name of the frame library. The frame buffer files can also be used as image libraries, similar to Fractint's par and frm formats. The frames contain all the information to reproduce an image at any supported size.

9 Help Topics

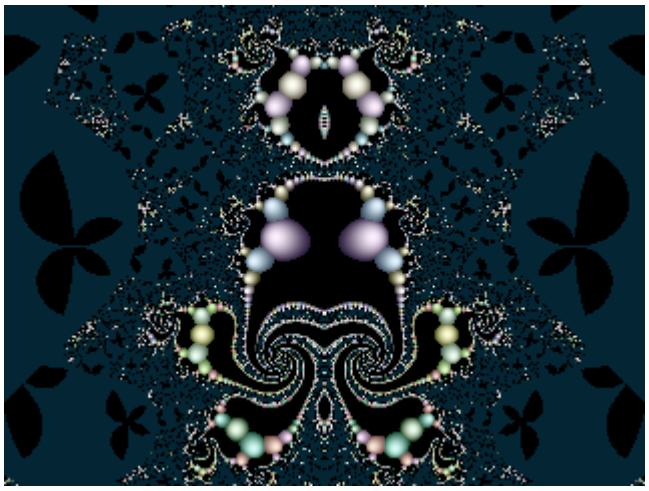
Help topics

Getting Started	Tutorial for new users of iViZionaire.
Tutorial	An introduction to CQuat fractals
Parser Information	Quick reference to iViZionaire's parser variables and functions.
Built-in Formulas	Quick reference to iViZionaire's built-in formulas.
Bibliography	Sources for fractal information and complex numbers.
About iViZionaire	Displays the version number and author info for this application.
Chronology	History of the programs preceding iViZionaire

9.1 Getting Started

Getting Started

Welcome to iViZionaire!



This is a short tutorial that will cover basic commands and background material necessary for a user to create a drawing or model with iViZionaire.

Start by changing the type of the fractal to Mandelbrot or MandelbrotP using those commands in the Type menu. The Mandelbrot border is often used as a map to find the most interesting Julia sets. Use Image/Scan command to turn on quaternion/Julia exploratory mode. If you left-click inside the draw window, the left-hand corner of the image will be erased, and a miniature version of the Julia set that uses that zspace as its constants will be drawn. You can continue to click on areas of the draw window and see what Julia set lies there, or you can press the space bar to open a new window and draw the Julia set full-size. Click in the status_bar area of the window to exit this mode without creating a new window.

Once you find an interesting 2D Julia figure, you can change it into a 3D figure just by changing the Type to Quaternion or one of the other 3D types. This figure can be converted to a 3D object by using any of the Export commands in the File menu, such as Save Quaternion to [obj]. The precision of the object is controlled by the Steps variable in the Parameters window, so to get a smooth object you'll need to increase the number of steps from its 200 default value. For a fully defined object Steps may need to be increased to 800 or more. Be careful with this though, as the size of the object file can increase exponentially depending on the Export setup.

Other 3-D Types in iViZionaire include hypernions and cquats. It's easy to switch from quaternion to hypernion using the Type menu. Most hypernions exhibit squared-off shapes rather than the round quaternion shapes. For a good preview of what iViZionaire is capable of, be sure to experiment with all of the Demo/Random commands.

iViZionaire allows you to Undo the last command in most cases.

This completes the Getting Started tutorial. Be sure to read the Parser Information and Built-in Formulas sections for additional info. The Bibliography lists additional reference material for a better understanding of the fractal types and functions contained in iViZionaire.

9.2 Tutorial

An Introduction To CQuat Fractals By Terry W. Gintz

In the process of exploring all possible extensions to a fractal generator of this type, I considered using discrete modifications of the standard quaternion algebra to discover new and exciting images. The author of *Fractal Ecstasy* [6] produced variations of the Mandelbrot set by altering the discrete complex algebra of z^2+c. The extension of this to quad algebra was intriguing. There was also the possibility of different forms of quad algebra besides quaternion or hypercomplex types.

Having modeled 3D fractals with complexified octonion algebra, as described in Charles Muses' non-distributive algebra [7], it was natural to speculate on what shapes a "complexified" quaternion algebra would produce. Would it be something that was between the images produced with hypercomplex and quaternion algebra? Quaternion shapes tend to be composed of mainly rounded lines, and hypercomplex shapes are mainly square (see Figures 1 and 2.)

Figure 1. Quaternion Julia set of -1+0i

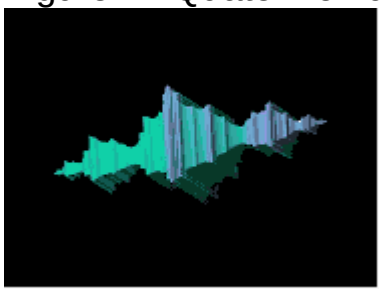


Figure 2. Hypercomplex Julia set of -1+0i

For those not familiar with the basics of hypercomplex and quaternion algebra, here are the algebraic rules that define how complex components interact with each other:

	i	*j*	*k*
i	-1	*k*	*-j*
j	*k*	-1	*-i*
k	*-j*	*-i*	1

Table 1 Hypercomplex variable multiplication rules

	i	*j*	*k*
i	-1	*k*	*-j*
j	*-k*	-1	*i*
k	*j*	*-i*	-1

Table 2 Quaternion variable multiplication rules

In both quaternion and hypercomplex algebra, i^2=-1. The hypercomplex rules provide for one real variable, two complex variables, (*i* and *j*) and one variable that Charles Muses refers to as countercomplex (*k*), since *k*k* = 1. It would appear from this that k = 1, but the rules in Table 1 show that k has complex characteristics. In quaternion algebra there is one real variable and three complex variables. In hypercomplex algebra, unlike quaternion algebra, the commutative law holds; that is, reversing the order of multiplication doesn't change the product. The basics of quaternion and hypercomplex algebra are covered in Appendix B of *Fractal Creations* [8]. One other concept important to non-distributive algebra is the idea of a "ring". There is one ring in quaternion and hypercomplex algebra (*i*,*j*,*k*). (There are seven rings in octonion algebra.) If you start anywhere in this ring and proceed to multiply three variables in a loop, backwards or forwards, you get the same number, 1 for hypercomplex, and 1 or -1 for quaternion, depending on the direction you follow on the ring. The latter emphasizes the non-commutative nature of quaternions. E.g. : using quaternion rules, *i*j*k* = *k*k* = -1, but *k*j*i* = *-i*i* = 1.

For "complexified" quaternion algebra, the following rules were conceived:

	i	*j*	*k*
i	-1	*-k*	*-j*
j	*-k*	1	*i*
k	*-j*	*i*	1

Table 3 CQuat variable multiplication rules

Note that there are two countercomplex variables here, *(j* and *k*). The commutative law holds like in hypercomplex algebra, and the "ring" equals -1 in either direction. Multiplying two identical quad numbers together, *(x+yi+zj+wk)(x+yi+zj+wk)* according to the rules of the complexified multiplication table, combining terms and adding the complex constant, the following iterative formula was derived for the "complexified" quaternion set, q^2+c:

*x -> x*x - y*y + z*z + w*w + cx*
*y -> 2.0*x*y + 2.0*w*z + cy*
*z -> 2.0*x*z - 2.0*w*y + cz*

*w -> 2.0*x*w - 2.0*y*z + cw*

Just to get a feel for this new formula, a fairly basic constant, -1+0i, was used for the initial 3D test. The extraordinary picture "Equilibrium"(Figure 3) was the result.

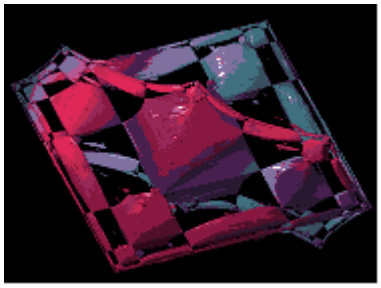


Figure 3. Equilibrium -- cquat Julia rendering of -1+0i

Being familiar with the quaternion and hypercomplex renditions of the Julia set -1+0i, it appeared that this image was a leap into hyperspace; the fractal seemed to literally expand in all directions at once. The next test used a Siegel disk constant, -.39054-.58679i, which Roger Bagula [9] had recently sent. The Siegel image (Figure 4) strongly suggested that cquats were indeed a new form of space-filling fractal.

Figure 4 Siegel -- cquat Julia rendering of -.39054-.58679i

Since then, Godwin Vickers has ported the cquat formula to the *Persistence of Vision Ray-tracer* [10], and verified that the equilibrium image wasn't just an artifact of QuaSZ. Nearly identical images have been obtained in POV, using Pascal Massimino's [11] custom formula algorithm for 3D Mandelbrot and Julia sets.

There remains the extension of cquat algebra to transcendental and exponential functions. Any ideas for this are welcome. The built-in formulas in QuaSZ include cquat variations where possible.

References

1. Hamilton, W. R. (1969) *Elements of Quaternions, Vol. I and II*, reprinted by Chelsea Publishing Co.: New York
2. Mandelbrot, B. (1983) *The Fractal Geometry of Nature,* Freeman, San Francisco.

3. Norton, A. (1982) Generation and display of geometric fractals in 3D, Computer Graphics (ACM-SIGGRAPH) July 23(3): 41-50.
4. Vickers, Godwin, http://www.hypercomplex.org
5. Gintz, T. W. (1989-2003) *Fractal Zplot*, originally Zplot.
6. *Fractal Ecstasy* (1993) Deep River Publishing, Inc.
7. Muses, C., http://www.innerx.net/personal/tsmith/NDalg.html
8. Tim Wegner and Bert Tyler (1993) *Fractal Creations*, Waite Group Press: CA.
9. Bagula, R., http://home.earthlink.net/~tftn/
10. POV Team, *Persistence of Vision Ray-tracer*, Victoria, Australia, http://www.povray.org/
11. Massimino, P., http://skal.planet-d,net/quat/Compute.ang.html#JULIA

9.3 Parser Information

Parser Information

Functions (capital letters are optional, and parenthesis are necessary around complex expressions)

The following information takes the form "standard function" ---"form used by iViZionaire to represent standard function".

sine z --- sin(z) or SIN(Z) ; where Z can be any quad expression or variable
hyperbolic sine z --- sinh(z) or SINH(Z)
arcsine z --- asin(z) or ASIN(Z)
cosine z --- cos(z) or COS(Z)
hyperbolic cosine z --- cosh(z) or COSH(Z)
arccosine z --- acos(z) or ACOS(Z)
tangent z --- tan(z) or TAN(Z)
hyperbolic tangent z --- tanh(z) or TANH(Z)
arctangent z --- atan(z_) or ATAN(Z)
cotangent z --- cotan(z) or COTAN(Z)
arccotangent z --- acotan(z) or ACOTAN(Z)
e^z --- exp(z) or EXP(z) -- the exponential function
natural log of z --- log(z) or LOG(Z)
absolute value of z --- abs(z) or ABS(Z)
square root of z --- sqrt(z) or SQRT(Z)
z squared --- sqr(z) or SQR(Z)
real part of z --- real(z) or REAL(Z)
imaginary part of z --- imag(z) or IMAG(Z)
'j' or third component of z --- imaj(z) or IMAJ(Z)
'k' or fourth component of z --- imak(z) or IMAK(Z)
modulus of z --- mod(z) or MOD(Z) or |z| -- (x*x + y*y)
conjugate of z -- conj(z) or CONJ(z) -- (x-yi)
flip(z) --- flip(z) or FLIP(Z) -- exchange real and imaginary parts of z (y+xi)
polar angle of z -- theta(z)
fn1(z) -- fn1(z) or FN1(Z) -- user-defined function from f1 list box

fn2(z) -- fn2(z) or FN2(z) -- user-defined function from f2 list box
fn3(z) -- fn3(z) or FN3(z) -- user-defined function from f3 list box
fn4(z) -- fn4(z) or FN4(z) -- user-defined function from f4 list box

if/then/endif – if(argument), then (phrase) endif -- if argument is true then do phrase else skip phrase('then' tag is optional, but a comma should follow argument or put 'if(argument)' on separate line)

if/then/else/endif - if(argument), then (phrase) else (phrase) endif -- if argument is true then do phrase else skip phrase and do alternate phrase('then' tag is optional, but a comma should follow argument or put 'if(argument)' on separate line)

Note: if/then/endif and if/then/else/endif loops can be nested only when endifs follow each other at the end of the loops. For example: if(argument) if(argument) then (phrase) endif endif.

Math operators

+ --- addition
- --- subtraction
* --- multiplication
/ --- division
^ --- power function
< --- less than
<= --- less than or equal to
> --- greater than
>= --- greater than or equal to
!= --- not equal to
== --- equal to
|| --- logical or (if arg1 is TRUE(1) or arg2 is TRUE)
&& --- logical and (if arg1 is TRUE and arg2 is TRUE)

Constants and variables

complex constant --- c or C, read/write.
complex conjugate --- cc# or CC#, read-only.
e --- e or E -- 1e^1 -- 2.71828, read/write.
i --- i or I -- square root of -1, read-only. This is equivalent to the quad constant 0+1i+0j+0k.
iteration --- iter# -- iteration loop counter
j --- j or J -- third component of quad constant, read-only. This is equivalent to the quad constant 0+0i+1j+0k.
k --- k or K -- fourth component of quad constant, read-only. This is equivalent to the quad constant 0+0i+0j+1k.
m --- m# or M# or pixel --a complex variable mapped to the pixel location as defined by the z coordinates entered in the Parameters window, read/write.
maxit -- the maximum number of iterations, as set in the Parameters window, read only
p --- p# or P# -- the real part of the complex constant, as entered in the cr box, read-only.

p1 – the quad constant entered in the cr, ci, cj and ck boxes, read-only.
pi --- pi or PI -- 3.14159, read/write.
q --- q# or Q# -- the imaginary part of the complex constant, [ci box] evaluated as a real number, read-only.
x --- x# or X# -- real part of Z, read/write.
y --- y# or Y# -- coefficient of the imaginary part of Z, read/write.
z --- z or Z -- function value at any stage of the iteration process, read/write.
zn# or ZN# -- the value of z at the previous stage of iteration, read-only.

9.4 Built-in Formulas

Built-in Formulas (select the following prefix in the Fun #1 or Fun #2 popup controls)

0 -- z^2+c --- the standard Mandelbrot or Julia set.
1 -- cz(1-z) --- the self-squared dragon set.
2 -- c(z-1/z) --- alternate Mandelbrot or Julia set.
3 -- cz^2-1 --- alternate Mandelbrot or Julia set.
4 -- c^2/(c+z^2) --- alternate Mandelbrot or Julia set.
5 -- z^3+c --- cubic Mandelbrot or Julia set.
6 -- ((z^2+c-1)/(2z+c-2))^2 -- renormalization formula #1 for x-plane or q-plane pictures.
7 -- z^2+j+kzn --- Phoenix curve (Ushiki).
8 -- Julia/Mandelbrot set (modified from M. Barnsley).
9 -- fn(z)-cfn(z) -- generalized frothy basin (J. Alexander.)

10 -- Newton/Halley map of z^3+conj(z)c -- exploratory function based on modified frothy basin.
11 -- z^z+z^s+c
12 -- z^s-z+c
13 -- fn(z)+exp(z)+c
14 -- solves Newton/Halley transformation of (z^2-c)(z+1).
15 -- cfn(z) -- transcendental Julia curve.
16 -- cexp(z) -- exponential Julia curve, etc. with additional plane checking when real value of Z exceeds 50. If cos(imag-Z) >= 0, point is considered part of Julia set.
17 -- fn(z)+cfn(z)+1 -- generalized form of t9.
18 -- foggy coastline #1 Mandelbrot IFS (M. Barnsley).
19 -- foggy coastline #2 Mandelbrot IFS (M. Barnsley).

20 -- solves Newton/Halley transformation of (z+j)(z+k)(z^2+1)
21 -- solves Newton/Halley transformation of (z+j)(z^2+z+k)
22 -- solves Newton/Halley transformation of (z-1)(z^2+z+c)
23 -- solves Newton/Halley transformation of (z+j)(z+k)(z+1)
24 -- Chaos Game Julia IFS (M. Barnsley).
25 -- snowflake Julia IFS (as described in Fractals Everywhere by M. Barnsley).
26 -- solves Newton/Halley transformation of logz-c.
27 -- solves Newton/Halley transformation of exp(z)-c.
28 -- solves Newton/Halley transformation of (z-c)(z+1)(z-1)

29 -- solves Newton/Halley transformation of (z-c)(z+c)(z^2+c^2) --- z^4-c^4.

30 -- solves Newton/Halley transformation of sinz-c.
31 -- sexpz+c -- transcendental Julia set.
32 -- c(1+z^2)^2/(z^2-1) -- alternate Julia set.
33 -- solves Newton/Halley transform of tan(z)-c .
34 -- IFS (x=sy+j,y=-sx+k (x>0); else x=sy-j,y=-sx-k (modified from M. Barnsley)
35 -- solves Newton/Halley transform of z^s-1.
36 -- composite function cz-c/z & z^2+c (C. Pickover).
37 -- transcendental function fn(z)+c .
38 -- ((z^3+3(c-1)z+(c-1)(c-2))/(3z^2+3(c-2)z+c^2-3c+3))^2 -- renormalization formula #2 for x-plane or q-plane pictures .
39 -- Newton/Halley map of z(z^limit-1).

40 -- Newton/Halley map of z(z^limit-c)
41 -- Newton/Halley map of Chebyshev function cos(n*arccos x).
42 -- Newton/Halley map of Hermite polynomial: 16x^4-48x^2+12.
43 -- alternate Newton/Halley map of tanz-c
44 -- Newton/Halley map of z^limit-c
45 -- fn(fn(z))+c -- user-defined complex set. When the first function is z^2 and the second function is conj(z), this becomes the z-conjugate set, zz^2+c, the tricorn set.
46 -- Volterra-Lotka equations discretized by modified Huen method (from The Beauty of Fractals).
47 -- c^z+c -- tetration of z.
48 -- q^2+c -- Quaternion set (from Computer, Pattern, Chaos and Beauty)
49 -- z+cz+1

50 -- spiral network -- C. Pickover.
51 -- z-(1/z+c) -- try with renormalization applied. Sequel to t9.
52 -- fn(z)-(fn(z)+c) -- generalized form of a1.
53 -- alternate Newton/Halley map of sinz-c.
54 -- user-defined complex set: fn(z)+fn(z)+c.
55 -- hypercomplex Newton/Halley map of h^3+c.
56 -- Hypercomplex Newton/Halley map of fn(h)+c.
57 -- user-defined complex set: fn(z)+fn(c).
58 -- fn(z)+zn+c -- from Fractal Creations.
59 -- q^3+c -- cubic Quaternion set.

60 -- alternate Newton/Halley map of exp(z)-c.
61 -- alternate Newton/Halley map of log(z)-c.
62 -- Newton/Halley map of phoenix curve.
63 -- cfn(z)+zn -- user-defined complex formula.
64 -- fn(z)+kzn+j -- generalized phoenix curve formula.
65 -- fn(z) a preformula for use with type 3 composite fractals. uses limit gadget to select function.
66 -- Newton/Halley map of fn(z)+fn(z)+c.
67 -- Newton/Halley map of cfn(z).

68 -- fn(z)*fn(z)+c.
69 -- Newton/Halley map of foggy coastline #1.

70 -- Newton/Halley map of foggy coastline #2.
71 -- Newton/Halley map of fn(fn(z))+c.
72 -- cfn'(z), where fn'(z)=first derivative of user-defined function.
73 -- fn(z)+fn'(z)+c.
74 -- fn'(z)+fn(c).
75 -- fn(fn'(z))).
76 -- first order gamma function: (z/e)^z*sqr(2*pi*z)+c.
77 -- Newton/Halley map of fifth degree Legendre polynomial: 1/8(63z^5-70z^3+15z
78 -- (z^2+e^(-z))/(z+1): second-order convergence formula for finding root of ze^z-1=0.
79 -- Newton/Halley map of fn(z)*fn(z)+c.

80 -- z^s/limit+c: anti-derivative of z^n
81 -- Sterling expansion of gamma function: (z/e)^z*sqr(2*pi/z)+c.
82 -- Newton map of fn'(z)-fn(z)+c: generalized first degree Laguerre polynomial.
83 -- fn(1/(fn(z)+c)).
84 -- z^2-c; where zreal=abs(zreal) (Paul Carlson's "alien" Julia set).
85 -- z^2; where zreal=abs(zreal)-cr, zimag=zimag-ci (Paul Carlson Julia set).
86 -- cfn(z)+c.
87 -- Newton's method applied to (x^3+y^2-cr=0 and y^3-x^2+ci=0). from Sylvie Gallet and Fract19.par.
88 -- Newton's method applied to fn1(x)+fn2(y)-cr=0 and fn3-fn4+ci=0
89 -- Bill13 from Bill Rossi via the Internet.

90 -- generalized form of Earl Hinrichs' sophomore sine function(ssin) -- limit*fn(z)+s+si, where fn(z)=(fn1(x),fn2(y)).
91 -- c^2/(1-cz^2) -- variant of p4.
92 -- Gallet-4-01, from Sylvie Gallet's extensive Internet collection
93 -- Gallet-4-02, from Sylvie Gallet.
94 -- Gallet-6-01, from Sylvie Gallet.
95 -- Gallet-6-02, from Sylvie Gallet.
96 -- Gallet-6-03, from Sylvie Gallet.
97 -- Gallet-6-04, from Sylvie Gallet.
98 -- Gallet-6-05, from Sylvie Gallet.
99 -- Gallet-7-01, from Sylvie Gallet.

Notes: the term 'fn(w)' represents any one of 47 user-defined functions.

0: sin(w).	1: sinh(w).	2: cos(w).	3: cosh(w).
4: tan(w).	5: tanh(w).	6: exp(w).	7: ln(w).
8: w^c	9: w^z.	10: 1/w.	11: w^2.
12: w^3.	13: abs(w).	14: sqrt(w).	15: w.
16: conj(w).	17: csc(w).	18: csch(w).	19: sec(w).
20: sech(w).	21: cot(w).	22: coth(w).	23: cw.
24: 1.	25: arsin(w).	26: arcsinh(w).	
27: arccos(w).	28: arccosh(w).	29: arctan(w).	

30: arctanh(w). 31: arccot(w). 32: arccoth(w).
33: vers(w). 34: covers(w). 35: L_3(w): 3rd degree Laguerre polynomial. 36: gamma(w): first order gamma function.
37: G(w): Gaussian probability function -- (1/sqr(2pi))*e^(.5w^2).
38: c^(s+si). 39: zero. 40: w^(s+si). 41: |(wx)|+|(wy)|*i(abs).
42: wy+wx*i(flip). 43: conj(cos(w))--cosxx. 44: theta(w) -- polar angle(w).
45: real(w). 46: imag(w).
When only fun#1 or fun#2 is used and a single user-defined function is involved, the function is taken from f1. When two user-defined functions appear in a function, the f2 gadget supplies the second function type, except as noted below. For plots that use both fun#1 and fun#2(type 2 or 3, etc), fun#1 takes its functions from f1 and f2 and fun#2 takes its functions from f3 and f4.

The Halley map requires the Newton flag to be set. This is another numerical approximation method for finding complex roots. For all Newton/Halley functions, the Newton map is the default. The Halley option is specified through the Arg Gadget, the second character being set to 'h', after the display method(1-9). E.g. '1hr' would designate a relaxed Halley map with display method 1.

Halley and Newton maps can use one of nine display methods:
#1 (the default mode, except for functions using sinz, expz, logz and tanz, or fn(z), which default to method 2, and don't use methods 1,4 or 5): ---colors represent the root(the zero) which a point converges to.
#2 (if the Arg Gadget is set to 2, or for functions of sinz, tanz, logz, expz, and fn(z)): ---colors represent the number of iterations a point takes to converge.
#3 (if the Arg Gadget is set to 3) -- colors represent the number of iterations a point takes to converge according to an alternate formula described by C. Pickover in Computers, Pattern, Chaos and Beauty.
#4 (if the Arg Gadget is set to 4) -- a merging of methods 1 and 3. After the point converges according to the alternate formula #3, its roots are colored according to #1.
#5 (if the Arg Gadget is set to 5) -- a variation of method 1, with double-convergence checking inside the loop.
#6, #7 and #8-- alternate convergent formulas.
#9 -- a variation of method 3, with double-convergence checking.

An additional third argument that affects the convergence speed of Newton/Halley maps may be one of the following six methods:
'r': relaxed Newton method uses the formula z = z-sf(z)/f'(z).
'm': modified Newton transformation uses the formula: z-(f(z)/(f'(z)+si)). Note: Si here references the s variable * i, not the complex variable s+si.
'd': relaxed modified Newton method uses the formula: z-(sf(z)/f'(z)+si)).
'p': premodified Newton transform uses the formula: sz-(f(z)/f'(z)).
'c': complex Newton transform uses the formula : z-(f(z)/(f'(z)+c)), where c is the complex constant.
'n': Nova variation by Paul Derbyshire, z-(f(z)/(f'(z))+c.

The s constant is entered via the S gadget.

Renormalization functions use the Arg Gadget for plotting options (1-4,6-8) as follows:

0 or 1: default renormal, with anti-ferromagnetic points mapped only for Julia sets. Paramagnetic points (those converging to 1) and ferromagnetic points (those escaping to infinity) are mapped for both Mandelbrot and Julia sets.

2: anti-ferromagnetic points are mapped for the Mandelbrot set. This is actually a level-set mapping for points that do not escape to infinity or converge to 1.

3: uses an alternate convergence formula for paramagnetic and anti-ferromagnetic points.

4: a combination of methods 0 and 3, with characteristics of both methods appearing in plot.

6-8: alternate convergence methods, same as those used with Newton/Halley maps

An optional argument for renormalization 'n' follows the convergence method. This is an alternate bailout method for ferromagnetic points.

Most of the built-in functions (except for real Newtons and the Gallet formulas) have hypercomplex extensions when values of cj, ck, hj or hk are non-zero.

Hypercomplex Newton/Halley maps use only type 2 and type 3 convergence tests.

The default version of hypercomplex conjugate is defined as conjugate(h)=hr-hi-hj+hk. A variant of the hypercomplex conjugate uses an arg limit with a non-integral value (e.g.: 2.1.) This makes all imaginary components of h negative, such that conjugate(h)=hr-hi-hj-hk.

The formula for a first degree Laguerre polynomial is e^t(d/dt(t/e^t))=d/dt(t)-t.

For real Newtons, the function selected from the f1-f4 boxes is a real function. For a real conjugate, the negation of the real term is used.

Formulas by Sylvie Gallet have been modified to allow both Mandelbrot and Julia sets to be drawn from them. Except for Gallet-6-02, the bailout is set with the built-in variable Bailout in the Parameters window. Gallet-6-02 uses the Arg Limit variable for bailout (a small value <1).

9.5 Bibliography

Bibliography

Complex Mathematics

Churchill, Ruel.V. and Brown, James Ward: "Complex Variables and Applications", Fifth Edition, McGraw-Hill Publishing Company, New York, 1990.

Korn, Granino A. and Korn, Theresa M.: "Manual of Mathematics, McGraw-Hill Publishing Company, New York, 1967.

Fractal Theory

Barnsley, Michael: "Fractals Everywhere", Academic Press, Inc., 1988.

Devaney, Robert L.: "Chaos, Fractals, and Dynamics", Addison-Westley Publishing Company, Menlo Park, California, 1990.

Mandelbrot, Benoit B.: "The Fractal Geometry of Nature", W.H.Freeman and Company, New York, 1983.

Peitgen, H.-O. and Richter, P.H.: "The Beauty of Fractals", Springer-Verlag, Berlin Heidelberg, 1986.

Formulas and Algorithms

Burington, Richard Stevens: "Handbook of Mathematical Tables and Formulas", McGraw-Hill Publishing Company, New York, 1973.

Kellison, Stephen G.: "Fundamentals of Numerical Analysis", Richard D. Irwin,Inc. Homewood, Illinois, 1975.

Peitgen, Heinz-Otto and Saupe, Deitmar: "The Science of Fractal Images", Springer-Verlag, New York, 1988.

Pickover, Clifford A.: "Computers, Pattern, Chaos and Beauty", St. Martin's Press, New York, 1990.

Stevens, Roger T.: "Fractal Programming in C", M&T Publishing, Inc.,Redwood City, California, 1989.

Wegner, Tim, Tyler,Bert, Peterson, Mark and Branderhorst, Pieter: "Fractals for Windows", Waite Group Press, Corte Madera, CA, 1992.

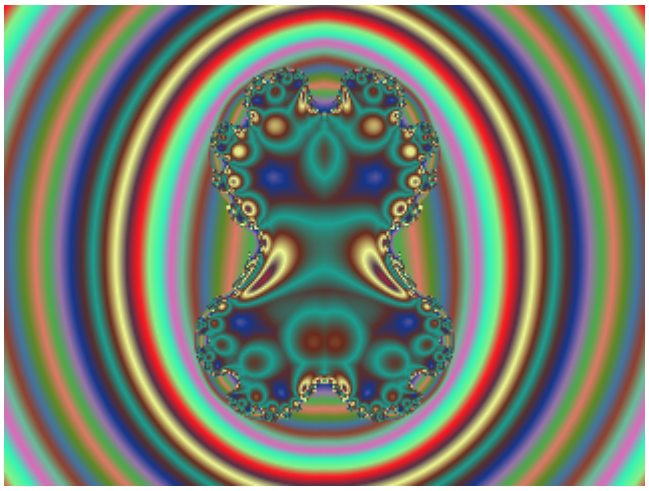
Wegner, Tim and Tyler, Bert: "Fractal Creations", Second Edition, Waite Group Press, Corte Madera, CA, 1993.

Whipkey, Kenneth L. and Whipkey, Mary Nell: "The Power of Calculus", John Wiley & Sons, New York, 1986.

9.6 About iViZionaire

About iViZionaire

>>>>> iViZionaire™ v1.0 © 2005-2008 Mystic Fractal <<<<<



IViZionaire graphs formulas based on 4D complex number planes. IViZionaire supports the Mandelbrot set, Julia sets, and Phoenix curves, with numerous mapping variations. 3D plot types include quaternion, hypernion, cquats (complexified quaternion). The complex math functions supported include sin(z), sinh(z), z^z, e^z, z^n, sqrt(z), cos(z), cosh(z), tan(z), tanh(z), log(z), ln(z), n^z and others. Random image generators and a random formula generator make the program easy for beginners and a powerful compliment to advanced fractal artists.

Up to two formulas for z using the above functions may be plotted, using traditional rules for generating Mandelbrot sets (Benoit Mandelbrot) and Julia sets (G. Julia.) Also, there are mapping options that use non-traditional methods, such as composite formulas and IFS (Michael Barnsley).

100 formulas have been hard-coded to reduce graphing time by 40 to 60 percent. Hypercomplex extensions (as described in Fractal Creations) are standard for most of the formulas.

iViZionaire requires a true-color video adapter for best results.

Acknowledgements: Many thanks to Paul Carlson for providing me his algorithms for 3D-like fractals, and allowing me to incorporate his original ideas into my programs. Special thanks to Frode Gill for his quaternion and ray-tracing algorithms and to Dirk Meyer for his Phong-shading algorithm.

For a short history of the programs preceding iViZionaire, see Chronology.

9.7 Chronology

Chronology

History of the programs preceding iViZionaire:

In September 1989, I first had the idea for a fractal program that allowed plotting all complex functions and formulas while attending a course on College Algebra at Lane College in

Eugene, Oregon. In November 1989, ZPlot 1.0 was done. This Amiga program supported up to 32 colors, 640X400 resolution, and included about 30 built-in formulas and a simple formula parser.

May 1990 -- ZPlot 1.3d -- added 3-D projections for all formulas in the form of height fields.

May 1991 -- ZPlot 2.0 -- first 236-color version of ZPlot for Windows 3.0.

May 1995 -- ZPlot 3.1 -- ZPlot for Windows 3.1 -- 60 built-in formulas. Added hypercomplex support for most built-in formulas.

May 1997 -- ZPlot 24.02 -- first true color version of ZPlot -- 91 built-in formulas. Included support for 3-D quaternion plots, Fractint par/frm files, Steve Ferguson's filters, anti-aliasing and Paul Carlson's orbit-trap routines.

June 1997 -- ZPlot 24.03 -- added Earl Hinrichs Torus method.

July 1997 -- ZPlot 24.08 -- added HSV filtering.

December 1997 -- Fractal Elite 1.14 -- 100 built-in formulas; added avi and midi support.

March 1998 --Split Fractal Elite into two programs, Dreamer and Medusa(multimedia.)

April 1998 -- Dofo 1.0 -- supports new Ferguson/Gintz plug-in spec.

June 1998 -- Dofo-Zon -- redesigned multi-window interface by Steve Ferguson, and includes Steve's 2-D coloring methods.

August 1998 --Dofo-Zon Elite -- combination of Fractal Elite and Dofo-Zon

October 1998 -- Dofo-Zon Elite v1.07 -- added orbital fractals and IFS slide show.

November 1998 -- Dofo-Zon Elite v1.08 -- added lsystems.

April 1999 -- Split Dofo-Zon Elite into two programs: Fractal Zplot using built-in formulas and rendering methods, and Dofo-Zon to support only plug-in formulas and rendering methods.

May 1999 -- Fractal Zplot 1.18 -- added Phong highlights, color-formula mapping and random fractal methods.

June 1999 -- completed Fractal ViZion -- first version with automatic selection of variables/options for all fractal types.

July 1999 -- Fractal Zplot 1.19 -- added cubic Mandelbrot support to quaternion option; first pc fractal program to render true 3-D Mandelbrots.

September 2000 -- Fractal Zplot 1.22 -- added support for full-screen AVI video, and extended quaternion design options

October 2000 -- QuaSZ (Quaternion System Z) 1.00 -- stand alone quaternion/hypernion/cubic Mandelbrot generator

November 2000 -- Added octonion fractals to QuaSZ 1.01.

March 2001 -- Cubics 1.0 -- my first totally-3-D fractal generator.

May 2001 -- QuaSZ 1.03 -- added Perlin noise and improved texture mapping so texture tracks with animations.

June 2001 -- Fractal Zplot 1.23 -- added Perlin noise and quat-trap method.

July 2001 -- QuaSZ 1.05 -- improved performance by converting many often-used dialogs to non-modal type.
November 2001 -- DynaMaSZ 1.0, the world's first Dynamic Matrix Systems fractal generator

January 2002 -- MiSZle 1.1 -- generalized fractal generator with matrix algebra extensions

May 2002 -- DynaMaSZ SE 1.04 (unreleased version)-- scientific edition of DMZ, includes support for user-variable matrix dimensions (3X3 to 12X12)

January 2003 -- PodME 1.0 -- first stand-alone 3-D loxodromic generator, Hydra 1.0 -- first 3-D generator with user-defined quad types and Fractal Projector a Fractal ViZion-like version of DMZ SE limited to 3X3 matrices

May 2003 -- QuaSZ 3.052 -- added genetic-style function type and increased built-in formulas to 180. Other additions since July 2001: generalized coloring, support for external coloring and formula libraries, and Thomas Kroner's loxodromic functions.

May 2003 -- FraSZle and Fractal Zplot 3.052 -- added random 3D orbital fractals, new 3D export methods, upgraded most frequently-used dialogs to non-modal type and added genetic-style function type. FZ now based on FraSZle except for built-in formula list and Newton support.

July 2004 – Added the features of Hydra, Cubics and PodME to QuaSZ, now renamed "Quad Surface Zplot". Merged FraSZle with Fractal Zplot, and Fractal Projector with DynaMaSZ SE to form DynaMaSZ 2, including support for the original DynaMaSZ files.

Index

- B -

- C -

- D -

- E -

- F -

- H -

- I -

- L -

- M -

- P -

- R -

- T -