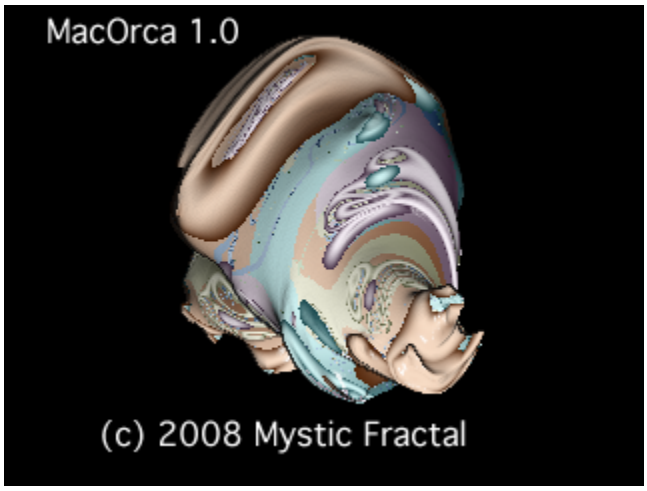
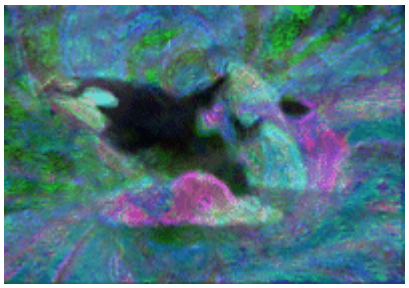


MacOrca Application Help

Table of Contents

1 Main Index

MacOrca Help Index



Getting Started

An Introduction To CQuat Fractals by Terry W. Gintz

Tutorial On Juliat 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 [MOT]	Load a data file in text (platform independent) format.
Save Text Format [MOT]	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 MacOrca. The image and data file for the opening picture (title.png and title.orca) 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 MacOrca stores the file that you want to open.

Directories
Select the directory in which MacOrca 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, MacOrca 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. MacOrca 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. MacOrca adds the extension .orca.

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. Quat-traps are not supported with this command.

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. Quat-traps are not supported with this command.

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. Quat-traps are not supported with this command.

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. Quat-traps are not supported with this command.

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 [ORT]

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

2.14 Save Text Format [ORT]

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 Formula Parameters

Formula Parameters

Function is the combo control for selecting the formula used to generate the fractal. There are additionally two fractal Type controls, an Arg control, and an 'S' control that are used in various fractal formulas and user-defined functions. User-defined functions are supplied through list boxes fn1 and fn2. Although there are only 20 built-in formulas in MacOrca, all of the formulas are generalized and can use up to two user-defined functions (out of a total of 52) and two fractal types, from a list of four types, quaternion, hypernion, compquat and juliat. These Types specify the type of quad math used in calculating the formula. The types are switched at various stages of the formula calculations, so that in effect you can merge the two fractal types into a totally different fractal type, a hybrid type, if you want. Note: some user-defined functions are only defined as quaternion or hypercomplex (hypernion) math. When the fractal type is neither of these, the math defaults to hypercomplex. This is true for all transcendental, logarithmic and power functions, except w^(int)s.

The Bail box is used to enter an optional bailout string, such as "|z|<4" or "z=sin(z), x<4" (without the quotes). This supersedes any bailout defined by the Mapping menu and Bailout variable in the Fractal Parameters window. A list of functions supported by the Bail box is listed in Parser Information.

Click on the Okay button to use the formulas currently displayed in the window, or Cancel to exit the window without making any changes. Click on Draw to apply a new formula, etc. without closing the Formula window.

The Reset button returns all boxes and slider values to their original values when the window was opened.

3.3 Fractal Parameters

Fractal Parameters Window

This is the data-collection window for the 3-D and 2-D generators.

Quad constants are cr, ci, cj and ck.

Three rotate variables determine the 3-D angle of rotation.

The Bailout variable can range from .000001 to 65000. For most escape-time formulas that have an attractive point at infinity, the bailout is set to a value 4 or greater. A larger value tends to make the 3-D figure smoother and fuller with some loss of detail.

Steps and Fine Tune are pitch adjustments that bear on the quality of the 3-D plot at the expense of lengthier calculations.

The Slice variables may be altered to display different planes of the quad set.

The scaling variable is used with 2-D plots to increase or decrease color useage, and with bubble and orbit-trap methods when used as 2-D fractals or 3-D surfaces. Selecting the Carlson option (in the Orbit-Traps window) or clicking on the Bubble Extensions button (in the coloring options window) automatically adjusts this variable to a suitable value. Hint: increase its magnitude to make the bubbles larger, but keep the value negative.

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 MacOrca.

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 MacOrca 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 MacOrca, 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 (either 2-D or 3-D) 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.

Once you find an interesting Julia 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.

Note: repeated zoom is not available for Cloud fractals, but when using this command you can "magnify" *once* a portion of the draw window, instead of altering the x/y ranges.

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 (See below for exception.) 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

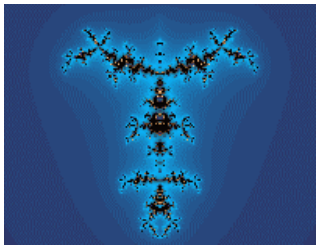
Reset Ranges (Image menu)

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.

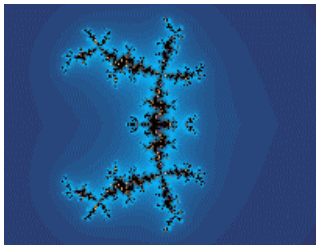
4.6 Symmetry

Symmetry (Image menu)

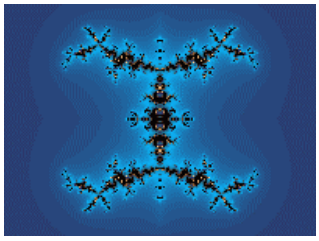
This produces a mirror image from left to right (vertical) or top to bottom (horizontal) or both (xy). You can zoom with symmetry, but the results will be uncertain if the zoom box is off-center on the window.



Vertical symmetry



Horizontal symmetry



XY symmetry

4.7 Mapping

Mapping

Mapping controls how the bailout is processed for each pixel.

Z-Real-Z-Imag Mapping based on difference of real and imaginary parts of Z.
Z-Real*Z-Real/Z-Imag*Z-Imag Mapping based on real part of Z squared divided by imaginary part of Z squared.
Z-Imag*Z-Imag/Z-Real*Z-Real Mapping based on imaginary part of Z squared divided by real part of Z squared.
(Z-Real*Z-Real+Z-Imag*Z-Imag)/Z-Real*Z-Real Mapping based on absolute value of of Z divided by real part of Z squared.

(Z-Real*Z-Real-Z-Imag*Z-Imag)/Z-Imag*Z-Imag Mapping based on the difference of Z parts squared divided by the imaginary part of Z squared.
Abs(Z-Real)*Abs(Z-Imag) Mapping based on product of 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.

Mapping supplies the bailout criteria for the iterative loops, both 3-D and 2-D fractal types.

4.7.1 Z-Real-Z-Imag

Z-Real-Z-Imag
Mapping based on difference of real and imaginary parts of Z.

4.7.2 Z-Real*Z-Real/Z-Imag*Z-Imag

Z-Real*Z-Real/Z-Imag*Z-Imag
Mapping based on real part of Z squared divided by imaginary part of Z squared.

4.7.3 Z-Imag*Z-Imag/Z-Real*Z-Real

Z-Imag*Z-Imag/Z-Real*Z-Real
Mapping based on imaginary part of Z squared divided by real part of Z squared.

4.7.4 (Z-Real*Z-Real+Z-Imag*Z-Imag)/Z-Real*Z-Real

(Z-Real*Z-Real+Z-Imag*Z-Imag)/Z-Real*Z-Real
Mapping based on absolute value of of Z divided by real part of Z squared.

4.7.5 (Z-Real*Z-Real-Z-Imag*Z-Imag)/Z-Imag*Z-Imag

(Z-Real*Z-Real-Z-Imag*Z-Imag)/Z-Imag*Z-Imag
Mapping based on the difference of Z parts squared divided by the imaginary part of Z squared. Changes the way banding appears in complex mappings.

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

Abs(Z-Real) * Abs(Z-Imag)
Mapping based on product of absolute value of parts of 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 Orca 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:

2-D Map	Set fractal type to two-dimensional mapping.
3-D Map	Set fractal type to three-dimensional mapping.
Mandelbrot0	Mandelbrot set (orbit starts at zero.)
MandelbrotP	Mandelbrot set (orbit starts at zpixel.)
Julia	Julia set.

5.1 2-D Map

The fractal type is set to 2-D. Use this command to create standard 2-D fractals. Bailout, max iterations and complex 'c' are set in the Fractal Parameters window.

5.2 3-D Map

The fractal type is set to 3-D. Use this command to create true 3-D fractals. The figure dynamics vary with the type of quad math used. The math type is set in the Formula Parameters window.

5.3 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.

5.4 MandelbrotP

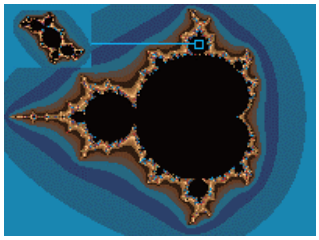
MandelbrotP (Type menu)

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.

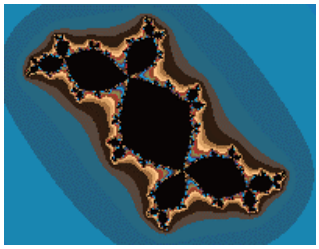
5.5 Julia

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

6 Render Menu

Render menu commands

The Render menu offers the following commands:

Coloring Options	Palette-based coloring options.
Noise	Edit noise factors.
Texture Scale	Set scaling factor for texture.

6.1 Coloring Options

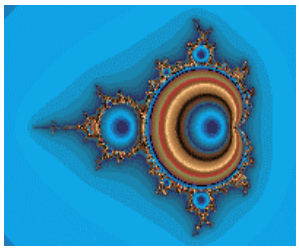
Coloring Options

The following options are described as 2-D coloring options, but the Level Curve, Biomorphic and Decomposition options are also used for 3-D renderings.

Inside Coloring Options

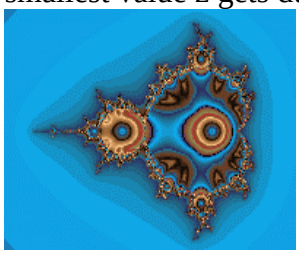
Level Curves
Level-curves map the set points based on the value of Z. This allows the inside of the complex set to be color-scaled.

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

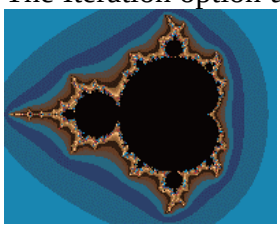
Atan Coloring
Uses an algorithm by David Makin to color an image. The angle of each point (as it escapes the set border) is used to color the image.

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 an image. The smallest magnitude of z (found while calculating the set border) is used to render the image.

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.

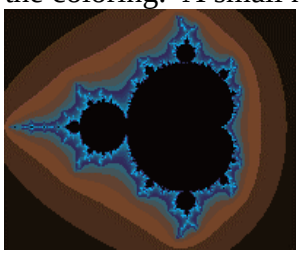
Outside Coloring Options

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



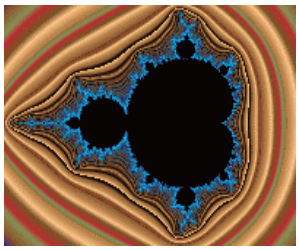
Escape-time coloring.

Log Palette
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 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.

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.

The Divide by (1-32) 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. 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.

6.1.1 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.

6.1.2 Small Log

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

6.1.3 Indexed Log

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

6.1.4 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.

6.1.5 Indexed Linear

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

6.1.6 Bubble

Bubble uses Paul Carlson's contour-mapping method to produce 3D-like bubble pictures. The method is very sensitive to which formula is used, working best with the basic Mandelbrot set z^2+c and the like. The Scaling variable (Fractal Parameters window) needs to be a small negative value, usually less than one. A larger than necessary Scaling value will cause the colors to overlap in the bubbles. For split palette pictures, the colors are divided according to their level index, as in Indexed Linear. Note: the magnitude of the Scaling variable needs to be increased or decreased proportionately by the number of palette splits to keep the "bubbles" the same size. The color ranges in the palette should be graded from light to dark to highlight the bubble centers.

6.1.7 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.

6.1.8 Iteration

Iteration option uses the point's escape time to color points outside the Mandelbrot or Julia set.

6.1.9 Log Palette

Log Palette -- 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..

6.1.10 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 works best here.

6.1.11 Angle

Angle uses the absolute value of a point's exit angle (theta.) This is the atan method in Fractint.

6.1.12 Angle-Iteration

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.

6.1.13 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 is best for Continuous Potential.

6.1.14 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.

6.1.15 Decomposition

Decomposition

When a Decomposition option is set, you have the option of performing either an external or internal binary decomposition. For Mandelbrot/Julia curves, z-arg is broken into two parts.

(Consult The Beauty of Fractals by Peitgen & Richter for a mathematical explanation of decomposition.) When Biomorph or Epsilon is decomposed, the tendrils or hairs are decomposed as external points. Use the Set Only flag to emphasize the tendrils and hairs when external decomposition is used.)

6.1.16 Biomorphic

Biomorphic

Biomorphs test the real Z and imaginary Z values after exiting 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.)

6.1.17 Set Only

Set Only plots all outside points in the background color.

6.2 Orbit Trap and Other Rendering Options

Orbit Trap and Other Rendering Options

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

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 (0-99) from orbit trapping.

Click on Okay to close the window and apply changes, if any. Click on Close to exit the window without applying changes.
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.

Phoenix

The Phoenix option rotates the planes, so that the imaginary plane is mapped horizontally and the real plane is mapped vertically. Select the Phoenix box to enable this option.

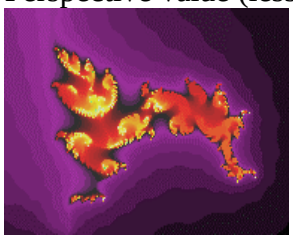
Torus

Pixels are mapped around a torus, and then expanded to fit the drawing area. Select the Torus box to enable this option. A generalized form of Earl Hinrichs' torus method, variables are provided for center x and center y to define the c and z radii and may both equal 0.0. Results will vary with the formula used, but resembles the warping effect found in hypercomplex

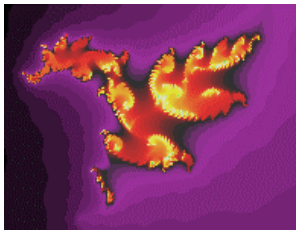
images. Two versions of this method are provided: the Pixel method which uses pixel values to map the torus to the fractal space, and the Two-Pi method which uses an initial rectangle 2 pi by 2 pi to map the torus to a fractal image. With the Two-Pi method, when you zoom the rectangle's size and starting points are changed to match the zooming area. The rectangle's coordinates are saved with the fractal. If you turn off the torus flag after zooming and then reinitialize the torus flag, the rectangle reverts to a 2X2 area, so the image will change accordingly. Rotating is not supported for the Two-Pi method, but does work in a limited way with the Pixel method.

Invert

The Invert option inverts the plane around a circle. Select the Invert box to enable this option. Select Auto Coords to let Orca calculate the center coordinates and circle radius. Using Auto Coords, the new radius and center coordinates are calculated when the picture is next drawn. You can zoom on an inverted picture as long as radius and center coordinates remain the same. Use the Perspective box to alter the X/Y symmetry of the inversion. A smaller Perspective value (less than 1.0) stretches the inversion in the vertical direction.



Original picture



Inverted

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.

7 Demo Menu

Demo menu commands

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

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

7.1 Random 2-D Julia

Random 2-D Julia (Demo menu)

A random Julia fractal is generated. Many of the built-in options of Orca are selected on a random basis (if enabled in the Random Setup window), and the Mandelbrot space for one of

the twenty-two built-in formulas is scanned for an interesting Julia set. The palette used is also randomized (if Random coloring is selected.). Note: In most cases the Julia search is a short one, but sometimes the "seek" mode can seem to get stuck when the criteria for an interesting Julia set fails to match the formula used. In the latter case, click the left mouse button and restart the search process. 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 Orca that the user may be unfamiliar with: no knowledge of fractal science/math required!

7.2 Random 3-D Julia command

Random 3-D Julia (Demo menu)

A random 3-D Julia fractal is generated. A set of formulas appropriate for 3-D fractals is scanned to find an interesting Julia set, and then the parameters are adjusted to produce a 3-D image. The ranges are reset (if z-space is unselected in the Random Setup window), Z is set to 2.0, and the lighting is set for optimum viewing (if lighting is checked in the Random Batch window.)

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 Render

Random Render (Demo menu)

The rendering options for the current fractal are randomized according to the options enabled in the Random Setup window.. Does not affect formula or range variables.

7.6 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
Lighting -- (default off) check to set default lighting
Symmetry -- (default off) check to randomize symmetry used
Rotation -- (default on) check to randomize camera angles
Coloring -- (default off) check to reset coloring parameters
Iteration -- (default off) check to randomize iterations
Z-Space -- (default on) check to set default z-space
Constants -- (default on) check to randomize the complex constants cj and ck
Slices -- (default off) check to randomize slice in Initial Values window
Stalks -- (default off) check to randomize orbit-trap
Level -- (default off) check to randomize level curve
Filter -- (default off) check to randomize filter
Fn1/Fn2 -- (default off) check to randomize user-defined functions
Types -- (default on) check to randomize quad math used
Mapping (default off) check to randomize bailout mapping

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 [OFRM]	Load frame buffer.
Save Frames [OFRM]	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

MacOrca 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 [OFRM]

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

8.6 Save Frames [OFRM]

This command saves the current frame buffer in a [ofrm] 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

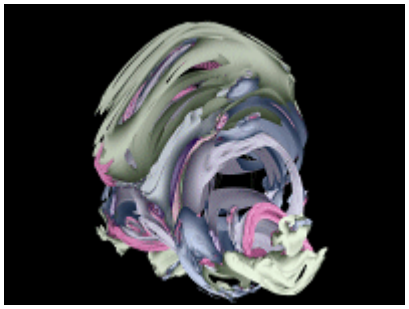
Help topics

Getting Started	Tutorial for new users of MacOrca.
Introduction To CQuat Fractals	CQuat tutorial
Tutorial On Juliat Fractals	Juliat Fractal Introduction
Parser Information	Quick reference to MacOrca's parser variables and functions.
Built-in Formulas	Quick reference to MacOrca's built-in formulas.
Bibliography	Sources for fractal information and complex numbers.
About MacOrca	Displays the version number and author info for this application.
Chronology	History of the programs preceding MacOrca

9.1 Getting Started

Getting Started

Welcome to MacOrca!



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

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 2-D Julia figure, you can change it into a 3-D figure just by changing the Type to 3-D map. This figure can be converted to a 3-D 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.

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

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

9.2 Introduction to CQuat Fractals

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

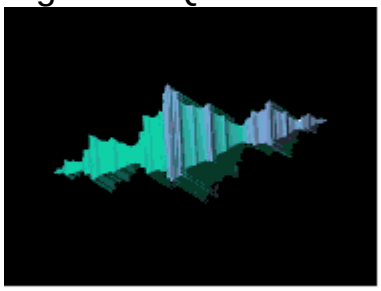


Figure 2. Hypercomplex Julia set of -1+0i

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.

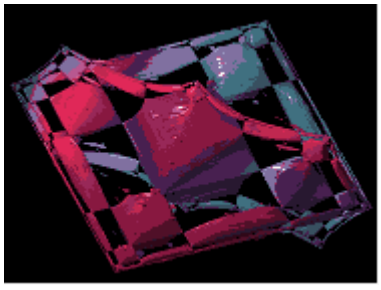


Figure 3. Equilibrium -- cquat Julia rendering of -1+0i

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.

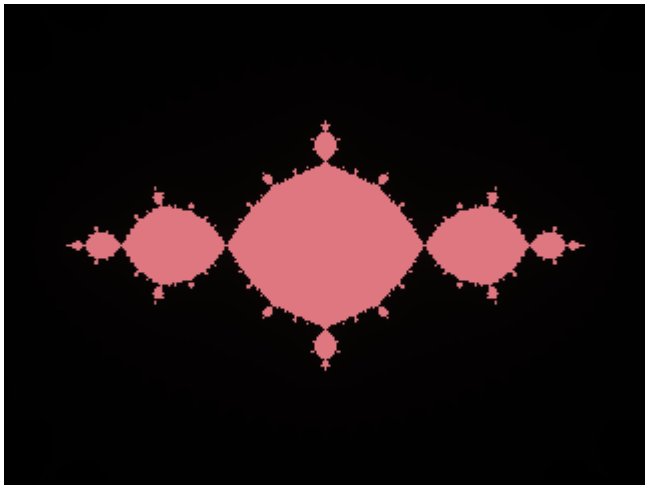
References

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 Tutorial On Juliat Fractals

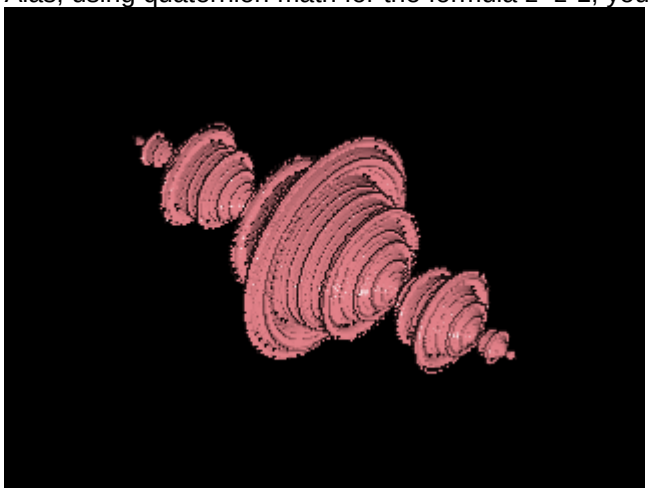
It is easy to speculate on what a true 3D representation of a 2D Julia set based on the complex constant -1+0i would look like, but thus far no quad math has produced the figure you might expect.

The 2-D version of this set is:



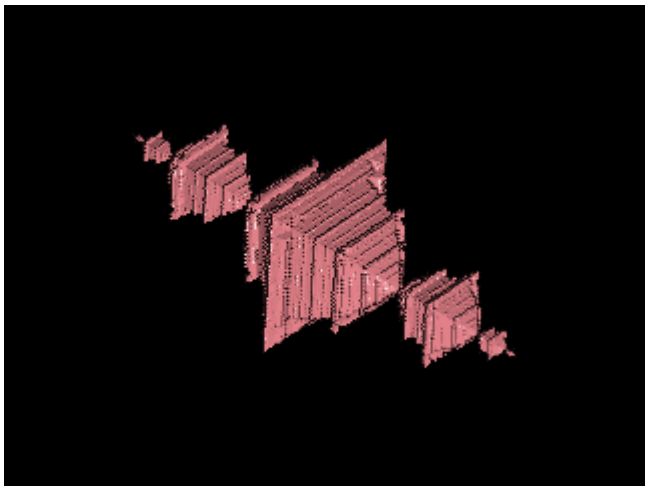
So given all the knobbies in this fractal, you would expect to see the same in a 3-D version.

Alas, using quaternion math for the formula z=z-1, you get:



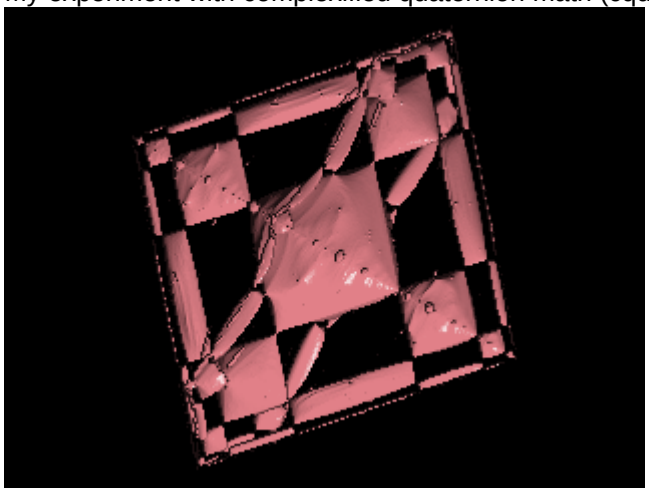
which blurs all the knobbies like a lathe-like turning.

Using hypercomplex math (hypernion type fractal), you get:



which again blurs all the knobbies, this time in a squred-off way.

My experiment with complexified quaternion math (cquat) produces:

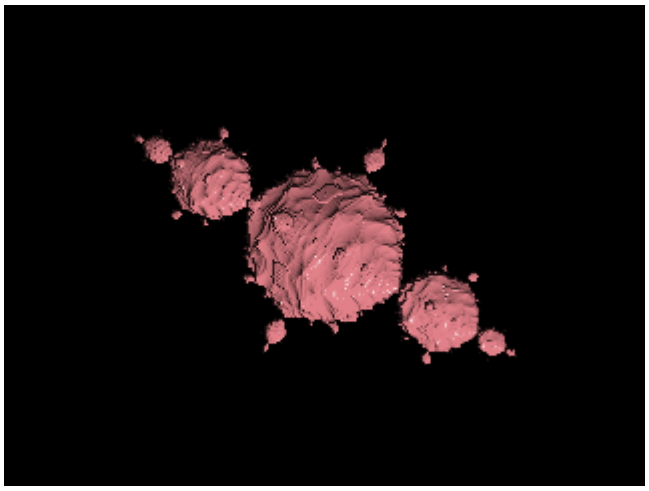


which interestingly enough retains some of the knobs and adds a lot of links that aren't in the original 2-D fractal.

Fortunately, there are other quad maths that can be explored for their fractal producing characteristics.

Using the random quad generator in Hydra, I discovered a quad matrix that seems to be the closest approximation to what is needed to generate true 3-D Julia sets.

Since the matrix was found visually instead of logically, the image is shown first:



The squaring matrix for this fractal looks like:

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

and the derived iterative formula (for z^2+c) is:

*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*

Considering that in the 2-D world the iterative formula for z^2+c is:
*x -> x*x - y*y + cx*
*y -> 2.0*x*y + cy*

it seems that this quad math (which I have named Juliat) hits the target right on the button!

How strange that no algebraic rules have been established for this matrix, nor is it discussed as a possible math for rendering 3-D fractals.

Note: these figures were all drawn using forty iterations, which gives a rather coarse 3-D surface, but is necessary to fully reveal the shape of the Juliat fractal. With 2-D fractals, the formulas are typically iterated 150 times or more. At fewer than 20 iterations, Juliat fractals appear to have dabs of paint spread across the figure.

9.4 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 MacOrca 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)

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.

9.5 Built-in Formulas

Built-in Formulas (select the following prefix in the Function popup controls)

0 -- fn1(z)*fn2(z)+c (Note 1)
1 -- fn1(z)-cfn2(z)

2 -- fn1(z)+fn2(z)+c
3 -- cfn1(z)*fn2(z)+c
4 -- fn1(z)+cfn1(z)+1"
5 -- fn1(fn2(z))+c"
6 -- fn1(z)+fn2(c) "
7 -- fn1(z)+fn2(zn)+c"
8 -- cfn1(z)+fn2(zn)"
9 -- fn1(z)+k*fn2(zn)+j"
10 -- fn1(z)/fn2(z)+c"
11 -- fn1(z)*(1/(fn2(z)+c))"
12 -- (fn1(z)/fn2(z))^2+c"
13 -- (fn1(z)/fn2(z))^3+c"
14 -- fn1(z)/(1-fn2(z))+c"
15 -- fn1(z)*fn2(z)-c (var)"
16 -- fn1(z)/(fn2(z)+c)"
17 -- cfn1(z)*(1-fn2(z))"
18 -- if(abs(z)>S fn1(z) else fn2(z))+c" (Note 2)
19 -- fn1(z)*(1-c*fn2(z))+c"
20 -- fn2(fn1(z)+c1)+c2" (Note 3)
21 -- fn1(z)-3*c1^2*fn2(z)+c2"

Note 1: fn1 and fn2 are functions selected from the fn1 and fn2 boxes in the Formula Parameters window:

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: w+c 48: w^2-c
49: w^3-c 50: pi*w 51: e*w

Note 2: S is entered in the S box in the Formula Parameters window.

Note 3: Part of c2 (real) and the rest of c2 (c2i, c2j and c2k) are entered as the Complex

constant in the Fractal Parameters window. C1 is defined as initial z (z-real and z-imag) and is handled independent of any user input.

9.6 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

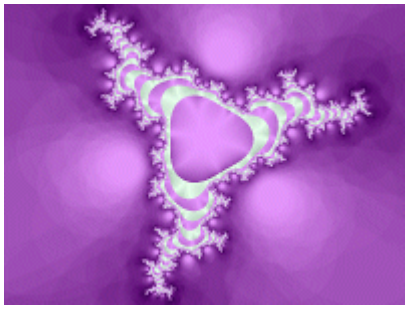
Madera, CA, 1993.

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

9.7 About MacOrbits

About MacOrca

>>>>> MacOrca™ v1.02 © 2008 by Terry W. Gintz



Orca graphs 2-D and 3-D slices of formulas based on 4-D complex number planes. Orca currently supports quaternion, hypernion, cquat and juliat 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 integrated video routines make the program easy for beginners and a powerful complement to advanced fractal artists.

Hypercomplex extensions (as described in Fractal Creations) are standard for all the formulas.

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

Acknowledgements: many thanks to Paul Carlson for his algorithms for 3-D-like fractals. Special thanks to Frode Gill for his quaternion and ray-tracing algorithms, to Dirk Meyer for his Phong-shading algorithm, and to David Makin for sharing his ideas on quaternion colorings and 3-D insights. Thanks also to Francois Guibert for his Perlin noise example.

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

9.8 Chronology

Chronology

History of the programs preceding MacOrca:

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

- 2 -

- C -

- D -

- E -

- F -

- H -

- I -

- M -

- P -

- R -

- T -