QuaSZ Mac Application Help

# Table of Contents

# 1    Main Index

## QuaSZ Mac Help Index
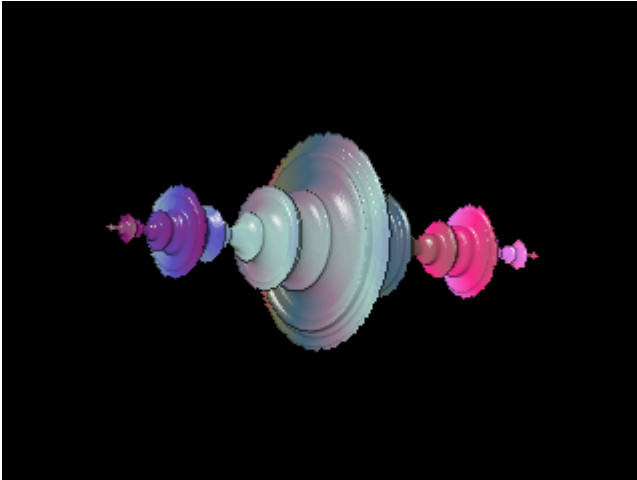
**Getting Started**

**An Introduction To CQuat Fractals by Terry W. Gintz**

**Menus**

    File menu
    Edit menu
    Image menu
    Type menu
    Demo menu
    Video menu

**Additional**

    Help topics

# 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. |

## 2.1    New command

**New command (File menu)**

Use this command to create a new drawing window in QuaSZ MAC.  The image and data file for the opening picture (title.png and title.qsz) 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 open multiple image files at once.

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 QuaSZ MAC stores the file that you want to open.
**Directories**
Select the directory in which QuaSZ MAC 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, QuaSZ MAC displays the <u>Save As dialog box</u> 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 <u>Save As command</u>.

## 2.5 Save As command

**Save As command (File menu)**

Use this command to save and name the active drawing. QuaSZ MAC displays the <u>Save As dialog box</u> so you can name your drawing.

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

### 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. QuaSZ MAC adds the extension .qsz.

**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].

# 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. (The Arg Limit variable is the 4th Center variable for Cubic Mandelbrot 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 <u>Parser Information</u>.
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.  QuaSZ Mac 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 QuaSZ Mac users to more easily convert Fractint formula types to QuaSZ Mac 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 QuaSZ Mac, 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.

Octonions (built-in functions 110-119) have a form of xr+xi+xj+xk+xE+xI+xJ+xK.
Additional options are entered via the Arg box.  To rotate the extra four octonion dimensions (E-K) use the following syntax:

|  |  |  |
|---|---|---|
| 0I | -- | rotate OE-OK to OI-OE |
| 0J | -- | rotate OE-OK to OJ-OI |
| 0K | -- | rotate OE-OK to OK-OJ |

To normalize the C and CC constants:

|  |  |  |
|---|---|---|
| n | -- | normalize C and CC (default is un-normalized) |

Alternate octonion initialization:

|  |  |  |
|---|---|---|
| c | -- | set OE-OK to .01 at beginning of each iteration |

## 3.3    Parameters

**Parameters**

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

Quad constants are cr, ci, cj and ck. Additional constants cE-cK are provided for octonion formulas.

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

A larger bailout can increase the fullness of a quad figure. 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 on loxodromic and cquat fractals.

Three rotate variables determine the 3-D 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 QuaSZ Mac.

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 QuaSZ Macpalette.

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 QuaSZ Mac, 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 |

## 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.)

## 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.
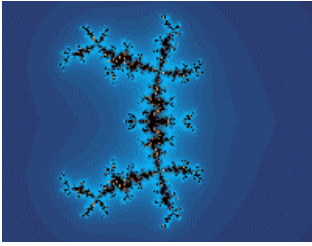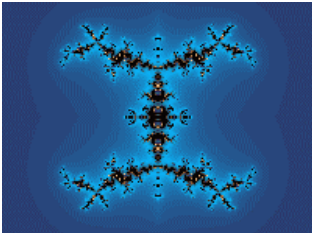
## 4.6 Symmetry

**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.
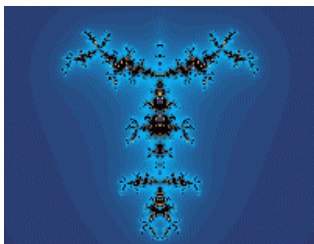


Vertical symmetry

Horizontal symmetry

XY symmetry

# 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. |
| Quaternion | Set fractal type to quaternion. |
| Hypernion | Set fractal type to hypercomplex quaternion. |
| Cubic Mandelbrot | Set fractal type to cubic Mandelbrot. |
| Complexified Quaternion | Set fractal type to 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.
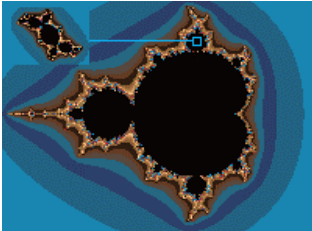
## 5.2     MandelbrotP

**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.
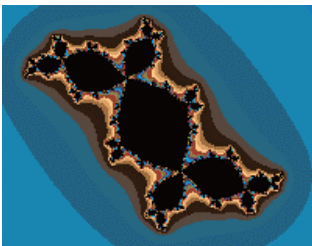
## 5.3    Julia

**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    Quaternion command

**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    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.  Cubic and octonion formulas can produce shapes with the Hypernion type that aren't like any standard formula.

## 5.6    Cubic command

**Cubic (Type menu)**

Use this command to set the fractal type to a cubic Mandelbrot.  Built-in formulas that support this type are 100-109.

Cubic Mandelbrots quaternions use the S (real) and S (imag) variables to set the initial value

and range of the 3rd dimension.  Starting from an initial value of 'S' (real), S (imag) is normally set to 2*abs (S), when the Type is Mandelbrot.  S (imag) can also be set to center the 3rd dimension on something besides 0.0.

In addition, the Arg value (in the Functions window) has the following affect on cubic Mandelbrots:
0 -- compute M+, using Z for z space
1 -- compute M+, using greater of S or Z for z space
2 -- compute M-, using greater of S or Z for z space
3 -- compute M+ and M-, use lesser of M+/M- for pixel depth
4 -- compute M+ and M-, use greater of two for pixel depth
5 -- compute M+ and M-, use difference of two for pixel depth
6 -- compute M+ and M-, use sum of two for pixel depth
7 -- compute M+ and M-, use vector magnitude of two for pixel depth
8 -- compute M+ and M-, use intersection of two (CCL)
9 -- compute M+ and M-, use M+ or difference of two if M+>M-
Args 3-9 affect only cubic Mandelbrots.

For args 0-9, a second argument determines the plane that is used for the fourth dimension:
b -- b-imag (default)
B -- b-real
a -- a-imag
A -- a-real

A third argument also works for args 0-9:
j -- alternate cubic Mandelbrot mapping

Therefore an arg value of '3B' uses the union of M+ and M- and b-real as the fourth dimension.

## 5.7    Complexified Quaternion command

### 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.

# 6    Render Menu

### Render menu commands

The Render menu offers the following commands:

## 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    Palette Coloring

**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    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.

## 6.4   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.

## 6.5   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.

### 6.5.1   Blend buttons

Various formulas are applied while mapping colors to pixels.

### 6.5.2   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.

### 6.5.3   RGB button

**RGB button**

Use red/green/blue mapping for pixels.

### 6.5.4   RBG button

**RBG button**

Use red/blue/green mapping for pixels.

### 6.5.5 GRB button

**GRB command (Render menu)**

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

### 6.5.6 GBR button

**GBR command (Render menu)**

Use green/blue/red mapping for pixels.

### 6.5.7 BRG button

**BRG command (Render menu)**

Use blue/red/green mapping for pixels.

### 6.5.8 BGR button

**BGR command (Render menu)**

Use blue/green/red mapping for pixels.

### 6.5.9 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.

### 6.5.10 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.

### 6.5.11 Gray Scale button

**Gray Scale button**

Color the active image with gray tones.

### 6.5.12 Invert button

**Invert command (Render menu)**

Invert image colors.

### 6.5.13  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 QuaSZ Mac:

Random Quaternion              Generate random quad fractal.
Random Cubic Mandelbrot    Generate random cubic Mandelbrot fractal
Random Cubic Julia             Generate random cubic Julia fractal.
Random Octonion               Generate random octonion fractal.
Random Composite             Generate random composite quad fractal.
Random Setup                    Set preferences for random commands.

## 7.1      Random Quaternion command

**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.

## 7.2      Random Cubic Mandelbrot command

**Random Cubic Mandelbrot (Demo menu)**

A random cubic Mandelbrot fractal is generated.  Like the inverse of a Random Quaternion Julia set, the essential cubic parameters are randomly adjusted to point into a four-dimensional formula 100-109.

## 7.3      Random Cubic Julia command

**Random Cubic Julia (Demo menu)**

A random cubic Julia fractal (the Julia analog of a cubic Mandelbrot fractal) is generated. The essential cubic parameters are randomly adjusted to point into a four-dimensional formula.  Like Random Quaternion, a set of formulas (100 -109) appropriate for cubic Julias is scanned to find an interesting Julia set, and then the parameters are adjusted to produce the cubic image.  Note:  This is a quasi-Julia approximation that doesn't follow traditional cubic Mandelbrot theory.  The "quaternions" produced by this method do exhibit characteristics of cubic Mandelbrots, but here I am more interested in esthetics than mathematical conformity.

## 7.4    Random Octonion command

**Random Octonion (Demo menu)**

A random octonion Julia fractal is generated.  The essential octonion parameters are randomly adjusted to point into an eight-dimensional formula 110-119, and then the octonion Mandelbrot set is scanned to find an interesting area to zoom into.

## 7.5    Random Composite command

**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, including cubic Mandelbrot and octonion formulas.

## 7.6    Random Setup

**Random Setup (Demo menu)**

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

For quaternions and octonions, you have the option of selecting only quaternion, hypernion (hypercomplex), or cquat types or a mixture of quaternion, hypernion and cquat types.

There are radio boxes that allow you to customize how random variables are processed to create new 3-D 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 quad formula

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

QuaSZ 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 [QFRM]

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

## 8.6 Save Frames [QFRM]

This command saves the current frame buffer in a [qfrm] 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 QuaSZ Mac. |
| Tutorial | An introduction to CQuat fractals |
| Parser Information | Quick reference to QuaSZ Mac's parser variables and functions. |
| Built-in Formulas | Quick reference to QuaSZ Mac's built-in formulas. |
| Bibliography | Sources for fractal information and complex numbers. |

About QuaSZ Mac    Displays the version number and author info for this application.
Chronology        History of the programs preceding QuaSZ Mac

## 9.1    Getting Started

**Getting Started**

Welcome to QuaSZ Mac!
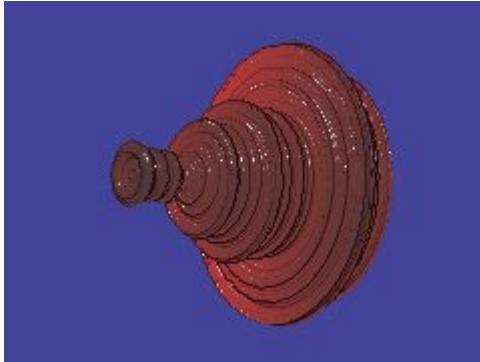


This is a short tutorial that will cover basic commands and background material necessary for a user to create a model with QuaSZ Mac.

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 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 quaternion that uses that zspace as its constants will be drawn.  You can continue to click on areas of the draw window and see what quaternion lies there, or you can press the space bar to open a new window and draw the quaternion 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 quad figure, you can make a 3D object from it using any of the Export commands in the File menu, such as Save Quaternion to [obj].

Other 3-D Types in QuaSZ Mac include hypernions, cubic Mandelbrots and octonions.  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.  Cubic Mandelbrots are more complicated to set up.  There are ten built-in formulas 100-109 that work with the cubic Mandelbrot type (and ten for octonions, 110-119.)  Unless you have a background in cubic or octonion math, it's best to use the Demo/Random commands to start with.  These automatically use a procedure like Scan to scan through cubic Mandelbrot/octonion space and pick out interesting areas to display.  For a good preview of what QuaSZ Mac is capable of, be sure to experiment with all of the Demo/Random commands.

QuaSZ Mac allows you to Undo the last command in most cases.

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

## 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.)
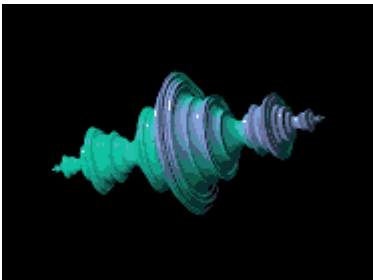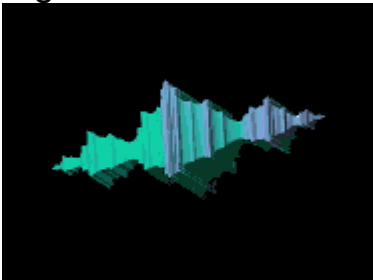


Figure 1.  Quaternion Julia set of -1+0i



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, $q^2+c$:

$x \rightarrow x*x - y*y + z*z + w*w + cx$
$y \rightarrow 2.0*x*y + 2.0*w*z + cy$
$z \rightarrow 2.0*x*z - 2.0*w*y + cz$
$w \rightarrow 2.0*x*w - 2.0*y*z + cw$

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.
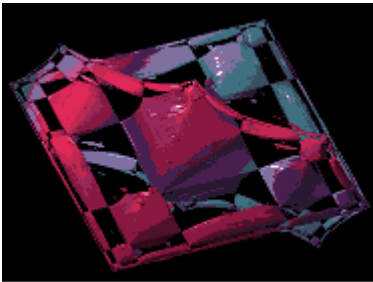


Figure 3.  Equilibrium -- cquat Julia rendering of -1+0i

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 Raytracer* [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 QuaSZ Mac 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 <u>Fun #1 or Fun #2</u> popup controls)

0; $z^2+c$ (Note 2)
1; $cz(1-z)$
2; $z(z-1/z)+c$
3; $cz^2-c$
4; $z^2+cz^2+c$
5; $z^3+c$
6; $((z^2)*(2z+c))^2+c$
7; $z^2+j+kzn$
8; $(x^2-y^2-j,2xy-k)$ when x>0; else $(x^2-y^2-c+jx,2xy+kx-k)$  -- Barnsley (Note 1)
9; $z^2-cz^3+c$
10; $z^3+conj(z)c+c$
11; $z^z+z^3+c$
12; $z^3-z+c$
13; $z^2+exp(z)+c$
14; $(z^2-c)(z+1)$
15; $cz^3+c$
16; $z^2+cexp(z)+c$
17; $sin(z)+cz^2+c$
18; $(z-1)/c$ when x>=0; else $(z+1)/cc$ -- Barnsley
19; $(z-1)/c$ when kx-jy>=0; else $(z+1)/c$ -- Barnsley
20; $(z+j)(z+k)(z^2+1)+c$
21; $(z+j)(z^2+z+k)+c$
22; $(z-1)(z^2+z+c)$
23; $(z+j)(z+k)(z+1)+c$
24; chaos formula -- Barnsley
25; snowflake IFS -- Barnsley
26; $cos(z)+c$
27; $z^3+exp(z)+c$
28; $(z-c)(z+1)(z-1)+c$

29; z^4-c^4
30; sin(z)-c
31; z^4+exp(z)+c
32; (c(z^2+1)^2)/(z^2-1)
33; z^2+tan(z)+c
34; strange attractor IFS (s=1.4142)  -- Barnsley
35; z^5-c^4
36; composite function cz-c/z && z^2+c
37; 1/z^2+c
38; (z^3+3(c-1)z+(c-1)(c-2)
39; z(z^5+c^2)
40; z(z^6+c^2)
41; z^7-z^5+z^3-z+c
42; z^4-z^2+c
43; z^3+tanz+c
44; z^2+z^c+c
45; z^3+c/z+c
46; discretizes Volterra-Lotka equations via modified Heun algorithm
47; z^3+c^z+c
48; Quaternion set -- q^2+c^2
49; z^2+cz^2+c
50; spiral network -- C. Pickover
51; z+z^3/c+c
52; tan(z)-(z^2+c)
53; z^2+arcsinz-c
54; cos(z)+csc(z)+c
55;  cos(z^3)+c
56; z^7+c
57; sin(z)+c^3
58; z^3+zn+c
59; Quaternion set -- q^3+c^3
60; 1/z^2+exp(z)-c
61; 1/z^3+logz-c
62;  z^3+j+kzn
63; cz^2+zn+c
64; sin(z)+kzn+j
65; vers(z)+c
66; vers(z)+covers(z)+c
67;  c*vers(z)+c
68; vers(z)*covers(z)+c
69;  (foggy coastline #1)^2+c -- Barnsley
70; (foggy coastline #2)^2+c -- Barnsley
71; sin(vers)+c
72; csec(z^2)+c
73; z^3+sech(z^2)+c
74; c*z^(c-1)+z^2
75; cos(-1/w^2)+c

76; (z/e)^z*sqr(2*pi*z)+c
77; 1/8(63z^5-70z^3+15z)+c
78; (z^2+e^(-z))/(z+1)+c
79; z^2*exp(z)+c
80; (z^3)/c+c
81; z^3*sqr(2*pi/z)+c
82; z^2-csc(h)cot(h)+c
83; (1/(sin(z)+c))^3
84; z^2-c; where x=abs(real(z)), y=imag(z) -- Paul Carlson
85; z^2; where x=abs(real(z))-cr, y=imag(z)-ci -- Paul Carlson
86; c*cos(z)+c
87; z*cos(z)+c
88; z*2+z/sin(z)+c
89; z^2+z^pi+c
90; z^2+z^3+sin(c)+cos(c)+c
91; z^2*(1-cz^2)+c
92; 1/(z*z/c)+c
93; 1/(z*z)-z+c
94; (1+c#)/(z*z)-z
95; (1+c)/(z*z/c)+c
96; (1/c)/(z*z/c)+c
97; 1/((z*z)/c)+(c/(1/z-c))
98; 1/(z*z*z/c)+c
99; 1/(z*z*z)-z+c
100; z^3-3c^2z+s -- cubic Mandelbrot (Note 3)
101; z^3-3sz+c -- alternate cubic Mandelbrot
102;  z^3-3c^2z^2+s -- cubic Mandelbrot variant
103; z^3-3sz^2+c -- alternate cubic Mandelbrot variant
104; z^3-3c^2/z+s -- cubic Mandelbrot variant
105; z^3-3s/z+c -- alternate cubic Mandelbrot variant
106; z^3-3c^3*z+s -- cubic Mandelbrot variant
107; z^3-3s/z^2+c -- alternate cubic Mandelbrot variant
108; z^3-3c^2+z+s -- cubic Mandelbrot variant
109; z^3-3s+z+c -- alternate cubic Mandelbrot variant
110; (O*C^-1)(C*O)+c  -- octonion set (Note 4)
111; cO*CC(1-C*O) -- octonion set
112; O*C(O-CC*O)+c -- octonion set
113; cO^2+O^2*CC-c -- octonion set
114; O^2*CC+O^2*C+c -- octonion set
115; O^3*CC+c -- octonion set
116; O^3*CC+O^3*C+c -- octonion set
117; O^4*CC+c -- octonion set
118; cO^3*CC+c -- octonion set
119; O^2*CC+O^3*C+c -- octonion set
120; 2*z*c#*cos(pi/z) -- Godwin Vickers
121; 2*z*c#*sin(pi/z) -- Godwin Vickers
122; 2*z*c#*tan(pi/z) -- Godwin Vickers

123; 1/z^3+sin(z)*c^2
124; cosh(z)/c*z+c^2
125; exp(z)/z^3-c
126; cosh(z)*z^2-c^2
127; 1/z^2-cz-c
128; tan(z)-czc
129; (tan(z)-1/z^3)/c^2
130; cosh(z)*cos(z)+1/c
131; z^4/(z^3-c^2)
132;1/z^2-tan(z)+c
133; tan(z)/z^3+c
134; 1/z^3-cz+1/c
135; z^3+tan(z)*c
136; 1/z^3-tan(z)-c
137; tan(z)-sin(z)+c
138; 1/z^2-tan(z)+1/c
139; z^4+sin(z)/c
140; Mandelbrot set(sine variation)
141; z^1.5+c -- Godwin Vickers
142; (z^2-z^(2-s))/s+c -- Escher set by Roger Bagula
143; z^2+z/(|z|+c) -- Roger Bagula
144; quantum set -- S.M. Ulam
145; prey predator #1 -- Roger Bagula
146; prey predator #2 -- Roger Bagula
147; Klein group #1 -- Roger Bagula
148; Klein group #2 -- Roger Bagula
149; Klein group #3 -- Roger Bagula
150;  Loxodromic by Thomas Kromer (fixed type)
151; squared loxodrome
152; Gedatou by Thomas Kroner (fixed type)
153; Ventri by Thomas Kroner (fixed type)
154; squared gedatou
155;  fn(z)-cfn(z)  (Note 5)
156;  fn(z)+fn(z)+c"
157; cfn(z)+c"
158;  fn(z)+cfn(z)+1"
159; fn(z)+c
160; fn(fn(z))+c
161; fn(z)+fn(c)
162; fn(z)+zn+c
163; cfn(z)+zn
164; fn(z)+kzn+j -- generalized phoenix curve
165; fn(z)*fn(z)+c
166; fn(1/(fn(z)+c))
167; (1/fn(z))^2+c
168; (1/fn(z))^3+c
169; fn(z)/(1-fn(z))+c

170;  Sinus by Thomas Kromer (fixed type)
171;  Sinus #2 by Thomas Kromer (fixed type)  (Note 6)
172;  Rings of Fire by Thomas Kromer (fixed type)  (Note 6)
173;  Teres by Thomas Kromer (fixed type)
174; z^2+y+c
175; z^2+y[n+1]+c
176; z^2+zi+c
177; z^2+zi[n+1]+c
178; zr^2+3zi+c
179; zr^2+4zi+c

Note 1: For further info on Michael Barnsley's formulas, see his "Fractals Everywhere".

Note 2:  The quaternion, hypernion and cquat types use the complex constant gadgets to input cr, ci, cj and ck.  These may be zero when generating a Mandelbrot-like set of these functions.  Quaternion sets may then be mapped by grabbing points (using Image/Scan) from interesting areas near this set.  Cj and ck must be entered manually for Julia sets.

Note 3:  For the traditional cubic Mandelbrot (example in The Science of Fractal Images), cj and ck (hypercomplex components of z and c) should be set to zero.

Note 4: Octonions have a form of xr+xi+xj+xk+xE+xI+xJ+xK.  For these formulas C is the octonion constant $(1,1,1,1,1,1,1,1)$ and CC is the octonion conjugate $(1,-1,-1,-1,-1,-1,-1,-1)$.  Additional options are entered via the Arg box in the Quaternion editor window.  To rotate the extra four-octonion dimensions (E-K) use the following syntax:

| | | |
|---|---|---|
| 0I | -- | rotate OE-OK to OI-OE |
| 0J | -- | rotate OE-OK to OJ-OI |
| 0K | -- | rotate OE-OK to OK-OJ |

To normalize the C and CC constants:

| | | |
|---|---|---|
| n | -- | normalize C and CC (default is un-normalized) |

Alternate octonion initialization:

| | | |
|---|---|---|
| c | -- | set OE-OK to .01 at beginning of each iteration |

With octonions, you have your choice of two different algebraic systems (depending on whether the Type is set to Quaternion or Hypernion.) Hyper-octonions use alternate definitions of the basic octonion multiplication tables.  This is similar to the difference between hypercomplex quaternions (hypernions) and quaternions.  The algebra for octonion and hyper-octonions differ in how they conform to (or fail in) the associative and commutative laws.

Note 5: the term 'fn(w)' represents any one of 54 user-defined functions chosen through the fn1-fn4 popup controls:

| | | | |
|---|---|---|---|
| 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)          47: loxodromic(w)     48: gedatou(w)     49: ventri(w)

50 sinus #1(w)          51 sinus #2(w)          52 rof(w)          53 teres(w)

When only fun#1 or fun#2 is used and a single user-defined function is involved, the function is taken from fn1.  When two user-defined functions appear in a function, the fn2 gadget supplies the second function type, except as noted below.  For composite plots that use both fun#1 and fun#2, fun#1 takes its functions from fn1 and fn2 and fun#2 takes its functions from fn3 and fn4.

Note 6:  For the loxodromic functions, Sinus #2 and Rings of Fire, the Arg Limit variable (in the Edit Function/Type window) is used as an additional ingredient.  Try values -1.5 to 1.5 for Sinus #2 and 1.5 or PI/2 for Rings of Fire.

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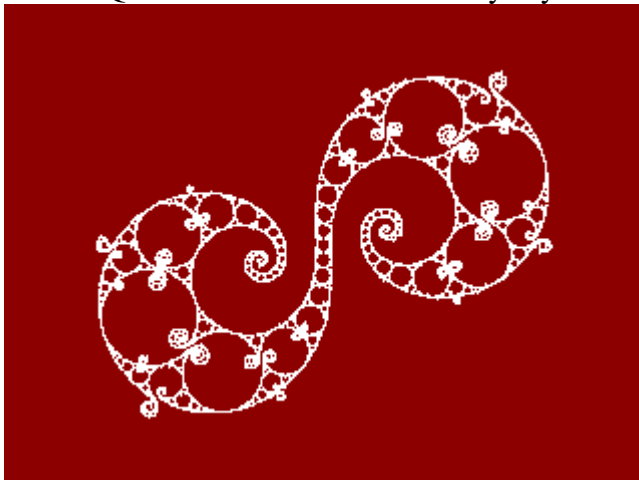
Wegner, Tim and Tyler, Bert: "Fractal Creations", Second Edition, Waite Group Press, Corte Madera, CA, 1993.

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

## 9.6    About QuaSZ Mac

**About QuaSZ Mac**

>>>>> QuaSZ Mac™  v1.0 © 2005 by Mystic Fractal <<<<<



QuaSZ Mac exports 3-D slices of formulas based on 4-D and 8-D complex number planes. QuaSZ Mac currently supports quaternion, hypernion, cubic Mandelbrot and octonion renderings of the Mandelbrot set and Julia sets.  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 B. Mandelbrot) and Julia sets (G. Julia.)  Also, there are mapping options that use non-traditional methods, such as composite formulas and IFS (Michael Barnsley).

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

QuaSZ Mac requires a true-color video adapter for best results.

Acknowledgements:  A delight to include Thomas Kromer's loxodromic algorithms in QuaSZ Mac.  Special thanks to Frode Gill for his quaternion and ray-tracing algorithms, to Dirk Meyer for his Phong-shading algorithm, to Onar Aam for tutoring me on octonion algebra and its esoteric details.  An honor to mention Stig Pettersson, for his ground-breaking work in cubic Mandelbrots, giving me clues to an historic fractal world I hardly knew existed...

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

## 9.7    Chronology

**Chronology**

History of the programs preceding QuaSZ Mac:

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

## - C -

## - D -

## - E -

## - F -

## - H -

## - I -

## - P -

## - R -

## - T -