

Fractal Imaginator Application Help

© 2004, 2005 ... Mystic Fractal

# Table of Contents

Foreword	0
<b>1 Main Index</b>	<b>8</b>
1 Title Bar .....	8
2 Scroll bars .....	9
3 Size .....	9
4 Move .....	9
5 Minimize Command .....	10
6 Maximize Command .....	10
7 Next Window .....	10
8 Previous Window .....	10
9 Close .....	11
10 Restore .....	11
11 Switch to .....	11
<b>2 Fractal Imaginator Remote</b>	<b>12</b>
1 New .....	12
2 Undo .....	12
3 Size .....	12
4 Color .....	12
5 Batch .....	12
6 Fvr .....	13
7 Draw button .....	13
8 Abort button .....	13
9 Rend .....	13
10 View .....	13
11 Help .....	13
12 Channel Guide .....	14
13 Channel J1 .....	14
14 Channel J2 .....	15
15 Channel ES .....	15
16 Channel SB .....	15
17 Channel Q1 .....	15
18 Channel Q2 .....	16
19 Channel C1 .....	16
20 Channel C2 .....	16
21 Channel CJ .....	16

22	Channel O1 .....	17
23	Channel O2 .....	17
24	Channel LS .....	17
25	Save .....	17
26	Load .....	18
27	Bmp .....	18
28	Png .....	18
29	Jpg .....	18
30	.....	18
31	> .....	19
32	[] .....	19
33	V .....	19
<b>3</b>	<b>File menu</b> .....	<b>20</b>
1	File New command .....	21
2	File Open command .....	21
	File Open dialog box .....	21
3	File Close command .....	22
4	File Save command .....	22
5	File Save As command .....	22
	File Save as dialog box .....	22
6	File Load Parameters command .....	23
7	Load Par File .....	23
8	File Load Palettes command .....	24
9	File Open [JPG] command .....	24
10	File Open [PNG] command .....	24
11	File Save Parameters command .....	24
12	Save Par File .....	24
13	File Save Palettes command .....	25
14	File Save As [JPG] command .....	25
15	File Save As [PNG] command .....	25
16	Load FraSZle Parameters [FSZ] .....	25
17	File Load Lsystem command .....	26
18	File Load Palette [PQZ] command .....	26
19	File Load Palette [MAP] command .....	26
20	File Save Palette command .....	26
21	File Save Lsystem command .....	26
22	File Save QuaSZ command .....	26
23	File Save Q Polygon [OBJ] command .....	27
24	File Save Q Polygon [STL] command .....	27

25	File Simplify mesh command .....	27
26	File Save Q Polygon [POV] command .....	28
27	File Smooth command .....	28
28	File Save Q Polygon [WRL] command .....	28
29	File Save Q Polygon [DXF] command .....	28
30	File Set Max Vertices command .....	29
31	File Save Lsystem to Obj command .....	29
32	File Save Lsystem to Stl command .....	29
33	File Save Lsystem to POV command .....	29
34	File Save Lsystem to Dxf command .....	29
35	Mesh Setup command .....	30
36	File 1, 2, 3, 4, 5, 6 command .....	30
37	File Exit command .....	30
<b>4</b>	<b>Edit menu</b>	<b>31</b>
1	Edit Undo command .....	31
2	Edit Copy command .....	31
3	Edit Paste command .....	32
4	Edit Copy Data command .....	32
5	Edit Paste Data command .....	32
6	Formula Window .....	32
7	Fractal Variables .....	35
	Parameters Window .....	36
	Quaternion Window .....	37
	Axiom Window .....	37
	Rules Window .....	38
8	Cubic Values Window .....	39
9	Octonion Values Window .....	40
10	Size .....	40
11	Ray-Tracing Window .....	40
12	Edit Palette .....	40
	Reverse button .....	42
	Neg Button .....	42
	Map Button .....	42
	H/R Button .....	42
	Spread Button .....	42
	Copy Button .....	42
	SRG Button .....	42
	SRB Button .....	43
	Okay Button .....	43
	Reset Button .....	43
	Cancel Button .....	43
	Red Slider .....	43
	Red edit box .....	43

Green Slider .....	43
Green edit box.....	43
Blue Slider .....	44
Blue edit box .....	44
Smooth Button .....	44
Scramble .....	44
13 Edit Text command .....	44
14 Preferences .....	44
<b>5 Image menu</b> .....	<b>45</b>
1 Image Draw command .....	45
Image Draw Lsystem command .....	45
2 Plot to file .....	47
3 Image Redraw command .....	47
4 Image Auto Clear command .....	47
5 Image Auto Alert command .....	47
6 Image Auto Remote command .....	47
7 Image Auto Time command .....	48
8 Image Abort command .....	48
9 Continue Draw .....	48
10 Zoom .....	48
11 Image New View on Zoom command .....	49
12 Image Clone .....	49
13 Dive .....	49
14 Full Screen .....	49
15 Pilot .....	49
16 Reset .....	50
17 Calculate Fractal Dimension (2D) .....	50
<b>6 Type menu</b> .....	<b>50</b>
1 Mandelbrot .....	51
2 Julia .....	51
3 Type Quaternion command .....	51
4 Type Hypernion command .....	51
5 Type Cubic command .....	52
6 Type Complexified Quaternion command .....	53
7 Type 2D Quaternion Map command .....	53
8 Type 2D Hypercomplex Map command .....	53
9 Type 2D Cubic Map command .....	53
10 Type 2D Complexified Map command .....	53
11 Type Lsystem command .....	53

<b>7 Break menu</b>	<b>54</b>
1 Biomorph .....	54
2 Biomorph Off .....	54
3 Orbit traps .....	54
Orbit trap values .....	55
<b>8 Render menu</b>	<b>56</b>
1 Level Curve .....	56
2 Decomposition .....	57
3 Decomposition Off .....	58
4 Coloring Filter .....	58
5 Atan Coloring .....	59
6 Bof60 Coloring .....	59
7 Potential Coloring .....	60
8 Texture Scale .....	60
<b>9 Pixel menu</b>	<b>60</b>
1 Phoenix .....	60
2 Symmetry .....	61
3 Solid Guessing .....	61
<b>10 Color menu</b>	<b>61</b>
1 Color Cycle command .....	62
2 Color-Scaling menu .....	62
Escape .....	62
Level .....	63
Continuous Potential .....	63
Use Level Curve .....	64
Set Only .....	64
Background .....	64
Graded Palette .....	64
3 Palette menu .....	65
Palette 1-21 command .....	65
4 Color Divpal1 command .....	65
5 Color Divpal2 command .....	65
6 Color Divpal4 command .....	65
7 Color Divpal8 command .....	65
<b>11 View menu</b>	<b>65</b>
1 View Toolbar command .....	66
toolbar .....	66
2 View Status Bar Command .....	67
status bar .....	67

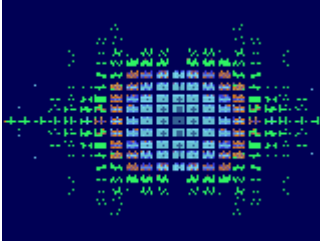
<b>12 Window menu</b>	<b>67</b>
1 Cascade .....	68
2 Tile .....	68
3 Arrange Icons .....	68
4 Size DeskTop .....	68
5 1, 2, ... .....	68
<b>13 A/V menu</b>	<b>68</b>
1 Open Avi Stream .....	69
2 Write Frames .....	69
3 Close Avi Stream .....	69
4 View Avi .....	70
5 AVI Object .....	70
6 AVI WRL .....	70
<b>14 Demo menu</b>	<b>70</b>
1 Random Render .....	71
2 Batch Mode .....	71
3 Random Julia .....	72
4 Random Julia2 .....	72
5 Random Escher .....	73
6 Random Newton .....	73
7 Random Stalks and Bubbles .....	73
8 Random Quaternion .....	73
9 Random Quaternion2 .....	74
10 Random Cubic Mandelbrot .....	74
11 Random Cubic Mandelbrot2 .....	74
12 Random Cubic Julia .....	75
13 Random Octonion .....	75
14 Random Octonion2 .....	75
15 Random Lsystem .....	75
<b>15 Growth menu</b>	<b>76</b>
1 BBM 20 .....	76
2 BBM 5 .....	77
3 BBM 1 .....	77
4 Non-standard J-set .....	77
5 Next Set .....	77
6 Previous Set .....	77
7 Show Orbits .....	77

8 Cores .....	77
9 J-set -- $z^2+c$ .....	78
10 J-set -- $z^{-2}+c$ .....	78
11 J-set -- $c*\sin(z)$ .....	78
12 J-set -- $c*\cos(z)$ .....	78
13 J-set -- $c*\tan(z)$ .....	78
14 J-set -- $c*\exp(z)$ .....	78
15 Basic Growth .....	79
16 Businesss Growth .....	79
17 Global Growth .....	79
18 Starting Business Growth .....	79
19 Business Core Growth .....	79
20 Ending Business Growth .....	79
21 Symmetric Growth .....	79
22 Exp (Same Color Quadrant) .....	80
23 Basic X Growth .....	80
24 Basic Core Growth .....	80
25 Dante Growth .....	80
26 Model 25S .....	80
27 Automatic 8-Fold Symmetry .....	80
28 Automatic 4-Fold Symmetry .....	81
29 Mito Growth .....	81
30 Mito 25 .....	81
<b>16 Help menu</b> .....	<b>81</b>
1 Getting Started .....	82
2 Index .....	83
3 Hot Keys .....	84
4 Parser .....	84
5 Built-in Formulas .....	86
6 Lsystem info .....	99
7 Bibliography .....	103
8 About Fractal Imaginator .....	105
Chronology .....	106
<b>Index</b> .....	<b>109</b>



# 1 Main Index

## Fractal Imaginator Help Index



[Getting Started](#)

[Fractal Imaginator Remote](#)

[Channel Guide](#)

### Commands

[File menu](#)

[Edit menu](#)

[Image menu](#)

[Type menu](#)

[Break menu](#)

[Render menu](#)

[Pixel menu](#)

[Color menu](#)

[View menu](#)

[Window menu](#)

[A/V menu](#)

[Demo menu](#)

[Growth menu](#)

[Help menu](#)

## 1.1 Title Bar

### Title Bar

The title bar is located along the top of a window. It contains the name of the application and drawing.

To move the window, drag the title bar. Note: You can also move dialog boxes by dragging their title bars.

A title bar may contain the following elements:

- Application Control-menu button
- Drawing Control-menu button
- Maximize button

- Minimize button
- Name of the application
- Name of the drawing
- Restore button

## 1.2 Scroll bars

### Scroll bars

Displayed at the right and bottom edges of the drawing window. The scroll boxes inside the scroll bars indicate your vertical and horizontal location in the drawing. You can use the mouse to scroll to other parts of the drawing.

## 1.3 Size

### Size command (System menu)

Use this command to display a four-headed arrow so you can size the active window with the arrow keys.



After the pointer changes to the four-headed arrow:

1. Press one of the DIRECTION keys (left, right, up, or down arrow key) to move the pointer to the border you want to move.
2. Press a DIRECTION key to move the border.
3. Press ENTER when the window is the size you want.

Note: This command is unavailable if you maximize the window.

### Shortcut

Mouse: Drag the size bars at the corners or edges of the window.

## 1.4 Move

### Move command (Control menu)

Use this command to display a four-headed arrow so you can move the active window or dialog box with the arrow keys.



Note: This command is unavailable if you maximize the window.

### Shortcut


Keys: CTRL+F7

## 1.5 Minimize Command

### Minimize command (application Control menu)

Use this command to reduce the Fractal Imaginator window to an icon.

#### Shortcut


Mouse: Click the minimize icon  on the title bar.  
Keys: ALT+F9

## 1.6 Maximize Command

### Maximize command (System menu)

Use this command to enlarge the active window to fill the available space.

#### Shortcut

Mouse: Click the maximize icon  on the title bar; or double-click the title bar.  
Keys: CTRL+F10 enlarges a drawing window.

## 1.7 Next Window

### Next Window command (drawing Control menu)

Use this command to switch to the next open drawing window. Fractal Imaginator determines which window is next according to the order in which you opened the windows.

#### Shortcut

Keys: CTRL+F6

## 1.8 Previous Window

### Previous Window command (drawing Control menu)

Use this command to switch to the previous open drawing window. Fractal Imaginator determines which window is previous according to the order in which you opened the windows.

#### Shortcut

Keys: SHIFT+CTRL+F6

## 1.9 Close

### Close command (Control menus)

Use this command to close the active window or dialog box.

Double-clicking a Control-menu box is the same as choosing the Close command.



### Shortcuts

Keys: CTRL+F4 closes a drawing window  
ALT+F4 closes the application

## 1.10 Restore

### Restore command (Control menu)

Use this command to return the active window to its size and position before you chose the Maximize or Minimize command.

## 1.11 Switch to

### Switch to command (application Control menu)

Use this command to display a list of all open applications. Use this "Task List" to switch to or close an application on the list.

### Shortcut

Keys: CTRL+ESC

### Dialog Box Options

When you choose the Switch To command, you will be presented with a dialog box with the following options:

#### Task List

Select the application you want to switch to or close.

#### Switch To

Makes the selected application active.

#### End Task

Closes the selected application.

#### Cancel

Closes the Task List box.

#### Cascade

Arranges open applications so they overlap and you can see each title bar. This option does not affect applications reduced to icons.

**Tile**

Arranges open applications into windows that do not overlap. This option does not affect applications reduced to icons.

**Arrange Icons**

Arranges the icons of all minimized applications across the bottom of the screen.

## 2 Fractal Imaginator Remote

### Fractal Imaginator Remote

The remote provides access to many of the most-used commands in Fractal Imaginator. Info about each button can be obtained by using the '?' located near the close box in the top right-hand corner.

### 2.1 New

#### New button

Use this button to create a new drawing window in Fractal Imaginator.

### 2.2 Undo

#### Undo button

Use this command to undo the last action. An image can be continued after an undo, if continue was enabled before the last action, but not after an image is resized. Color-cycling is disabled after using Undo.

### 2.3 Size

#### Size button

This allows you to set the drawing area for a picture, independent of the Windows screen size. It also shows which size is currently in use. The aspect for the drawing is based on the ratio of X (horizontal width) to Y (vertical height.) The custom setting allows for any size/aspect that system memory will permit. Videos are limited to the standard 4/3 vga aspect or 1/1. The minimum size for an image is 40X30. Note: if the image is less than 100 width, the aspect must be 4/3 for solid-guessing to work properly.

### 2.4 Color

#### Color button

Use the palette editor to modify the palette(s) in use.

### 2.5 Batch

#### Batch button

Here you set parameters for batching and saving random-generated images to disk.

## 2.6 Fvr

### FVR button

The window opened varies with the fractal type selected, and contains all the major variables that Fractal Imaginator now scales between key frames of an avi stream.

## 2.7 Draw button

### Draw button

Use this button to draw or redraw the image for the current fractal variables. Clicking inside the draw window with the left-mouse button stops all plotting. Use the Cont button to restart plotting from the current column. When the lsystem option is selected, a custom dialog is opened to enter lsystem draw options.

## 2.8 Abort button

### Abort button

Use this command to stop drawing. Clicking inside a window's drawing area or close box (or the program close box) will also stop the drawing. Note: once a plot has started Fractal Imaginator continues to draw the image for that window regardless of which drawing window has the input focus, until done or aborted. You can open and close other drawing windows without affecting the current drawing, but only one drawing is active at any time.

## 2.9 Rend

### Rend button

For quaternions, the current coloring filter and lighting variables are applied. This allows you to see what the surface texture looks like before the fractal is finished drawing. Note: to randomize the coloring filter, select Random Render from the Demo menu.

## 2.10 View

### View button

Displays the entire plot, expanding or shrinking the image to fit in a maximized window without title bar, scroll bars or menu bar. At all other times, part of the picture is hidden by the inclusion of the title bar, toolbar, scroll bars and menu bar. To exit full-screen mode, press any key or click the left-mouse button.

## 2.11 Help

### Help button

Use this button to open the help index for Fractal Imaginator.

## 2.12 Channel Guide

### Channel Guide

The eleven channels accessed via the Fractal Imaginator remote:

J1: Random Julia -- basic Julia sets, using one formula and many different rendering options

J2: Random Julia 2 -- includes more advanced composite-type fractals, using two formulas

ES: Random Escher-like fractals -- Julia fractal set using Escher mapping

S/B: Random Bubbles and Stalks -- based on Paul Carlson's bubble and stalk methods

Q1: Random Quaternion/Hypernion -- traditional 3D quaternion and hypercomplex quaternion fractals

Q2: Random Quaternion/Hypernion 2 -- extended search through non-traditional formulas

C1: Random Cubic Mandelbrot fractals

C2: Random Cubic Mandelbrot2 -- relaxed formulas/parameters

CJ: Random Cubic Julia fractals

O1: Random Octonion fractals

O2: Random Octonion2 fractals -- extended dimensional search

## 2.13 Channel J1

### Random Julia (Channel J1 button)

A random Julia fractal is generated. Many of the rendering options of Fractal Imaginator are selected on a random basis, and the Mandelbrot space for one of the 190+ built-in formulas is scanned for an interesting Julia set. The palette used is also randomized. 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. The algorithm doesn't check the type of split palette that is used, so a mismatch in the "Divide by" color option may need correcting, e.g. Divide-by-four may be selected for a divide-by-eight palette. Feel free to experiment with all the parameters, reframe the image, change palettes etc. This routine provides a fast intro to many options in Fractal Imaginator that the user may be unfamiliar with: no knowledge of fractal science/math required!

## 2.14 Channel J2

### Random Julia (Channel J2 button)

A random Julia fractal is generated. Many of the rendering options of Fractal Imaginator are selected on a random basis, and the Mandelbrot space for one of the 180 built-in formulas is scanned for an interesting Julia set. The palette used is also randomized. 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.

This is like the Random Julia command, except that more options are randomized, including spin and switch, and the Formula Type can be composite or Escher, so the search/draw time may be somewhat longer, and the results not as certain. But the images can be quite weird!

Tip: some things remain to be done after the Julia set is drawn. The algorithm doesn't check the type of split palette that is used, so a mismatch in the "Divide by" color option may need correcting, e.g. Divide-by-four may be selected for a divide-by-eight palette. Feel free to experiment with all the parameters, reframe the image, change palettes etc. This routine provides a fast intro to many options in Fractal Imaginator that the user may be unfamiliar with: no knowledge of fractal science/math required!

## 2.15 Channel ES

### Random Escher (Channel QT button)

A random Julia fractal is generated using Escher-like tilings (Type 10 in the Formula window.) The Rise variable in the Parameters window sets the degree of tiling (1-235). Note: Sometimes it helps to reduce iterations if the tilings do not appear to be distributed throughout the circular plane. 20-25 iterations is sufficient for many Escher plots.

## 2.16 Channel SB

### Random Stalks and Bubbles (Channel S/B button)

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

## 2.17 Channel Q1

### Random Quaternion (Channel Q1 button)

A random quaternion/ hypernion fractal is generated. Like Random Julia, a set of formulas appropriate for quaternions is scanned to find an interesting Julia set, and then the parameters are adjusted to produce a quaternion or hypernion image. The ranges are reset, H<sub>j</sub> is set to 2.0, and the lighting is set for optimum viewing.

Note: for some images an h<sub>j</sub> value of 2.0 may result in a partially clipped image. Sometimes it helps to increase this value to 2.5 or 3.0, but too high a value may interfere with Solid



guessing.

## 2.18 Channel Q2

### Random Quaternion2 (Channel Q2 button)

A random quaternion/hyperfractal is generated. Like Random Julia, a set of formulas appropriate for quaternions is scanned to find an interesting Julia set, and then the parameters are adjusted to produce a quaternion or hyperfractal image. The ranges are reset, H<sub>j</sub> is set to 2.0, and the lighting is set for optimum viewing.

This option uses an extended set of formulas, not all of which may produce useable images all the time. The images can be quite different from the traditional quaternion.

Note: for some images an h<sub>j</sub> value of 2.0 may result in a partially clipped image. Sometimes it helps to increase this value to 2.5 or 3.0, but too high a value may interfere with Solid guessing.

## 2.19 Channel C1

### Random Cubic Mandelbrot (Channel C1 button)

A random cubic Mandelbrot fractal is generated. Like the inverse of Random Julia, the essential cubic parameters are randomly adjusted to point into the four-dimensional formula G<sub>0</sub>, and then the Mandelbrot set is scanned to find an interesting area to zoom into. The ranges are reset, and the lighting is set for optimum viewing.

## 2.20 Channel C2

### Random Cubic Mandelbrot2 (Channel C2 button)

A random cubic Mandelbrot fractal is generated. Like the inverse of Random Julia, the essential cubic parameters are randomly adjusted to point into the four-dimensional formula G<sub>0</sub>, and then the Mandelbrot set is scanned to find an interesting area to zoom into. The ranges are reset, and the lighting is set for optimum viewing.

This option uses the cubic formulas G<sub>0</sub> and G<sub>1</sub>, with relaxed parameters to create cubic-Mandelbrot like fractals that may extend to six or more dimensions.

## 2.21 Channel CJ

### Random Cubic Julia (Channel CJ button)

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 Julia, a set of formulas (G0 and G1) appropriate for cubic Julias is scanned to find an interesting Julia set, and then the parameters are adjusted to produce the cubic image. The ranges are reset, H<sub>j</sub> is set to 2.0, and the lighting is set for optimum viewing. 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.

## 2.22 Channel O1

### Random Octonion (Channel O1 button)

A random octonion Julia fractal is generated. The essential octonion parameters are randomly adjusted to point into an eight-dimensional formula H0-H9, and then the octonion Mandelbrot set is scanned to find an interesting area to zoom into. The ranges are reset, and the lighting is set for optimum viewing (if lighting is checked in the Random Batch window.)

## 2.23 Channel O2

### Random Octonion2 (Channel O2 button)

A random octonion Julia fractal is generated. The essential octonion parameters are randomly adjusted to point into an eight-dimensional formula H0-H9, and then the octonion Mandelbrot set is scanned to find an interesting area to zoom into. The ranges are reset, and the lighting is set for optimum viewing.

This option uses the octonion formulas H0-H9, with random dimensional switching (one of OE-OK for O<sub>i</sub>) to create octonion fractals that may extend to eight dimensions.

## 2.24 Channel LS

### Random Lsystem (Channel LS button)

A random lsystem fractal is generated. An .ls file is selected at random from the startup directory, the palette randomized, and then a random amount of mutation is applied. The lighting is set for optimal viewing, and the figure rotated a random amount on the x y and z-axis.

## 2.25 Save

### Save button

Use this button to save and name the active drawing. Fractal Imaginator displays the Save As dialog box so you can name your drawing.

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

## 2.26 Load

### Load button

Use this button to open an existing data/image file in a new window. You can open multiple image files at once. Use the Window menu to switch among the multiple open images.

## 2.27 Bmp

### BMP button

Use this button to select the BMP format when loading and saving fractals. This is the default Windows bitmap format, readable by most Windows programs that use image files. This is also the fastest method of loading and saving fractals, but requires more disk space, since no compression is used. Windows keeps track of the last six BMP files saved or loaded (displayed in the Files menu.)

## 2.28 Png

### PNG radio button

Use this button to select the PNG format when loading and saving fractals. This format uses medium compression without loss of image quality.

## 2.29 Jpg

### JPG radio button

Use this button to select the JPEG format when loading and saving fractals. This format uses moderate compression but with some loss of image quality. Preferable for posting to the net, since most browsers can display jpeg files.

## 2.30 |||||

### ||||| button

Through a series of windows, this allows you to name and open an avi animation stream and choose a compression method. After choosing the frame rate (1-60) and using the file requester to name the file, you are given a choice of compression methods. You can also choose no compression for optimum view quality. (All compression methods degrade the original images, some more than others.) The first key frame in the stream is then drawn and written to the file.

Note: after the stream is opened, the size of the fractal that can be drawn is fixed at the size of the frame. No changes can be made to the size until the stream is closed.

**2.31 >****> button**

With this option, frames are written to a stream based on the difference between the current key frame and the previous key frame. The first key frame is written when you open a stream. The next key frame is created each time you use this option. In between you can zoom or change Fvr variables as much as necessary. The stream is only written to when this option is used. The last key frame is automatically saved after the 'tween' series is written. The number of frames may range from 1-1500 frames between keys. With a frame number of 1 only the key frames are written. This allows animation to be created that incorporate all scalable variables in Fractal Imaginator.

Use the Cancel button to exit this dialog without initializing a new series of frames.

Check the Log Scaling box if you want the frames to be written with logarithmic space between frames, else linear space is used. Useful when zooming, where frames would otherwise be packed together at the end of the frame series.

**2.32 []****[ ] button**

Closes any open avi stream file. You need to do this before viewing the file or creating a new avi file. The stream is also closed when you exit Fractal Imaginator.

**2.33 V****V button**

Opens an avi file for viewing. You can preview any multimedia file by clicking on its file name. A multimedia box will appear to the right of the file list. Click on okay to open the main view window.

There are buttons to Play a file forwards or Backwards, or forward automatically with Auto rewind/repeat. Click on Slow to slow down a video. Each click on Slow halves the viewing speed. A click on Stop freezes viewing and restores the view speed to normal playback.

Use the Open button to view a different avi file. Use the Save button to save the file in a different compression format. You must use a different name to save the file than the name that was used to open it. Click on the left-mouse button or any key to abort a save operation.

Note: the view avi requester can be used to preview any multimedia file, including midi files.

## 3 File menu

### File menu commands

The File menu offers the following commands:

<a href="#">New</a>	Creates a new drawing.
<a href="#">Open</a>	Opens an existing drawing.
<a href="#">Close</a>	Closes an opened drawing.
<a href="#">Save</a>	Saves an opened drawing using the same file name.
<a href="#">Save As</a>	Saves an opened drawing to a specified file name.
<a href="#">Load Parameters</a>	Load parameters from an existing drawing.
<a href="#">Load Par File</a>	Load a parameters definition file.
<a href="#">Load Palettes [PL]</a>	Load palettes file.
<a href="#">Open [JPEG]</a>	Load jpeg.
<a href="#">Open [PNG]</a>	Load png.
<a href="#">Save Parameters</a>	Save parameters for an opened drawing to a specified file name.
<a href="#">Save Par File</a>	Save a parameters definition file.
<a href="#">Save Palettes [PL]</a>	Save palettes to file.
<a href="#">Save As [JPEG]</a>	Save in jpeg format.
<a href="#">Save As [PNG]</a>	Save in png format.
<b>Import</b>	
<a href="#">Load FraSZle Parameters [FSZ]</a>	Load a FraSZle data file.
<a href="#">Load Palette [PQZ]</a>	Load QuaSZ palette file.
<a href="#">Load LSystem [LS]</a>	Load a LParser compatible file.
<a href="#">Load Palette [MAP]</a>	Load a Fractint map file.
<b>Export</b>	
<a href="#">Save Palette [PQZ]</a>	Save current palette.
<a href="#">Save LSystem [LS]</a>	Save a LParser compatible file.
<a href="#">Save QuaSZ Parameters [QSZ].</a>	Save quaternion variables to QuaSZ data file
<a href="#">Save Q Polygon [OBJ]</a>	Save polygonized quaternion as Wavefront object.
<a href="#">Save Q Polygon [STL]</a>	Save polygonized quaternion as STL solid file.
<a href="#">Simplify Mesh</a>	Simplify mesh.
<a href="#">Save Q Polygon [POV]</a>	Save polygonized quaternion as a pov triangle object.
<a href="#">Smooth Triangles</a>	Convert triangle mesh to smooth_triangle mesh.
<a href="#">Save Q Polygon [WRL]</a>	Save polygonized quaternion as virtual reality file.
<a href="#">Save Q Polygon [DXF]</a>	Save polygonized quaternion as AutoCad dxf file.
<a href="#">Set Max Vertices</a>	Set maximum number of vertices allocated for Q polygon.
<a href="#">Save Lsystem to OBJ</a>	Save Lsystem to Wavefront Object file.
<a href="#">Save Lsystem to STL</a>	Save Lsystem to STL solid file.
<a href="#">Save Lsystem to POV</a>	Save Lsystem as POV triangle object.
<a href="#">Save Lsystem to DXF</a>	Save Lsystem to AutoCad dxf file.
<a href="#">Mesh Setup</a>	Edit/View parameters for exporting meshes.

[Exit](#)

Exits Fractal Imaginator.

### 3.1 File New command

#### New command (File menu)

Use this command to create a new drawing window in Fractal Imaginator. The image and data for the opening picture are used to create the new window.

You can open an existing data/image file with the [Open command](#).

#### Shortcuts

Keys: CTRL+N

### 3.2 File 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. Use the Window menu to switch among the multiple open images. See [Window 1, 2, ... command](#).

You can create new images with the [New command](#).

#### Shortcuts

Toolbar:   
Keys: CTRL+O

#### 3.2.1 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. This box lists files with the extension you select in the List Files of Type box.

##### List Files of Type

Select the type of file you want to open.

##### Drives

Select the drive in which Fractal Imaginator stores the file that you want to open.

##### Directories

Select the directory in which Fractal Imaginator stores the file that you want to open.

##### Network...

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

### 3.3 File 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.

You can also close a drawing by using the Close icon on the drawing window, as shown below:



### 3.4 File 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, Fractal Imaginator 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](#).

#### Shortcuts

Toolbar:   
Keys: CTRL+S

### 3.5 File Save As command

#### Save As command (File menu)

Use this command to save and name the active drawing. Fractal Imaginator 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](#).

#### 3.5.1 File 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. Fractal Imaginator adds the extension .fim.

**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.

## 3.6 File Load Parameters command

### Load Parameters command (File menu)

Use this command to load a data file [.fim or .zp]. The data file contains all variables to recreate an image created previously with Fractal Imaginator.

## 3.7 Load Par File

### Load Par File

Opens a window that allows the user to select and convert Fractint/Fractal Imaginator parameter files to Fractal Imaginator format. A list of par titles is displayed for each .par file loaded. The user then selects a title and clicks on Convert to translate it to the current image data. This works for most Fractint v18 and v19 escape-time fractals up to 19.3 (and is downward compatible with most Fractal Imaginator fractals.) Fractint 19.6 par files with internal formula definitions (and/or the if/then/else structure) are supported. Excluded are the frothy basin and complex phoenix types. Formula types may now be loaded, although some options such as rand and LastSqr are not supported. (Some built-in functions like p0 use the LastSqr speedup.) 3-D fractals and other types, such as Orbit Fractals and Bifurcation are not translated. Some of Fractint's 2D options are implemented differently in Fractal Imaginator, so the translation may not be exact.

Some Fractint functions are not supported through the f1-f4 boxes, but when they are encountered in a par file, Fractal Imaginator writes the name of the unsupported function in the fun#2 box.

The Fractint palette specified in the colors option (if any) is loaded into F12. Since Fractal Imaginator uses a 236-color palette (interpolated to 60161 colors in Fractal Imaginator), the Fractint palette is condensed to fit Fractal Imaginator's palette size. The attempt here is to match color spreads rather than lone colors. So some Fractint palettes may need to be edited to work. The skew parameter in the center-mag coordinate mapping mode is not supported.

Select Load Palette to load a par palette *only* without changing the current function data.

Related Topics:

[Edit Palette](#) describes the Fractal Imaginator palette editor.



### 3.8 File Load Palettes command

#### Load Palettes command (File menu)

Use this command to load a palette file [.pl]. The palette file contains 21 palettes created previously with Fractal Imaginator (or another version of the program.)

### 3.9 File Open [JPG] command

#### Open [JPEG] command (File menu)

Use this command to load parameters and a bitmap file that were saved in jpeg format. There is an option in the file-type box to load only the bitmap too. This replaces the Open command for those who need a smaller sized bitmap file. Note: the last files list doesn't keep track of images loaded in JPEG format.

### 3.10 File Open [PNG] command

#### Open [PNG] command (File menu)

Use this command to load parameters and a bitmap file that was saved in png format. There is an option in the file-type box to load only the bitmap too. This replaces the Open command for those who need a smaller sized bitmap file. Note: the last files list doesn't keep track of images loaded in PNG format.

### 3.11 File Save Parameters command

#### Save Parameters command (File menu)

Use this command to save all data elements for the current image in a data file [.fim].

### 3.12 Save Par File

#### Save Par File

This option allows you to save Fractal Imaginator parameters and color info in a text format similar to Fractint's .par format. You can build par libraries of your favorite fractals to share with other users of Fractal Imaginator. The par definitions can be easily posted on the net and reloaded in Fractal Imaginator through the Load Par command.

Specify the par file name with the Filename button. This can be a generic file name, such as "complex.par", for a library of par titles, or a specific file name based on the fractal's .fim. The later file name is the default. Use the Par Title box to specify the fractal's name or description. Add a comment to the par definition with the Comment box. When you click on Okay, if a file with the Par Filename doesn't exist, it will be created and the par definition added to it. If the file exists, Fractal Imaginator will scan the file for a par title the same as

the one being added. If it doesn't exist, the par definition will be added to the end of the par file. If it exists, you have the option to replace the old definition with the new one. If you choose not to replace the old definition, the existing par file remains unchanged and no further action is taken.

Compatibility issues: Since Fractal Imaginator has many options not found in Fractint, no attempt was made to make the Fractal Imaginator par format compatible with Fractint's. The object was to make Fractal Imaginator picture parameters more accessible to Fractal Imaginator users. As formula files add another unnecessary layer to the parameter puzzle, Fractal Imaginator's par format saves any user-defined formulas in the par definition rather than creating a separate formula file. The color info is converted into a condensed ASCII format for the par definition that also differs from Fractint's color-coding.

### **3.13 File Save Palettes command**

#### **Save Palettes command (File menu)**

Use this command to save all palettes for the current session in a palette file [.pl].

### **3.14 File Save As [JPG] command**

#### **Save As [JPEG] command (File menu)**

Use this command to save the parameters and active bitmap in jpeg format. There is an option in the file-type box to save only the bitmap too. This replaces the Save and Save As command for those who need a smaller sized bitmap file. Note: the last files list doesn't keep track of images saved in JPEG format.

### **3.15 File Save As [PNG] command**

#### **Save As [PNG] command (File menu)**

Use this command to save the parameters and active bitmap in png format. There is an option in the file-type box to save only the bitmap too. This replaces the Save and Save As command for those who need a smaller sized bitmap file. Note: the last files list doesn't keep track of images saved in PNG format.

### **3.16 Load FraSZle Parameters [FSZ]**

#### **Load FraSZle Parameters [FSZ]**

Use this command to load a FraSZle data file [.fsz]. The data file contains all variables to recreate an image created previously with FraSZle.

### 3.17 File Load Lsystem command

#### Load Lsystem [LS] command (File menu)

Use this command to load an LParser-compatible .ls file. The level of recursion, basic angle, starting line width, axiom and rules are extracted from the file and converted to Fractal Imaginator format.

### 3.18 File Load Palette [PQZ] command

#### Load Palette [PQZ] command (File menu)

Use this command to load a QuaSZ-style palette file [.pqz]. The palette file contains a single palette that replaces the current palette.

### 3.19 File Load Palette [MAP] command

#### Load Palette [MAP] command (File menu)

Use this command to load a Fractint-type map file. The palette in the map file replaces the currently selected palette.

### 3.20 File Save Palette command

#### Save Palette [PQZ] command (File menu)

Use this command to save the current palette to a QuaSZ-style palette file [.pqz].

### 3.21 File Save Lsystem command

#### Save Lsystem [LS] command (File menu)

Use this command to save the current lsystem in an LParser-compatible .ls file. This is useful for exporting an lsystem to Crocus or another program that supports the LParser format.

### 3.22 File Save QuaSZ command

#### Save QuaSZ parameters command (File menu)

Use this command to save the quaternion variables that make up the current figure in a QuaSZ-compatible data file [QSZ]. You can load this file into QuaSZ via the File/Load Parameters command.

### 3.23 File Save Q Polygon [OBJ] command

#### Save Q Polygon [OBJ] 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 Wavefront object file. The memory requirements for this routine are high, 20MB or more for a typical Julia set quaternion rendered at 320X240. The output file can be very large too, 4MB or more, depending on the precision required. The higher the precision, the smoother the finished object. Precision is set with the Steps variable in the Quaternion window, where  $\text{precision} = 10/\text{Steps}$ . See also the [Simplify mesh](#) command for ways to reduce object file size.

Note: some formulas produce unsymmetrical object files with this routine, where one side of the q polygon isn't resolved completely. Usually one side is markedly smoother in this case.

### 3.24 File Save Q Polygon [STL] command

#### Save Q Polygon [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 memory requirements for this routine are high, 20MB or more for a typical Julia set quaternion rendered at 320X240. The output file can be very large too, 2MB or more, depending on the precision required. The higher the precision, the smoother the finished object. Precision is set with the Steps variable in the Quaternion window, where  $\text{precision} = 10/\text{Steps}$ . See also the [Simplify mesh](#) command for ways to reduce object file size.

Note: some formulas produce unsymmetrical object files with this routine, where one side of the q polygon isn't resolved completely. Usually one side is markedly smoother in this case.

### 3.25 File Simplify mesh command

#### Export -> Simplify Mesh command (File menu)

When this flag is set (default on) the object meshes are simplified using Garland's mesh-simplification algorithm before outputting to a Wavefront obj or POV mesh file, resulting in a much smaller export file. Set the number of facets in the target mesh file with the Mesh Setup command. You can set the resolution of the object as high as necessary (with the Params/Steps variable) to produce a finely detailed quaternion, but the output file remains about the same. Use the smoothing feature in Bryce to smooth the resulting object mesh.

### 3.26 File Save Q Polygon [POV] command

#### Export Q Polygon [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 outputs the triangles to a pov file. The pov file is written as a simple scene, the triangles part of a "union" object, with camera and lighting elements compatible with POV 3.5. This can be used as a starting point for more complex compositions. The memory requirements for this routine are 20MB or more for a typical Julia set quaternion rendered at 320X240. The output file can be very large too, up to 40MB or more, at the highest precision. The higher the precision, the smoother the finished object becomes. Precision is set with the Steps variable in the Quaternion window, where  $\text{precision} = 10/\text{Steps}$ . See also the [Simplify mesh](#) command for ways to reduce object file size.

Note: some formulas produce asymmetrical object files with this routine, where one side of the q polygon isn't resolved completely. Usually one side is markedly smoother in this case.

### 3.27 File Smooth command

#### Export -> Smooth command (File menu)

When this flag is set (default on) the object facets are converted to smooth\_triangles before outputting to a POV mesh file. Surface normals are calculated for all triangles that share common vertices.

### 3.28 File Save Q Polygon [WRL] command

#### Save Q Polygon [WRL] 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 virtual reality file. The memory requirements for this routine are high, 20MB or more for a typical Julia set quaternion rendered at 320X240. The output file can be very large too, 4MB or more, depending on the precision required. The higher the precision, the smoother the finished object. Precision is set with the Steps variable in the Quaternion window, where  $\text{precision} = 10/\text{Steps}$ .

Note: some formulas produce unsymmetrical object files with this routine, where one side of the q polygon isn't resolved completely. Usually one side is markedly smoother in this case.

### 3.29 File Save Q Polygon [DXF] command

#### Export Q Polygon [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 outputs the

triangles to a dxf file. The memory requirements for this routine are 20MB or more for a typical Julia set quaternion rendered at 320X240. The output file can be very large too, up to 40MB or more, at the highest precision. The higher the precision, the smoother the finished object becomes. Precision is set with the Steps variable in the Quaternion window, where  $\text{precision} = 10/\text{Steps}$ .

Note: some formulas produce asymmetrical object files with this routine, where one side of the q polygon isn't resolved completely. Usually one side is markedly smoother in this case.

### **3.30 File Set Max Vertices command**

#### **Set Max Indices (File menu)**

Use this command to set the maximum number of indices that are allocated by the polygonizing routine. Default is 500,000 indices. Use less to limit the amount of memory used while polygonizing. Use more if necessary for higher resolution. Note: unless you have an application that can use very large object files, there's a limit to how much resolution is obtainable with the polygonizing routine. Bryce4 has problems with object files produced by Fractal Imaginator that are much larger than 2.5MB.

### **3.31 File Save Lsystem to Obj command**

#### **Save Lsystem to Obj command (File menu)**

Use this command to save an lsystem as a true 3D object. This is useful to export an lsystem for use in Bryce or another program that supports the Wavefront format. Precision is defined by the number of Steps in the Draw LS window (default 1). Hint: You can reduce the complexity of an lsystem (and its obj file size) by decreasing the Max. Lines variable.

### **3.32 File Save Lsystem to Stl command**

#### **Save Lsystem to Stl command (File menu)**

Use this command to save an lsystem as a stereolithographic solid object. Precision is defined by the number of Steps in the Draw LS window (default 1). Hint: You can reduce the complexity of an lsystem (and its stl file size) by decreasing the Max. Lines variable.

### **3.33 File Save Lsystem to POV command**

#### **Save Lsystem to POV command (File menu)**

Use this command to save a lsystem as a POV triangle object. Precision is defined by the number of Steps in the Draw LS window (default 1). Hint: You can reduce the complexity of an lsystem (and its pov file size) by decreasing the Max. Lines variable.

### **3.34 File Save Lsystem to Dxf command**

#### **Save Lsystem to Dxf command (File menu)**

Use this command to save an lsystem as a true 3D object. This is useful to export an lsystem for use in Bryce or another program that supports the AutoCad dxf format. Precision is defined by the number of Steps in the Draw LS window (default 1). Hint: You can reduce the complexity of an lsystem (and its dxf file size) by decreasing the Max. Lines variable.

### 3.35 Mesh Setup command

#### Mesh Setup command (File menu)

Here you edit or view the parameters for simplifying meshes when outputting in Wavefront [obj] or POV format.

Max export faces determines the size of the export mesh, if the original mesh is larger than the target size. Triangle faces are "collapsed" until the target face size is reached.

Max input faces and max input vertices determine how much memory is set aside as buffers for processing the meshes. Increase or decrease from the default values as the size of the mesh warrants, or as system memory permits.

The "min width" variable controls how narrow individual segments in a lsystem are allowed to shrink at the highest level of recursion. Increase this value if the cylinders in the object file become too narrow to render correctly in Bryce. Decrease if the cylinders look oversized.

The weld factor controls how close adjacent triangle vertices of a mesh may be before they are merged into one vertex. This effectively flattens adjacent triangles or "collapses" them and reduces mesh size. Use a small enough weld factor that produces an evenly simplified mesh without destroying the integrity of the smallest elements of the mesh. Use the smoothing routine in Bryce to restore the mesh to optimum curvature. POV triangle meshes are further processed into "smooth\_triangles" before exporting into the final mesh file.

### 3.36 File 1, 2, 3, 4, 5, 6 command

#### 1, 2, 3, 4, 5, 6 command (File menu)

Use the numbers and filenames listed at the bottom of the File menu to open the last six drawings you closed. Choose the number that corresponds with the drawing you want to open.

### 3.37 File Exit command

#### Exit command (File menu)

Use this command to end your Fractal Imaginator session. You can also use the Close command on the application Control menu. Note: if you choose to exit while plotting, the program does not terminate, but stops the plotting so the program can be safely exited.

#### Shortcuts

Mouse: Double-click the application's Control menu button.



Keys: ALT+F4

## 4 Edit menu

### Edit menu commands

The Edit menu offers the following commands:

<a href="#">Undo</a>	Undo last edit, action or zoom.
<a href="#">Copy</a>	Copy the active view and put it on the Clipboard.
<a href="#">Paste</a>	Insert Clipboard contents.
<a href="#">Copy Data</a>	Copy fractal data to buffer.
<a href="#">Paste Data</a>	Copy data from copy buffer.
<a href="#">Formulas/Type</a>	Edit formula/type data.
<a href="#">Fractal Variables</a>	Edit fractal variables.
<a href="#">Cubic Values</a>	Edit cubic parameters.
<a href="#">Octonion Values</a>	Edit octonion constants.
<a href="#">Size</a>	Sets the image size.
<a href="#">Ray-Tracing Variables</a>	Edit lighting and viewpoint variables.
<a href="#">Palette Editor</a>	Edit palette.
<a href="#">Text</a>	Edit and add text to drawing.
<a href="#">Preferences</a>	Startup preferences and defaults.

### 4.1 Edit Undo command

#### Undo command (Edit menu)

Use this command to undo the last action. An image can be continued after an undo, if continue was enabled before the last action. Color-cycling is disabled after using Undo, though.

#### Shortcut

Keys: CTRL+Z

### 4.2 Edit Copy command

#### Copy command (Edit menu)

Use this command to copy the active view to the clipboard. The entire view is copied to the clipboard.

#### Shortcut

Keys: CTRL+C



## 4.3 Edit Paste command

### Paste command (Edit menu)

Use this command to paste from the clipboard. The clipboard must contain a bitmap. If the bitmap is larger than the view, it is clipped. The zoom cursor is used to set the left/top corner in the view where the bitmap will be pasted. Click outside the view frame or press escape to exit this option.

#### Shortcut

Keys: CTRL+V

## 4.4 Edit Copy Data command

### Copy Data command (Edit menu)

Use this command to copy the fractal data for the active view to the file "(defaultdir)\zcopy.fim". The current palette for the view is also copied.

#### Shortcut

Keys: CTRL+F

## 4.5 Edit Paste Data command

### Paste Data command (Edit menu)

Use this command to paste the data in the file "(defaultdir)\zcopy.fim" to the active view. The palette stored in the file is copied to palette 10(F11).

#### Shortcut

Keys: CTRL+R

## 4.6 Formula Window

Fun #1 and Fun #2 are combo controls for entering/selecting up to two formulas either from their drop-down lists or in the custom form of  $Z = AZ + BZ + c$ .  $Z$  is the complex variable or function, 'c' is the complex constant, and A and B are optional real constants. There are additionally a Type control, an Arg control and an Arg Limit control that determine how the above formulas are processed. Note: when the fractal type is quaternion, custom formulas are only processed through the Fun #1 or the Formula box. The Fun #2 combo only uses the

built-in formulas in its drop-down list.

The Type control accepts a value of 0 to 11. 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 values 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 1000 characters per formula.

Text can be pasted from the clipboard to the formula box by using the keystrokes shift-insert. Text may be moved from box to box by using shift-delete to move it first to the clipboard.

Type 10 produces Escher-like tilings for Julia sets, as described in *The Science of Fractal Images*. This uses both fun#1 and fun#2 in combination. Fun#1 sets the stage for the target set fun#2. To reproduce the images in *The Science of Fractal Images*, use  $z=z*z$  for fun#1 and  $p0$  for fun#2. Fun#1 may be any formula other than  $z=z*z$ , but may produce unsymmetrical tilings. Fun#2 can be any built-in formula. This type is mainly applicable to Julia-type sets, but may be used with limited effect with the MandelbrotP-type set. Use the Rise box in the Parameters window to set the degree of tiling desired. The higher the Rise number (1-235), the greater the density of the tiling. Note: for quaternion "Escher" fractals, the Rise number has no effect, and Fun#1 is processed without hypercomplex extensions. This results in a 3D tiling effect on the x/y axis that can be multiplied by increasing the number of terms in Fun #1, as in  $z=z*z*z$ .

Type 11 uses Fun#1 to produce "genetic" style fractals. Here you enter a character string or word that can contain any of 25 letters 'a' through 'y'. Each letter or "gene" references a built-in formula, as defined in the Genes window. For every iteration the program cycles through the character string and uses whatever formula is referenced by the characters. Notes: If Max. Iter is shorter than the gene string then only part of the string is used. For quaternions, which usually have a very short iteration cycle, changing Max. Iter can have a dramatic effect on the

fractal output.

The f1-f4 combo boxes are used to designate the 'fn' user-definable part of a generalized function. This can be one of the 41 function types listed (sin(w),cos(w) etc.) A few of the options are formulas themselves (L3 (w) the Legendre function, the gamma function and G(w)the Gaussian function), so quite complex (though mathematically unintelligible) formulas are possible.

The S control is used to enter the variable 's' used in many of the built-in functions. The Si control is used to enter the variable 'si' the imaginary component for s in some of the built-in functions.

The Converge control is used to enter the convergence limit for Renormalization or Convergence plots. This is a measure for how close z needs to approach an attractor to be mapped to that attractor color. A small value .0001-.000001 is normally used. Smaller values of the convergence variable produce a more accurate plot, while increasing the computation time somewhat. On some formulas it may be necessary to reduce this variable to its smallest value (.00000001) to eliminate some artifacts (spots) caused by non-convergence (at a higher limit value.)

For the random displacement routines, the S box is used to control the fractal dimension (valid inputs are 0-1.0). A lower number gives the fractal a higher dimension, and thus more ragged and choppy appearance. In 3D mode the magnify box works as for other 3D plots, except that a value of .4-1.0 is adequate for most plots. For 2D plots or backgrounds, the magnification factor is used as a color scalar. A value less than 1.0 reduces the number of colors by the same factor. This is necessary to create more realistic clouds. The color palette for 2D plots is rotated by the cutoff number. This allows some independent color control given the limitations of a palette-based system. The arg box is used to control shaping and fullness of the finished plot. The plot may include random additions at every point or just the end points. The shape may be scaled linearly or non-linearly. The following values for the arg box control random additions and shape:

- |   |                     |
|---|---------------------|
| 0 -- random additions at end points only; | linear shaping.     |
| 1 -- random additions at every point;     | linear shaping.     |
| 2 -- random additions at end points only; | non-linear shaping. |
| 3 -- random additions at every point;     | non-linear shaping. |

About formula syntax: This applies if you elect to enter your own formula into one of the function boxes and use the parser to generate the plot. The use of parenthesis is necessary around complex exponents or variables. E.g.: '(z-i)^(1.5e)'. For a complete list of variables, operators and functions recognized by the parser, see [Parser Information](#).

Up to 500 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. Fractal Imaginator 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 preceding ';'. Some variables such as 'pixel' and p1 are

named after Fractint's predefined variables. These are provided to allow Fractal Imaginator users to more easily convert Fractint formula types to Fractal Imaginator use. However, Fractal Imaginator doesn't prompt you to enter values into p1 (the cr and ci boxes) or p2 (the s and si boxes) or p3 (the limit and converge boxes). Since p1 is used in the iteration process as 'c', p1 cannot be used as a variable independent of c. 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. Use ctrl-enter to terminate a line in the formula box. An optional bailout routine may be entered as a phrase at the end of the formula. If the bailout phrase equals a value other than TRUE during iteration, the iteration loop is exited. There are other flags such as Convergence and Biomorph, if set, that can force exit also.

The arg control is limited to 25 characters.

The Title text box is used with the hot key 'T' to annotate a picture with text. Use the Edit/Text command to change font, text color or format text into multiple lines. Text in this box is not saved in a picture's data file, but once entered the same text can be used over and over for different pictures. Useful for adding copyright/author info to batches of pictures. Since the same title text may be used many times, it is shared among views and saved in the file "prefs.txt" in Fractal Imaginator's startup directory.

Click on the Apply button to use the formulas currently displayed in the window. Use the Okay button to apply the formulas and close the window, or use Cancel to exit the window without making any changes or cancel current changes. If changes have been applied before clicking on Cancel, then the formulas will revert to the state when the window was last opened. Note: commands exterior to this window (as in the Demo menu) may cause the window to be closed and reopened with new values. In this case Cancel only returns the formula values to the "new" values.

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

The three buttons named Rand fun#1, Rand fun#2 and Rand f1-f4 are used to pick formulas/functions at random. Clicking on Rand fun#1, a formula is chosen (from the 100 built-in formulas) for fun #1. Clicking on Rand fun#2, a formula is chosen for fun #2. Clicking on Rand f1-f4, functions are chosen for f1-f4.

Select FraSZle Formulas to switch to the FraSZle formula set for more compatibility with QuaSZ. You can then export any quaternion done with Fractal Imaginator as a QuaSZ parameter file [QSZ] and import it into QuaSZ for more advanced surface coloring, etc.

## 4.7 Fractal Variables

### Fractal Variables (Edit Menu)

The window opened varies with the fractal type selected, and contains all the major variables that Fractal Imaginator now scales between key frames of an avi stream.

### 4.7.1 Parameters Window

There are edit controls for entering the complex constant (real and imaginary parts), and the min/max ranges for the real and imaginary window coordinates. Fractal Imaginator uses three-corner plotting for easier rotating, so boxes are provided for Top Left, Top Right and Bottom Right real/imaginary coordinates. These reflect the current range values that may have been derived from zooming with the Zoom option. Edit controls affect the number of iterations (1-65000), or the bailout (1-65000). Cj, ck, and hj are for entering hypercomplex parameters.

The Starting cr, Starting ci, Increment cr and Increment ci are used with the Growth examples of BBM 5 and BBM 20, image sets of 25 and 441 respectively. The Julia sets are initialized with incremented complex constants to form either Julius sets (BBM 20) or sets of 25 images like Dante Growth.

The Orbit limit determines a bailout for the outer portion or escape zone of the Julia basin. A higher value allows more of the escape rings to be viewed if the Growth/Orbits option is off. A smaller value excludes more of the escape zone or "environment" in BBM terms.

The Set Limit tends to reveal more of the inner "core" or structure of a Julia set. You want to adjust this variable so that the structure is as detailed as possible. Too high a value will produce a solid core of one color. Too low a value will produce circular patterns that obscure the core. When Cores=2, the Set limit can have a lower value and the detail can be increased by adjusting both Set Limit and Orbit Limit. The challenge is to find the right values for these variables to produce the most detailed (and beautiful) Julia set. Values can be very large for the Orbit Limit and very small for the Set Limit, so each of these variables is entered in exponential form, as in "1E-003" or "4E+085". When the Growth command Next Set is executed the Set Limit and Orbits Limit are multiplied by their incremental complements, Set Dec and Orb Inc. When the Growth command Previous Set is executed the Set Limit and Orbits Limit are divided by their incremental complements, Set Dec and Orb Inc.

The more iterations used, the longer it takes to plot a function, but more detail will be present. 10-20 is sufficient for most biomorphs, while more iterations will be required for Mandelbrot and Julia sets, depending on the detail required. Bailout is used mainly when Cores=0 for added compatibility with rendering options like Biomorph and Orbit-traps. A lower value should be used (4.0) when these options are selected. A higher value (65000) should be used with default Growth examples. But these are just suggestions. Feel free to experiment with any variable, as that is where the fun is.

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

The Cutoff box acts as a palette multiplier or divider, depending on whether the value entered is less than or greater than 1.0. The palette color is divided by the Cutoff to speed up or slow down color changes. Cutoff values are limited to a low minimum of .001. For level curves and the add and multiply color-scaling options, use a negative cutoff value to maintain a

smooth palette. This ensures that the multiplier is used before the (floating-point) palette values are converted to (integer) palette indexes.

Related Topic:

[Quaternion](#) describes the Quaternion generator's data-collection window.

## 4.7.2 Quaternion Window

### Quaternion Window

This is the data-collection window for Fractal Imaginator's Quaternion generator. Minx, maxx, miny and maxy are the spatial variables for framing the quaternion object. These are usually updated automatically when you use the zoom box. Min Z and Max Z define the three-dimensional space that is used to map the quaternion image. Normally Min Z is the negative of Max Z, but Min Z can be adjusted in the positive direction to shear off the front of the quaternion object. This has the effect of exposing the insides of a quaternion. Const1-4 are cr, ci, cj and ck respectively. Maxiter is the same as iterations in the Parameter's window. Three rotate variables determine the 3D angle of rotation. Step and Fine are pitch adjustments that bear on the quality of the plot at an expense of lengthier calculations.

When you click on Okay, the quaternion generator looks at the Fun#1 gadget in the New Formula window. If this contains a preset variable (P0-O9) that function is iterated for its escape time, then the results are ray-traced in any 3D object that may be created. If the Type is set to 9, any custom formula (Fractint-style: as in 'z=z^3+c#') in the Formula box is used. Not all functions may produce a usable 3D object with this method, but it's interesting to experiment with. The Quaternion option works for both Mandelbrot and Julia sets, depending on which type is selected.

The Slice variables may be altered to display different 3-D planes of the quad set.

The Plane variables B (Back), F (Front), and P (Position) allow you to flatten part of the quaternion figure. For normal plotting, these variables default to 0. To have any effect on the image, the Back variable must be greater in value than the Front variable. The difference between back and front variables determines where the image is flattened. These variables are limited to +/-9.99, with normal values being in the range  $-z$  to  $+z$ . The position variable sets the color of the flattened plane. Note: if you set the Back variable less than  $-z$ , and the front variable greater than or equal to  $-z$ , you can get a gradient in the background, depending on the lighting and coloring settings. This sometimes has the effect of placing part of the quaternion in a fog-like dimension.

## 4.7.3 Axiom Window

### Axiom Window

Enter the basic angle, 0.5-360. This angle will be used for any turning command that doesn't use an additional argument. Enter an axiom, up to 255 characters in the axiom box. Enter the initial width of the line, ( $\geq 1$ .) Angle and width variables can be non-integer. A line with

width greater than one causes the turtle to draw a tubular line in 3D space. See the 's' rule (Demo/Lsystem info) in the extended rule set for info on changing the line shape. Click on Apply to apply the current axiom to the image. Click on Okay to set the new axiom into memory and close the window. Click on Sample or Random Sample to use one of the pre-existing lsystems in the startup directory. Click on Rules to edit the rules associated with the current axiom.

#### 4.7.4 Rules Window

##### LSystem Rules

The top row contains the command line buttons, and the input boxes below are used to enter production rules and labels.

The input boxes:

'Rule' -- enter up to 255 characters for each production rule.

'Label' -- the label for the production rule. Can be any ASCII character. If the character has no keyboard equivalent, you can enter it as its decimal number (0-255.) This may or may not produce a visible character when drawn in the rules list. Using shift-delete and shift-insert for these characters enables them to be copied and pasted into any rule.

The command line buttons:

'Add' -- adds whatever production rule has been entered into the string gadgets, if the label does not duplicate a label already in use. Up to 255 rules can be entered. The window below the string gadgets displays the current production rules in use. Use the slider device to scroll through the rules, if more than nineteen rules are used.

'#' & 'Recall' are used in conjunction to edit a previously added rule. Enter the number of the rule, as displayed below, into the # box and click on Recall. The rule is entered into the input boxes. (Alternately, you may click on a displayed rule with the cursor and left-mouse button, and it will be recalled.) This enables 'Change' and 'Delete' to change or delete the rule. An extra step here prevents changing or deleting the wrong rule by mistake. Hint: you can also change the rule and then Add it in the same operation, if the label is changed. Use 'Cancel' to change another rule, if you don't really want to change or delete the rule you Recalled. (# box is disabled during a Recall operation.)

'Cancel' -- cancels the recall operation.

'Insert' -- inserts a rule before the numbered rule as entered in the # box. You can insert a rule after recalling and changing it, too. You can't insert a rule with a label already in use. You must add at least one rule before you can insert a rule.

'Okay' -- terminates the edit-rule session and closes the edit rule window.

## 4.8 Cubic Values Window

To support the cubic Mandelbrot formulas (g0 thru g9), variables have been added to adjust z-origin and magnify the z-plane while calculating pixel depth. Normally you can keep origin set at (0,0,0) and z-mag at 1.0. But these can be useful to tweak in small increments when drawing Mandelbrot quaternions. Then a small change in z-origin can rotate details on a close-up slightly, so you can adjust the view accordingly. Each shift in z-origin will require re-zooming to get back to the area of interest. The z-mag variable makes the z-plane non-symmetrical; pushing up some details while other details recede. So it too is useful to change viewpoint. The affect on Julia quaternions is less noticeable for z-origin shifts. You can usually re-zoom to produce the same Julia image.

Cubic Mandelbrots quaternions use the S and Si variables to set the initial value and range of the 3rd dimension. Starting from an initial value of 'S', Si is normally set to  $2 * \text{abs}(S)$ , when the Type is Mandelbrot. Si can also be set to center the 3rd dimension on something besides 0.0. Si has no affect in 2-D hypercomplex mode. (In 2-D mode, the S variable acts as one of the two fixed dimensions, along with the 4th Center variable.)

The 4th Center points to the center of the fourth dimension (with quaternions/hypernions the 4th Dimension variable is used.)

In addition, the Arg value 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.



## 4.9 Octonion Values Window

### Octonion Values Window

Octonions (built-in functions H0-H9) have a form of  $xr+xi+xj+xk+xE+xI+xJ+xK$ . Additional options are entered via the Arg box in the [Edit/Formula window](#). Use the Randomize button to set a random value for octonion planes E-K and complex constants cj-cK.

## 4.10 Size

### Size (Edit menu)

This allows you to set the drawing area for a picture, independent of the Windows screen size. It also shows which size is currently in use. The aspect for the drawing is based on the ratio of X (horizontal width) to Y (vertical height.) The custom setting allows for any size/aspect that system memory will permit. Videos are limited to the standard 4/3 vga aspect or 1/1. Midi output is limited to images with the standard 4/3 aspect. The minimum size for an image is 40X30. Note: if the image is less than 100 width, the aspect must be 4/3 for solid-guessing to work properly.

## 4.11 Ray-Tracing Window

### Ray-Tracing Window

The Light Point 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 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 object and the apparent sheen on the object's surface. The ambient value controls the amount of ambient light that illuminates the object. 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 started, the lighting variables and light point can be changed without redrawing the object.

Click the Apply button to redisplay a plot after changing the lighting variables or light point. Click the Okay button to close the Ray-Tracing Window, applying new settings, if the variables were modified. Click on Cancel to revert to the state that existed when the ray-tracing window was opened. Click on Defaults to set the lighting and viewpoint variables to the built-in defaults for these variables.

## 4.12 Edit Palette

### Edit Palettes

Use the palette editor to modify the palette(s) in use.

There are copy and spread options to smooth or customize the existing palettes in Fractal Imaginator. You can then save all the palettes in a .pl file, or by saving the entire function and bitmap.

Colors are shown in 8 groups of 29 color indexes, with four colors on the last row. This makes it easy to create divide-by-8, divide-by-4 and divide-by-2 palettes with 232 colors. With Fractal Imaginator, a palette is actually 60160 colors, with each succeeding color (except the last) followed by 255 colors that are evenly spread from one color to the next. The background color for an image is usually index #1.

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 Fractal Imaginator, colors do not cycle smoothly when you adjust the RGB/HSV 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. In the HSV mode, color spreads are based on HSV values instead of RGB values, which in some cases results in brighter color spreads.

Right-click on any point on the main window and the palette color for that pixel will be displayed in the palette editor. You can use any of the color-cycling keys (after clicking on the main window) to see the effects of the cycling in the palette editor window. Note: color cycling and color-selection-from-pixel only works when the image has been drawn in the current session. If you load a pre-existing image file or undo an action, you must redraw it to cycle colors, etc.

Use Reset to reset the colors of the palette in use, to where it was before it was cycled or modified. Note: if you change palettes with one of the function keys, any modifications to a previous palette are unaffected by the Reset button.

Use Reverse to reverse the order of the colors in the palette. This affects only those colors in the start-color to end-color range. Useful for reversing divide-by-eight palettes, etc., for orbit-trap pictures that require a reversed palette.

Use Neg to create a palette that is the complement of the current palette.

Use Smooth to create a random palette with smooth color spreads. Use Scramble to create a palette with random color indexes.

Use SRG to switch the red and green components of all palette colors.

Use SRB to switch the red and blue components of all palette colors. SRB and SRG are disabled in HSV mode. 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. The Randomize variables,

rmin, rmax, bmin, bmax, gmin, and gmax act as limits that are applied after the palette after initial randomizing, to make the palette conform to the desired spectrum of colors.

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.)

#### **4.12.1 Reverse button**

##### **Reverse button**

Use Reverse to reverse the order of the colors in the palette. This affects only those colors in the start-color to end-color range. Useful for reversing divide-by-eight palettes, etc., for orbit-trap pictures that require a reversed palette.

#### **4.12.2 Neg Button**

##### **Neg button**

Use Neg to create a palette that is the complement of the current palette.

#### **4.12.3 Map Button**

##### **Map button**

In Fractal Imaginator, colors do not cycle smoothly when you adjust the RGB/HSV 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.

#### **4.12.4 H/R Button**

##### **H/R button**

Change from HSV to RGB mode or back. In the HSV mode, color spreads are based on HSV values instead of RGB values, which in some cases results in brighter color spreads.

#### **4.12.5 Spread Button**

##### **Spread button**

Select Spread to define a smooth spread of colors from the current spot to another spot in the palette.

#### **4.12.6 Copy Button**

##### **Copy button**

Select Copy to copy any color to another spot in the palette.

#### **4.12.7 SRG Button**

##### **SRG button**

Use SRG to switch the red and green components of all palette colors. RGB mode only.

#### **4.12.8 SRB Button**

##### **SRB button**

Use SRG to switch the red and blue components of all palette colors. RGB mode only.

#### **4.12.9 Okay Button**

##### **Okay button**

Click on Okay to exit the palette editor, applying unmapped color changes to picture (if color-cycling is enabled.)

#### **4.12.10 Reset Button**

##### **Reset button**

Use Reset to reset the colors of the palette in use, to where it was before it was cycled or modified. Note: if you change palettes with one of the function keys, any modifications to a previous palette are unaffected by the Reset button.

#### **4.12.11 Cancel Button**

##### **Cancel button**

You can cancel a copy or spread operation with the Cancel button.

#### **4.12.12 Red Slider**

##### **Red slider**

Use the RGB/HSV-slider controls to edit any color in the palette.

##### **4.12.12.1 Red edit box**

###### **Red edit box**

Shows red/hue value of selected color index.

#### **4.12.13 Green Slider**

##### **Green slider**

Use the RGB/HSV-slider controls to edit any color in the palette.

##### **4.12.13.1 Green edit box**

###### **Green edit box**

Shows green/saturation value of selected color index.

#### 4.12.14 Blue Slider

##### Blue slider

Use the RGB/HSV-slider controls to edit any color in the palette.

##### 4.12.14.1 Blue edit box

##### Blue edit box

Shows blue/value magnitude of selected color index.

#### 4.12.15 Smooth Button

##### Smooth palette button

Use to create a random palette with smooth color spreads.

#### 4.12.16 Scramble

##### Scramble

Use to create a palette with random color indexes.

### 4.13 Edit Text command

#### Text (Edit menu)

Allows you to edit text and font and apply it to a drawing. Select the font button to set the font style, size and color. In the text window click on Okay to add a line of text to the current image. (You can add multiple lines of text too, up to 80 characters.) The cursor will change to a crosshair. Position the cursor where you want the text to start and left-click the mouse. Note: font and title text are saved in the file "prefs.txt" in Fractal Imaginator's startup directory. Title text can also be edited (as a single line only) in the Edit/Formula window.

### 4.14 Preferences

#### Preferences (Edit menu)

Each time you use the Reset command, Fractal Imaginator restores data variables to built-in defaults. The Set Defaults button allows you to change some of the data variable defaults to whatever the current settings are. Some of the customizable variables include step, fine, formula, viewpoint, lighting, rotational angles, Phong and x/y space. The new Reset defaults are saved in the file "prefs.txt" when you close the program (if the Defaults check box is selected.) The check boxes in the group "Save on Program Close" allow you to change the default startup mode of a few Auto options, such as Auto Redraw, and the Random Setup variables. By keeping the boxes selected, Fractal Imaginator saves the last changes you make to these options. If you want to go back to the initial settings (the way Fractal Imaginator was packaged originally) you can click on the Reset Defaults button. This restores the data, Auto variables and random setup defaults.

## 5 Image menu

### Image menu commands

The Image menu offers the following commands:

<a href="#">Draw</a>	Draw the picture.
<a href="#">Plot To File</a>	Plot large bitmap images directly to png file.
<a href="#">Auto Redraw</a>	Redraw image on command.
<a href="#">Auto Clear</a>	Clear drawing area before new plot.
<a href="#">Auto Sound Alert</a>	Enable or turn off sound alerts.
<a href="#">Auto Remote</a>	Open remote automatically at startup.
<a href="#">Auto Time</a>	Show time used to plot image.
<a href="#">Abort</a>	Abort drawing.
<a href="#">Continue</a>	Continue drawing.
<a href="#">Zoom</a>	Zoom into rectangle.
<a href="#">New View on Zoom</a>	New view on zoom.
<a href="#">Clone</a>	Clone current view.
<a href="#">Dive</a>	Peel off outer layer of quaternion.
<a href="#">Full Screen</a>	View image full-screen.
<a href="#">Pilot</a>	Use Pilot to rotate and alter key cubic variables.
<a href="#">Reset (Ranges or Figure)</a>	Reset coordinates or figure parameters
<a href="#">Calculate Fractal Dimension (2D)</a>	Calculate fractal dimension of 2D image using "box method"

### 5.1 Image 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. Use the Continue command to restart plotting from the current column. When the lsystem option is selected, a custom [window](#) is opened to enter lsystem draw options.

#### 5.1.1 Image Draw Lsystem command

##### Draw (Lsystem) command (Image menu)

The lsystem draw window has edit boxes to set projection angles, roll pitch and yaw. The angles have a range of -360 to 360 degrees. The initial turtle heading is 0,1,0 -- which moves the turtle up in the direction of the y-axis when the projection angles are all 0. The view may be improved when some of the projection angles are non-zero, or to rotate an object done with LParser to an appropriate viewing angle.

The skewing angles affect the line styles when the line width is greater than 1. These can warp the lines(tubular or poly) to a non-uniform width.

The light point is similar to that used when Fractal Imaginator draws 3D quaternions.

The Lighting variables shininess, highlight, gamma and ambient are used to adjust ambient light and Phong 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 correction increases or decreases the intensity of the light source's illumination.

Note: once a plot has been drawn, the lighting variables and light point can be changed without redrawing the lsystem. Click the light bulb button to redisplay a picture after changing the lighting variables or light point.

You enter the level of recursion into the level box. The maximum level of recursion is limited to 50.

Use the scale slider to set the size of the drawing, .05 to 1.0. Each lsystem is pre-scaled prior to drawing, to approximate the size of the draw window. To speed up the pre-scaling, the width of the line drawn isn't used to map image limits. So it's possible an object may extend off the window's edges, when a line width is greater than 1 and the scale is 1. In this case, use a scale less than 1 to "shrink fit" the drawing. You also have the option of turning off auto-scaling. It is useful to uncheck the Auto Scale box when rotating an lsystem in an animation. Without auto-scaling, the origin of the lsystem is fixed to a single point in the draw window [xpos,ypos], so that rotations turn smoothly around that point. Note: you may have to manually set the x and y position variables so that the entire image fits within the draw window. The default values for xpos and ypos place the origin of the lsystem at the center of the screen.

Click on Draw to recompute and draw the current axiom.

Select Wire Frame to draw a wire frame model of the current axiom/rules. This is useful to speed up drawing while prototyping an lsystem.

Enter additional Steps (default 1), to increase the resolution of solid rendering. Each additional step increases rendering time proportionally, but can increase the smoothness of certain area fills, as in polygon rendering of large objects.

Enter a non-zero mutation value to mutate the current lsystem. This follows the same mutation rules as LParser. Small values (less than 5) can produce extremely lengthy mutations. Use Fixed Mutation to limit the size of the mutated lsystem to the original size. In this case the no rules are appended or inserted. Only angle and movement commands are switched.

Select the extended set box to enable the extended rule set. Most input files made for LParser will work with the extended set enabled, but the extended rules may not work if the command is overridden by a production rule.

## 5.2 Plot to file

### Plot to File

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 group sets the bitmap resolution (width 800 to 14400. Drawing aspect is that of the current image.) Click on Okay to set the target file name and start a new plot to file. Note: the 3200X2400 bitmap size is suitable for 8 1/2X11 printouts at 320-720 dpi. The larger bitmap sizes are suitable for poster-size printouts. Solid-guessing is disabled when using this option.

## 5.3 Image Redraw command

### Auto Redraw command (Image menu)

With this command disabled (on by default), redraw does not occur except when the Draw command is executed, or Continue. Most of the time you want to see the results of changing a parameter or mapping option, so redraw occurs automatically with parameter or mapping changes. Sometimes you want to change more than one parameter before redrawing the image. So you need to turn this option off then.

## 5.4 Image Auto Clear command

### Auto Clear command (Image menu)

With this command enabled (on by default), the drawing area is cleared before starting a new plot. You can turn off this option when you want to see the effect of minor changes to parameters, as they affect the plot pixel by pixel. You can use the shift-c command ([hot keys](#)) to clear the drawing area at any time.

## 5.5 Image Auto Alert command

### Auto Sound Alert command (Image menu)

With this command enabled (on by default), the user is notified by a sound clip when a drawing is completed or user-canceled. By disabling this command the completion exclamation is suppressed and also any alert that contains a message box. Note: some sound clips are automatically generated by Windows, or there is no text alert for a given error condition. In these cases the sound alert is unaffected by the Auto Alert command.

## 5.6 Image Auto Remote command

### Auto Remote command (Image menu)

With this command enabled (on by default), the remote is opened immediately at program startup. Handy if you find the remote useful and don't want to click on the toolbar button each time the program starts up.



## 5.7 Image Auto Time command

### Auto Time command (Image menu)

With this command enabled (on by default), the time that an image takes to plot is displayed when the plot is complete. Fractal Imaginator saves the condition of this option at session's end, so if you disable it and close the program, the option will be disabled when you restart Fractal Imaginator.

## 5.8 Image Abort command

### Abort command (Image menu)

Use this command to stop drawing. Clicking inside a window's drawing area or close box (or the program close box) will also stop the drawing. Note: once a plot has started Fractal Imaginator continues to draw the image for that window regardless of which drawing window has the input focus, until done or aborted. You can open and close other drawing windows without affecting the current drawing, but only one drawing is active at any time.

## 5.9 Continue Draw

### Continue Draw (Image menu)

Continues a plot that was aborted early. The plot is restarted at the beginning of the last row drawn. Continue is disabled with any Growth menu option.

## 5.10 Zoom

### Zoom (Image menu)

Turns on zoom mode, so that detail of the current plot may be magnified. 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 to shrink the box or the right button to enlarge it. Use the left and right arrow keys to rotate the box counter-clockwise or clockwise. Use the up and down arrow keys to squash or expand the box, changing the aspect of the image. You start a zoom by pressing the space bar. You abort a zoom by clicking outside the main window or in the title bar, or by pressing the escape key. The program will begin a new plot at the new coordinates. 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. The outer zoom limits are between -1000 and 1000. The precision is that of double precision (64 bits)

Rotating a quaternion while zooming is possible, but inexact. Depending on the rotational angles set in the quaternion window, you may end up rotating on the z-axis, which may rapidly result in overshooting the area of zoom. So avoid large rotations combined with high levels of zoom. It is better to rotate at low zoom levels, and then zoom without rotating, for close-ups. If you change screen resolutions, you must redraw the bitmap image for a function before you can accurately zoom on it.

## 5.11 Image New View on Zoom command

### New view on zoom (Image menu)

With this option enabled, a new window is opened with each zoom, instead of the zoom box area replacing the original image. Ignored in avi mode.

## 5.12 Image Clone

### Clone (Image menu)

A new draw window is opened that contains the same fractal data as the window it was opened from. This is useful for comparing minor changes in texturing options, etc.

## 5.13 Dive

### Dive (Image menu)

Select Dive to go beneath the surface of a quaternion. Some quaternions have a smooth border that doesn't show the turbulence below the surface. Using the Dive option strips off the border layer to reveal what's underneath.

## 5.14 Full Screen

### Full Screen (Image menu)

Displays the entire plot, expanding or shrinking the image to fit in a maximized window without title bar, scroll bars or menu bar. At all other times, part of the picture is hidden by the inclusion of the title bar, toolbar, scroll bars and menu bar. To exit full-screen mode, press any key or click the left-mouse button.

## 5.15 Pilot

### Pilot (Image menu)

Opens the Pilot window to adjust key parameters, rotate, zoom and redraw the figure interactively. The current image is reduced to one quarter normal for faster redraw. Each click on a Pilot button increments or decrements a parameter. The Speed slider controls the rate at which the buttons operate (default is 10.)

Press the space bar or Click on Ok to open a new window and draw the altered image full-size. Press Esc or click on Cancel to exit this mode without opening a new window. Note: when using this option while an AVI stream is open, a new window isn't opened, but the altered figure is drawn in the current draw window, the changed parameters replacing the previous ones.

## 5.16 Reset

### Reset (Ranges or Figure)

The Reset Ranges command resets the real Z and imaginary Z ranges in the Parameters window (to +/-2.0 and +/-1.5.) No other menus or variables are affected. This is useful in conjunction with the "P" command to generate and view Julia sets. After setting the complex-C variable via shift-P (Caps Lock off), you need to reset the Z ranges to see the entire Julia set after zooming into a Mandelbrot set.

The Reset Figure command resets the parameters in the active figure to generate a basic Mandelbrot set for  $z^2+c$ .

## 5.17 Calculate Fractal Dimension (2D)

### Calculate Fractal Dimension (2D)

This routine calculates the fractal dimension of a 2D object using the Barnsley's "box" method. The formula is  $\log(N)/\log(m)$ , where  $N = 2$  and 'm' is the magnification factor. For maximum accuracy, the background color should be composed entirely of color index #1 (no orbits), and the object should not contain any pixels colored with the background color. The fractal dimension for Mandelbrot and Julia sets will vary with the size of the bitmap, since in a larger bitmap more "cusps" are visible.

## 6 Type menu

### Type menu commands

The Type menu offers the following commands:

<a href="#">Mandelbrot</a>	Mandelbrot set.
<a href="#">Julia</a>	Julia set.
<a href="#">Quaternion</a>	Set fractal type to quaternion.
<a href="#">Hypernion</a>	Set fractal type to hypercomplex quaternion.
<a href="#">Cubic Mandelbrot</a>	Set fractal type to cubic Mandelbrot.
<a href="#">Complexified Quaternion</a>	Set fractal type to complexified quaternion.
<a href="#">2D Hypercomplex Map</a>	Set fractal type to two-dimensional hypercomplex mapping.
<a href="#">2D Quaternion Map</a>	Set fractal type to two-dimensional quaternion mapping.
<a href="#">2D Complexified Map</a>	Set fractal type to two-dimensional complexified quaternion mapping.
<a href="#">2D Cubic Map</a>	Set fractal type to two-dimensional cubic Mandelbrot mapping.
<a href="#">Lsystem</a>	Set fractal type to lsystem.

## 6.1 Mandelbrot

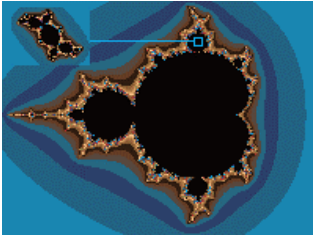
### Mandelbrot

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

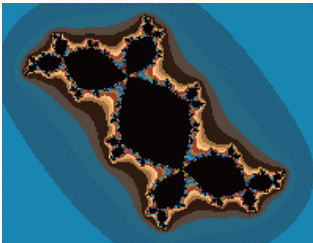
## 6.2 Julia

### Julia

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. This option, without the Bound flag set, generates the so-called 'filled-in' Julia set, which includes non-escaping points as well as the Julia set.



Julia from Mandelbrot



Julia set

## 6.3 Type Quaternion command

### Quaternion (Type menu)

Use this command to set the fractal type to a 3D quaternion. Variables that affect this fractal type are defined in the Edit/Quaternion window.

## 6.4 Type Hypernion command

### Hypernion (Type menu)

Use this command to set the fractal type to a hypercomplex 3D quaternion. Variables that affect this fractal type are defined in the Edit/Quaternion window. A hypernion uses hypercomplex math to shape the 3D object, which usually results in a squared-off shape,

rather than the rounded shape of the typical quaternion.

## 6.5 Type Cubic command

Use this command to set the fractal type to a cubic Mandelbrot. Variables that affect this fractal type are defined in the Edit/Quaternion window. Built-in formulas that support this type are G0-G9. Fun#1 must be set to one of these formulas to enable this type.

Cubic Mandelbrots quaternions use the S and Si variables to set the initial value and range of the 3rd dimension. Starting from an initial value of 'S', Si is normally set to  $2*\text{abs}(S)$ , when the Type is Mandelbrot. Si can also be set to center the 3rd dimension on something besides 0.0. Si has no affect in 2D hypercomplex mode. (In 2D mode, the S variable acts as one of the two fixed dimensions, along with the 4th Center variable.)

The 4th Center points to the center of the fourth dimension (with quaternions/hypernions the 4th Dimension variable is used.)

In addition, the Arg value (in the Formula 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.

## 6.6 Type Complexified Quaternion command

### Complexified Quaternion (Type menu)

Use this command to set the fractal type to a 3D complexified quaternion. Variables that affect this fractal type are defined in the Edit/Quaternion window. A compquat uses another variation of quad math to shape the 3D 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.

## 6.7 Type 2D Quaternion Map command

### 2D Quaternion Map (Type menu)

This command sets the fractal type to a 2D quaternion mapping.

## 6.8 Type 2D Hypercomplex Map command

### 2D Hypercomplex Map (Type menu)

This command sets the fractal type to a 2D hypercomplex mapping.

## 6.9 Type 2D Cubic Map command

### 2D Cubic Map (Type menu)

This command sets the fractal type to a 2D cubic Mandelbrot mapping. Only enabled with the built-in cubic formulas (G0-G0.)

## 6.10 Type 2D Complexified Map command

### 2D Complexified Map (Type menu)

This command sets the fractal type to a 2D complexified quaternion mapping.

## 6.11 Type Lsystem command

### Lsystem (Type menu)

Use this command to set the fractal type to the lsystem-type fractal instead of the default Mandelbrot/Julia contour-type fractals. The Edit/Lsystem Axiom and Edit/Lsystem Rules commands are used to enter the lsystem. Fractal Imaginator's lsystem supports 3D drawing, and all plots use a z-buffer for hidden-line removal and ray-tracing. For an introduction to lsystems and complete list of rules recognized by the lsystem parser, see [Lsystem Info](#). Most options for the contour plots don't apply to lsystem plots. Clicking inside the draw window

with the left-mouse button stops all plotting. No Continue option is available for lsystems, though a redraw option is offered to cut down on pre-scaling time.

## 7 Break menu

### Break menu commands

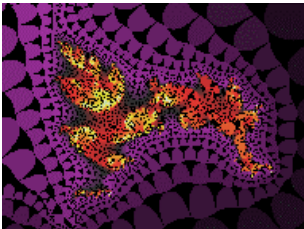
The Break menu offers the following commands:

<a href="#">Biomorph</a>	Escape on any part of z.
<a href="#">Biomorph Off</a>	Reset biomorph flag.
<a href="#">Orbit Traps</a>	Set orbit trapping method.

### 7.1 Biomorph

#### Biomorph

Biomorphs test the real  $Z$  and imaginary  $Z$  values after breaking the iteration loop. If the absolute value of either is less than the preset  $zlimit$ , the point is mapped as part of the set. This method produces biological-like structures in the complex plane. Normally the biomorph tendrils are colored in the set color (the color reserved for non-divergent or inner points.) With the Set Only flag on, the tendrils are colored according to the color-scaling option used (other external points are colored in the background color.) A window is opened each time this option is selected to set a color for the area that falls within the biomorph trap. This can be 0-235.



### 7.2 Biomorph Off

#### Biomorph Off

Turns off the biomorph flag. Alternatively you can enter -1 in the Biomorph window to turn off the bio-flag.

### 7.3 Orbit traps

#### Orbit Traps

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

The Epsilon-Cross method colors points only if the absolute value of  $Z$ -real or  $Z$ -imaginary is less than or equal to Epsilon (a small value.) Other points are mapped at the time

they blow up (exceed the zlimit.) This produces hair-like structures that branch wildly from the complex set boundaries.

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.

Epsilon2 is used to create windows into the stalks. The default value is 0.0, which produces solid stalks.

The Display Even Only option is used to un-clutter some epsilon plots by coloring points that escape on even iterations only. Odd points are plotted in the background color.

A window is opened to enter a value for Epsilon and Epsilon2, which are used to define the size of the trap areas (.001-2.0 and 0.0-epsilon.) The exclude box is used to exclude the first # iterations (0-99) from orbit trapping.

To produce the maximum 3-D effects (as Phil Pickard and Paul Carlson do) with these options, Level Curve #4 must be set, and the Cutoff value (in the Parameters window) should equal the negation of the epsilon value (-epsilon.) You'll need to set up a special palette with a number of color ranges that matches the split-palette number if set. Built-in examples d3-d6 illustrate how to set up 3d-like fractals.

To automate the process of producing Paul Carlson's 3-D like fractals, a check box has been added to this window for 'Carlson extensions'. This sets the Background and Set Only flags as well as the Level Curve #4 and the cutoff value, and sets the Map to  $\langle \text{Abs}(Z\text{-Real}) \text{ or } \text{Abs}(Z\text{-Imag}) \rangle$ . An exclude value of 2 is also used. The Map and exclude values are extra parameters that Paul uses in some of his formulas, and may be omitted in other cases. This is easily done by first enabling the Carlson extensions by checking the box and clicking on Okay, then opening the window again and un-checking the box and changing the appropriate variables/flags.

### 7.3.1 Orbit trap values

#### Orbit Trap Values

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 Apply to apply changes without closing the window. Click on Okay to close the window and apply changes, if any. Click on Cancel to exit the window without changing parameters.



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.

Check Carlson extensions to apply additional options that are used in Paul Carlson's 3D-like orbit-trap methods.

## 8 Render menu

### Render menu commands

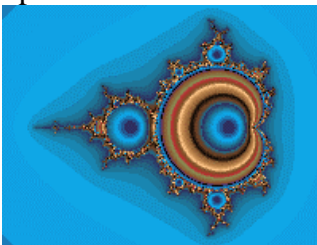
The Render menu offers the following commands:

<a href="#">Level Curve-&gt;</a>	Set level curve or reset level curve flag.
<a href="#">Decomposition-&gt;</a>	Binary or continuous decomposition.
<a href="#">Decomposition Off</a>	Reset decomposition flag.
<a href="#">Coloring Filter</a>	Define coloring filter.
<a href="#">Atan Coloring</a>	Use Atan algorithm for coloring.
<a href="#">Bof60 Coloring</a>	Use Bof60 algorithm for coloring.
<a href="#">Potential Coloring</a>	Color by magnitude of z.
<a href="#">Texture Scale</a>	Set scaling factor for texture.

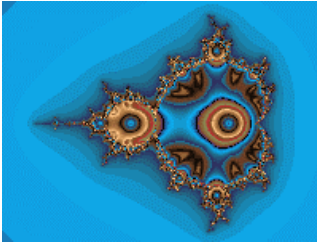
### 8.1 Level Curve

#### Level Curve

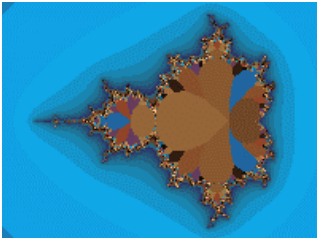
Level-curves map the set points based on how small the value of Z gets. This allows the inside of the complex set to be color-scaled. Log Map #1 produces colored bands on the inside of the complex set. Points are mapped according to what the value of z is at final iteration. Small Log #2 and Linear Map #5 produce circular patterns inside the complex set. Points are mapped according to the smallest value z gets during iteration. Indexed Log #3 and Indexed Linear #6 are mapped according to the time it takes z to reach its smallest value. Level curves 2,3(and 5,6) are described more fully in *The Beauty of Fractals*. Linear Map #4 is mapped like Log Map #1 (with the mapped value of the function at its final iteration applied to the color palette) and produces 3D-like effects with the Epsilon-Cross method. The Log methods use a log palette, while method #4-6 use linear palettes. This option can override (or may be overridden by) many of the options in the Color-Scaling menu. Decomposition doesn't use Level Curve shading, unless you select the Use Level Curve option.



Log Map #1



Small Log #2



Indexed Log #3

Bubble #7 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. Color-mapping should be set to Use Level Curve. This is a trial and error method that uses two other variables to produce the final effect, magnify and cutoff, as entered in the Parameters window. Magnify is used to screen unwanted background contours in the plot, while cutoff is used to fill out the color palette. Magnify should be a low value, usually less than .1, to eliminate the contours that usually appear in escape-type Mandelbrot/Julia sets. If it is too large the bubbles will be too crowded, while too small a value will cause the bubbles to disappear. Cutoff needs to be a small negative value, usually equal to the magnify value times the number of color splits. E.G., for a magnify value of .1 you should use a cutoff value of -.8 for a divide-by-eight palette. An incorrect cutoff 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 #6. The color ranges should be graded from light to dark to highlight the bubble centers.

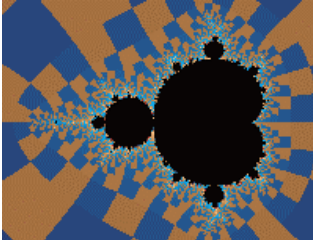
## 8.2 Decomposition

### Decomposition

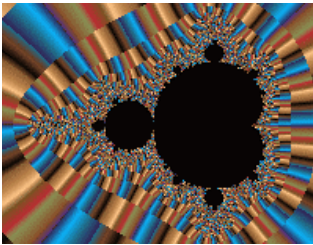
When a Decomposition flag is set, you have the option of performing either a binary or continuous decomposition. Toggle the External/Internal option for either an external or internal decomposition. The Angle-Only option excludes the Coloring-scaling options from consideration when plotting points derived from decomposition. It also excludes escape times in connection with the Angle-Iteration option on the Color-Scaling menu. An external decomposition decomposes points that are outside the complex set. An internal decomposition decomposes the complex set. For Mandelbrot/Julia curves,  $z$ -arg is broken into two parts for a binary decomposition. For Newton/Renormalization curves, the binary decomposition is also related to the number of solutions a formula has, if it supports mapping option 1. Continuous decomposition breaks  $z$ -arg into  $n$  parts, where  $n$ =angles (2-256), as set in the Continuous Decomposition window.

Note: With the graded-palette option checked, the decomposition option is extended for extra smoothness in Fractal Imaginator. The number of angles is internally multiplied by 236 to track the decomposition angle more closely.

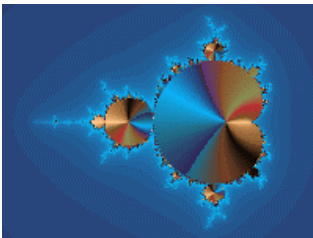
(Consult *The Beauty of Fractals* by Peitgen & Richter for a mathematical explanation of decomposition.) When Biomorph or Epsilon is decomposed, the tendrils or hairs are decomposed as external points. Use the Set Only flag to emphasize the tendrils and hairs when external decomposition is used.)



Binary Decomposition



Continuous External



Continuous Internal

### 8.3 Decomposition Off

#### Decomposition Off

Turns off all decomposition flags and resets the Internal/Eternal option to Eternal.

### 8.4 Coloring Filter

#### Coloring Filter

Here you define an hsv 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  $z$  components  $x$  and  $y$ .  $X$  and  $y$  are the real and imaginary parts of the last  $z$  value in the iteration loop. Sample function:  $\sin(x+y)+\cos(x*x)$ . The Magnify slider is used to control

the intensity of the filter (in conjunction with the Magnify variable in the HSV filter window.) Use the Preview button to see what the filter looks like with x and y ranges of  $-2\pi$  to  $2\pi$ .

Note: the coloring filter can only be used with quaternions. 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 ( $\pm 100.0$  for the exponential,  $\pm 200.0$  for hyperbolic functions,  $\pm 1.0$  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.

## 8.5 Atan Coloring

### Atan Coloring

Uses an Atan algorithm by David Makin to color a quaternion image.

## 8.6 Bof60 Coloring

### Bof60 Coloring

A variation of the Bof60 algorithm found in the classic Pietgen/Richter text, *The Beauty of*

Fractals, adapted by David Makin to color a quaternion image.

## 8.7 Potential Coloring

### Potential Coloring

The magnitude of  $z$  (at the quaternion border) is used to color the (quaternion) image.

## 8.8 Texture Scale

### Texture Scale

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, Bof60 and Potential coloring options.

## 9 Pixel menu

### Pixel menu commands

The Pixel menu offers the following commands:

<a href="#">Phoenix</a>	Phoenix Curve.
<a href="#">Four-Fold Symmetry</a>	XY symmetry.
<a href="#">Solid-Guessing</a>	Solid-guessing plotting mode.

## 9.1 Phoenix

### Phoenix

The Phoenix flag rotates the planes, so that the imaginary plane is mapped horizontally and the real plane is mapped vertically.



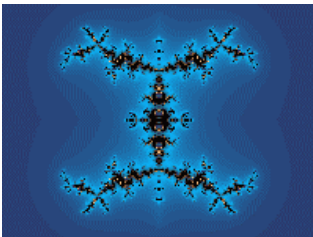
This option is normally used for mapping Phoenix curves (Shigehiro Ushiki), which are Julia-related curves based on the formula  $f(z+1)=z^2+p+qz$ . 'p' and 'q' are constants, and the 'z' term of 'qz' is actually the value of  $z^{n-1}$ , or the previous value of  $z$  before the current iteration. 'zn' is reserved by Fractal Imaginator to represent this value, while the complex constant set in the Parameters window becomes 'p' and 'q'. The real part of the complex constant is 'p' and the imaginary part of the constant is 'q' (when the Phoenix option is chosen).

If the Phoenix flag is used with the Mandelbrot option, 'j' and 'k' should be used as the constants, since the complex constants p and q are already used as the starting value of 'z0'.

## 9.2 Symmetry

### Four-Fold Symmetry

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



## 9.3 Solid Guessing

### Solid Guessing

In the solid-guessing plotting mode, the program guesses at colors that lie inside rectangular areas of the plot. It first computes all the perimeter pixels of a rectangle, and checks if all the pixels have the same color. If so, all the pixels inside the rectangle are colored the same and no further calculations are done on that rectangle. Otherwise the rectangle is broken into four parts and the above procedure is repeated for each part. If any of the perimeter pixels are different at this point, all the remaining pixels in the smaller rectangle are computed. The screen is updated in groups of 16 lines.

This method can be much faster than the default single-pass mode that Fractal Imaginator starts in. This is especially true for plots that have large areas of a single color. For very intricate plots that have little open space, the solid-guessing mode can still be 15-20% faster.

The solid-guessing mode can fail for some plots that have small areas of one color that are islands inside the rectangular areas under test.

Solid-guessing is not available for lsystems, orbital fractals, 3D height fields or midpoint displacement.

## 10 Color menu

### Color menu commands

The Color menu offers the following commands:

<a href="#">Cycle</a>	Cycle colors.
<a href="#">Color-Scaling menu</a>	
<a href="#">Palette menu</a>	
<a href="#">Divide By One Palette</a>	No split palette.
<a href="#">Divide By Two Palette</a>	Split palette into two sections.
<a href="#">Divide By Four Palette</a>	Split palette into four sections.
<a href="#">Divide By Eight Palette</a>	Split palette into eight sections.

## 10.1 Color Cycle command

### Cycle command (Color menu)

Use this command to cycle colors when not plotting. Undoing an action disables the cycle command until the image is redrawn.

## 10.2 Color-Scaling menu

### Color-Scaling menu commands

The Color-Scaling menu (in Color menu) offers the following commands:

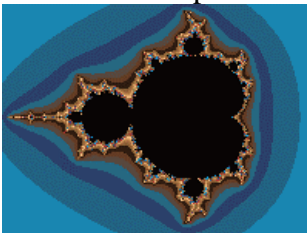
<a href="#">Escape-&gt;</a>	Escape-time color scaling.
<a href="#">Level</a>	Color scaling based on $\log(z)$ .
<a href="#">Continuous Potential</a>	Color scaling based on continuous potential.
<a href="#">Use Level</a>	Use level curve option for all coloring.
<a href="#">Background</a>	Set external points to background color.
<a href="#">Set Only</a>	Plot points in complex set only.
<a href="#">Graded Palette-&gt;</a>	Use non-repeating vs modulus palette.

### 10.2.1 Escape

#### Escape

Five options are included that color a point based on its escape time (when it blows up.)

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



Escape-time coloring.

The Iteration+ option uses the sum of a point's escape time and the value chosen (which can be picked from a menu that mirrors the Map menu.) A window is opened to set a q factor (1-200), which scales the sum value.

The Iteration\* option uses the product of a point's escape time and the value chosen. ) A window is opened to set a q factor (1-200), which scales the product value.

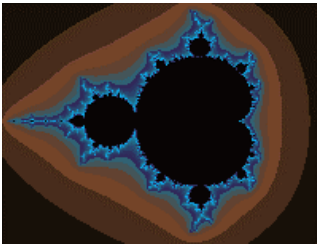
The Angle option use the absolute value of a point's exit angle (theta.) This is the atan method in Fractint.

The Angle-Iteration option uses the angle formed by the difference between a point's last two exit values and subtracts the point's escape time. Using the Angle-Only option on the Decomposition menu, escape times are not subtracted from the difference angle. This is Paul Carlson's atan method.

### 10.2.2 Level

#### Level

A point is colored based on its logarithmic escape. A window is opened to set a q factor, which controls the smoothness of picture color. A higher q factor results in grainier pictures and excess detail. Too low a q factor results in loss of colors and detail.

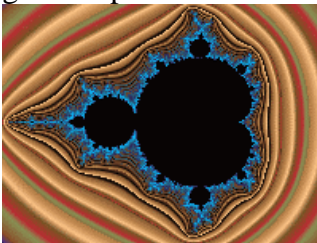


Level coloring.

### 10.2.3 Continuous Potential

#### Continuous Potential

A point is colored based on its continuous potential (when it blows up.) A window is opened to set a q factor, which controls the smoothness of picture color. A higher q factor results in grainier pictures and excess detail. Too low a q factor results in loss of colors and detail.



Continuous-Potential coloring.



## 10.2.4 Use Level Curve

### Use Level Curve

All points are colored according to the choice selected from the Level-Curve Flag option. Defaults to Linear Map #4 if no Level Curve is checked. This works with Decomposition and other methods that would normally not use Level-Curve shading.

## 10.2.5 Set Only

### Set Only

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

## 10.2.6 Background

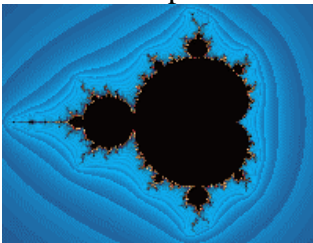
### Background

An external point is colored with the background color. This works like the Set Only flag, except with decomposition plots and Biomorph/Epsilon plots. Normally, when a point is decomposed, its escape time or level color is added to its arg (exit angle) to determine its final coloring. With Background color-scaling, only a point's arg determines its color. With Biomorph/Epsilon plots, all external points are colored with the background color and all Biomorph/Epsilon points are colored with the set color.

## 10.2.7 Graded Palette

### Graded Palette

All points are colored with a non-repeating graded palette versus the default repeating(modulus) color scaling. Has no effect on the Background or Use Level Curve options, or when you use the cutoff value in the Parameters window as a color multiplier. When used with the Escape/Iteration coloring mode and a negative cutoff value, iterations are interpolated to reduce banding in escape-time pictures. Three options are provided for smoothing. The Interpolated version works for most escape-time formulas except convergent types (Newton and renormalization.) The Mandelbrot version is based on Linas Vepstas' log log algorithm, and is designed mainly for formulas that use  $z^2$  as their main focus, such as  $z^2+c$ . The Exponential smoothing method is based on Ron Barnett's algorithm, and works for both escape time and convergent-type fractals.



Level-graded coloring.

## 10.3 Palette menu

### Palette menu commands

The Palette menu (in Color menu) offers the following commands:

Palette #1-21      Use one of 21 palettes.

### 10.3.1 Palette 1-21 command

#### Palette command (Palette menu)

Switch to palette #. Used with palette-coloring mode.

## 10.4 Color Divpal1 command

### Divide by One Palette (Color menu)

Palette is not split before applying to pixel.

## 10.5 Color Divpal2 command

### Divide by Two Palette (Color menu)

Palette is split into two parts before applying to pixel.

## 10.6 Color Divpal4 command

### Divide by Four Palette (Color menu)

Palette is split into four parts before applying to pixel.

## 10.7 Color Divpal8 command

### Divide by Eight Palette (Color menu)

Palette is split into eight parts before applying to pixel.

## 11 View menu

### View menu commands

The View menu offers the following commands:

[Toolbar](#) Shows or hides the toolbar.  
[Status Bar](#) Shows or hides the status bar.

## 11.1 View Toolbar command

### Toolbar command (View menu)

Use this command to display and hide the Toolbar, which includes buttons for some of the most common commands in Fractal Imaginator, such as File Open. A check mark appears next to the menu item when the Toolbar is displayed.

See [Toolbar](#) for help on using the toolbar.

### 11.1.1 toolbar

#### Toolbar



The toolbar is displayed across the top of the application window, below the menu bar. The toolbar provides quick mouse access to many tools used in Fractal Imaginator,

To hide or display the Toolbar, choose Toolbar from the View menu (ALT, V, T).

Click	To
	Open the remote which contains shortcut buttons for many common tasks and options in Fractal Imaginator
	Open an existing drawing. Fractal Imaginator displays the Open dialog box, in which you can locate and open the desired file.
	Save the active drawing or template with a new name. Fractal Imaginator displays the Save As dialog box.
	Draw Mandelbrot set
	Draw Julia set
	Zoom into rectangle.
	Set image size.
	Edit palette.
	Edit formula/type data.
	Edit fractal parameters.
	Draw image from current parameters.
	Continue drawing.
	Reset coordinates.
	Show picture full-screen.
	Display info about Fractal Imaginator.
	Display Fractal Imaginator's help index.

## 11.2 View Status Bar Command

### Status Bar command (View menu)

Use this command to display and hide the Status Bar, which describes the action to be executed by the selected menu item or depressed toolbar button, and keyboard latch state. A check mark appears next to the menu item when the Status Bar is displayed.

See [Status Bar](#) for help on using the status bar.

### 11.2.1 status bar

#### Status Bar



The status bar is displayed at the bottom of the Fractal Imaginator window. To display or hide the status bar, use the Status Bar command in the View menu.

The left area of the status bar describes actions of menu items as you use the arrow keys to navigate through menus. This area similarly shows messages that describe the actions of toolbar buttons as you depress them, before releasing them. If after viewing the description of the toolbar button command you wish not to execute the command, then release the mouse button while the pointer is off the toolbar button.

The right areas of the status bar indicate which of the following keys are latched down:

Indicator	Description
CAP	The Caps Lock key is latched down.
NUM	The Num Lock key is latched down.
SCRL	The Scroll Lock key is latched down.

## 12 Window menu

### Window menu commands

The Window menu offers the following commands, which enable you to arrange multiple images in the application window:

<a href="#">Cascade</a>	Arranges windows in an overlapped fashion.
<a href="#">Tile</a>	Arranges windows in non-overlapped tiles.
<a href="#">Arrange Icons</a>	Arranges icons of closed windows.
<a href="#">Size Desktop</a>	Size drawing area to window frame.
<a href="#">Window 1, 2, ...</a>	Goes to specified window.

## 12.1 Cascade

### Cascade command (Window menu)

Use this command to arrange multiple opened windows in an overlapped fashion.

## 12.2 Tile

### Tile command (Window menu)

Use this command to arrange multiple opened windows in a non-overlapped fashion.

## 12.3 Arrange Icons

### Window Arrange Icons Command

Use this command to arrange the icons for minimized windows at the bottom of the main window. If there is an open drawing window at the bottom of the main window, then some or all of the icons may not be visible because they will be underneath this drawing window.

## 12.4 Size DeskTop

### Window Size DeskTop Command

Use this command to size the active drawing window to its frame size. Use after Tile command to reduce white space around a drawing that is smaller than screen size.

## 12.5 1, 2, ...

### 1, 2, ... command (Window menu)

Fractal Imaginator displays a list of currently open drawing windows at the bottom of the Window menu. A check mark appears in front of the drawing name of the active window. Choose a drawing from this list to make its window active.

## 13 A/V menu

### A/V menu commands

The A/V menu offers the following commands:

<a href="#">Open AVI Stream</a>	Open AVI file for writing and draw initial frame.
<a href="#">Write Frames</a>	Write frames to AVI file.
<a href="#">Close AVI Stream</a>	Close an existing AVI stream.
<a href="#">View AVI</a>	View an AVI animation file.

[AVI Object](#)

Output video frames as obj file.

[AVI WRL](#)

Output video frames as wrl files.

## 13.1 Open Avi Stream

### Open Avi Stream...

Through a series of windows, this allows you to name and open an avi animation stream and choose a compression method. After using the file requester to name the file, you are given a choice of compression methods. The compression methods include Intel Indeo Video®, Microsoft Video 1 and Cinepak Codec by Radius. (All compression methods degrade the original images, some more than others.) The first key frame in the stream is then drawn and written to the file.

Notes: after the stream is opened, the size of the fractal that can be drawn is fixed at the size of the frame. No changes can be made to the size until the stream is closed. New: If you open a video stream after setting up a batch mode (Demo menu), then the frames will be written as a series of bmp, obj or wrl files, depending on whether AVI Object or AVI WRL is also checked.

## 13.2 Write Frames

### Write Frames...

With this option, frames are written to a stream based on the difference between the current key frame and the previous key frame. The first key frame is written when you open a stream. The next key frame is created each time you use this option. In between you can zoom or change Avi variables as much as necessary. The stream is only written to when this option is used. The last key frame is automatically saved after the 'tween' series is written. The number of frames may range from 1-1500 frames between keys. With a frame number of 1 only the key frames are written. This allows animation to be created that incorporates all scalable variables in Fractal Imaginator.

Use the Cancel button to exit this dialog without initializing a new series of frames.

Check the Log Scaling box if you want the frames to be written with logarithmic space between frames, else linear space is used. Useful when zooming, where frames would otherwise be packed together at the end of the frame series.

Notes: key frames are saved in parameter files (.fim), with filenames of "bvf\_image#\_title.fim", where '#' is the number of the keyframe and 'title' is the name of the working fractal file.

## 13.3 Close Avi Stream

### Close Avi Stream

Closes any open avi stream file. You need to do this before viewing the file or creating a new avi file. The stream is also closed when you exit Fractal Imaginator.

## 13.4 View Avi

### View Avi...

Opens an avi file for viewing. You can preview any multimedia file by clicking on its file name. A multimedia box will appear to the right of the file list. Click on okay to open the main view window.

There are buttons to Play a file forwards or Backwards, or forward automatically with Auto rewind/repeat. Click on Slow to slow down a video. Each click on Slow halves the viewing speed. A click on Stop freezes viewing and restores the view speed to normal playback.

Use the Open button to view a different avi file. Use the Save button to save the file in a different compression format. You must use a different name to save the file than the name that was used to open it. Click on the left-mouse button or any key to abort a save operation.

Note: the view avi requester can be used to preview any multimedia file, including midi files.

## 13.5 AVI Object

### AVI Object

When this flag is set, Fractal Imaginator generates single frames in obj format instead of opening a video stream. The 3D object files can be exported into a program such as Bryce, for post-processing into videos. This works in conjunction with the Demo/Batch mode command, to set the target disk directory and file name, so is only enabled when Batch mode is set.

## 13.6 AVI WRL

### AVI WRL

When this flag is set, Fractal Imaginator generates single frames in wrl format instead of opening a video stream. The 3D object files can be exported into a program such as Bryce, for post-processing into videos. This works in conjunction with the Demo/Batch mode command, to set the target disk directory and file name, so is only enabled when Batch mode is set.

## 14 Demo menu

### Demo menu commands

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

<a href="#">Random Render</a>	Select a random rendering.
<a href="#">Batch Mode</a>	Repeat random fractal and save to file.
<a href="#">Random Julia</a>	Generate random Julia fractal.
<a href="#">Random Julia2</a>	Generate random Julia fractal(includes composites).
<a href="#">Random Escher</a>	Generate random Julia fractal using Escher mapping.

<a href="#"><u>Random Newton</u></a>	Generate random Julia fractal using Newton or Halley's method.
<a href="#"><u>Random Stalks and Bubbles</u></a>	Generate random orbit-trap or bubble fractal.
<a href="#"><u>Random Quaternion</u></a>	Generate random quaternion/hyperion fractal.
<a href="#"><u>Random Quaternion2</u></a>	Generate random quaternion/hyperion fractal(extended formula search).
<a href="#"><u>Random Cubic Mandelbrot</u></a>	Generate random cubic Mandelbrot fractal
<a href="#"><u>Random Cubic Mandelbrot2</u></a>	Generate random cubic Mandelbrot(relaxed formulas/parameters)
<a href="#"><u>Random Cubic Julia</u></a>	Generate random cubic Julia fractal.
<a href="#"><u>Random Octonion</u></a>	Generate octonion/hyper-octonion fractal.
<a href="#"><u>Random Octonion2</u></a>	Generate random octonion/hyper-octonion fractal (extended dimensional search).
<a href="#"><u>Random Lsystem</u></a>	Generate random lsystem fractal.

## 14.1 Random Render

### Random Render (Demo menu)

The rendering options for the current fractal are randomized. Does not affect formula or range variables. For quaternion and 3D height fields, a random coloring filter is applied.

## 14.2 Batch Mode

### Batch mode (Demo menu)

Here you set parameters for batching and saving random-generated images to disk. When the Repetitions value is non-zero, up to 1000 random images can be generated and saved to disk. Use a unique Filename to prevent batch files from overwriting existing image files. The Scan Limit directs the program on how many scans it makes through each formula before it skips to a new formula (if an interesting Julia fractal hasn't been found.) For lsystems, the scan limit is a limit on how many mutations on the rules may be done before drawing. There are check boxes that affect only random quaternions or lsystems also.

New: If you select this option before opening a video stream, then instead of an AVI stream, the program initializes a set of bmp image files. Each 'frame' is written to the directory specified in the Demo/Batch mode window. If AVI Object or AVI WRL is checked before opening the stream, then the frames are written as a series of 3D object files. Each frame is numbered with a postfix to the Batch-mode name from 0000 to 9999, e.g. 'quat0001.bmp'.

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
- 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  $c_j - c_K$   
Genetic -- (default off) check to randomize genetic word strings and 'genes' in Formula window

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

For lsystems, you can set the mutation mode to Fixed, to allow a limited number of mutation factors. This results in an lsystem that remains the same number of lines as the original (un-mutated) lsystem. The base lsystem can be chosen at random, or only the current lsystem is mutated over and over (Use the Edit/Axiom window to set the current lsystem.) Each random lsystem starts with the base lsystem, to keep some consistency in the mutated image. You can also control the size of the lsystem (small mutations can create very large lsystems, which can bog down the batch mode) by setting the Max Lines variable to a smaller value (default is 25000.) A good value to use is 5000. Up to 5000 lines of the lsystem will be drawn before moving on to the next lsystem in the batch. Note: the max lines variable isn't saved with the image's data file, so the image can be recreated with any suitable lines limit, when not in batch mode. Many lsystems don't exceed 5000 lines.

## 14.3 Random Julia

### Random Julia (Demo menu)

A random Julia fractal is generated. Many of the built-in options of Fractal Imaginator are selected on a random basis, and the Mandelbrot space for one of the hundred built-in formulas is scanned for an interesting Julia set. The palette used is also randomized. Note: In most case 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. The algorithm doesn't check the type of split palette that is used, so a mismatch in the "Divide by" color option may need correcting, e.g. Divide-by-four may be selected for a divide-by-eight palette. Feel free to experiment with all the parameters, reframe the image, change palettes etc. This routine provides a fast intro to many options in Fractal Imaginator that the user may be unfamiliar with: no knowledge of fractal science/math required! See the [hot keys](#) section also for a description of the 'F' command.

## 14.4 Random Julia2

### Random Julia (Demo menu)

A random Julia fractal is generated. Many of the built-in options of Fractal Imaginator are

selected on a random basis, and the Mandelbrot space for one of the hundred built-in formulas is scanned for an interesting Julia set. The palette used is also randomized. Note: In most case 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.

This is like the Random Julia command, except that more options are randomized, including spin and switch, and the Formula Type can be composite or Escher, so the search/draw time may be somewhat longer, and the results not as certain. But the images can be quite weird!

Tip: some things remain to be done after the Julia set is drawn. The algorithm doesn't check the type of split palette that is used, so a mismatch in the "Divide by" color option may need correcting, e.g. Divide-by-four may be selected for a divide-by-eight palette. Feel free to experiment with all the parameters, reframe the image, change palettes etc. This routine provides a fast intro to many options in Fractal Imaginator that the user may be unfamiliar with: no knowledge of fractal science/math required! See the [hot keys](#) section also for a description of the 'F' command.

## 14.5 Random Escher

### Random Escher (Demo menu)

A random Julia fractal is generated using Escher-like tilings (Type 10 in the Formula window.) The Rise variable in the Parameters window sets the degree of tiling (1-235). Note: Sometimes it helps to reduce iterations if the tilings do not appear to be distributed throughout the circular plane. 20-25 iterations is sufficient for many Escher plots.

## 14.6 Random Newton

### Random Newton/Halley (Demo menu)

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

## 14.7 Random Stalks and Bubbles

### Random Stalks and Bubbles (Demo menu)

A random Julia fractal is generated using one of Paul Carlson's orbit traps or bubble method, or a custom orbit-trap formula for Newton's method renderings.

## 14.8 Random Quaternion

### Random Quaternion (Demo menu)

A random quaternion/hypernion fractal is generated. Like Random Julia, a set of formulas

appropriate for quaternions is scanned to find an interesting Julia set, and then the parameters are adjusted to produce a quaternion or hypernion image. The ranges are reset, Hj is set to 2.0, and the lighting is set for optimum viewing.

Note: for some images an hj value of 2.0 may result in a partially clipped image. Sometimes it helps to increase this value to 2.5 or 3.0, but too high a value may interfere with Solid guessing.

See the [hot keys](#) section also for a description of the 'G' command.

## 14.9 Random Quaternion2

### Random Quaternion2 (Demo menu)

A random quaternion/hypernion fractal is generated. Like Random Julia, a set of formulas appropriate for quaternions is scanned to find an interesting Julia set, and then the parameters are adjusted to produce a quaternion or hypernion image. The ranges are reset, Hj is set to 2.0, and the lighting is set for optimum viewing.

This option uses an extended set of formulas, not all of which may produce useable images all the time. The images can be quite different from the traditional quaternion.

Note: for some images an hj value of 2.0 may result in a partially clipped image. Sometimes it helps to increase this value to 2.5 or 3.0, but too high a value may interfere with Solid guessing.

See the [hot keys](#) section also for a description of the 'G' command.

## 14.10 Random Cubic Mandelbrot

### Random Cubic Mandelbrot (Demo menu)

A random cubic Mandelbrot fractal is generated. Like the inverse of Random Julia, the essential cubic parameters are randomly adjusted to point into the four-dimensional formula G0, and then the Mandelbrot set is scanned to find an interesting area to zoom into. The ranges are reset, and the lighting is set for optimum viewing.

## 14.11 Random Cubic Mandelbrot2

### Random Cubic Mandelbrot2 (Demo menu)

A random cubic Mandelbrot fractal is generated. Like the inverse of Random Julia, the essential cubic parameters are randomly adjusted to point into the four-dimensional formula G0, and then the Mandelbrot set is scanned to find an interesting area to zoom into. The ranges are reset, and the lighting is set for optimum viewing.

This option uses the cubic formulas G0 and G1, with relaxed parameters to create cubic-

Mandelbrot like fractals that may extend to six or more dimensions.

## 14.12 Random Cubic Julia

### 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 Julia, a set of formulas (G0 and G1) appropriate for cubic Julias is scanned to find an interesting Julia set, and then the parameters are adjusted to produce the cubic image. The ranges are reset, H<sub>j</sub> is set to 2.0, and the lighting is set for optimum viewing. 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.

## 14.13 Random Octonion

### 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 H<sub>0</sub>-H<sub>9</sub>, and then the octonion Mandelbrot set is scanned to find an interesting area to zoom into. The ranges are reset, and the lighting is set for optimum viewing.

## 14.14 Random Octonion2

### Random Octonion2 (Demo menu)

A random octonion Julia fractal is generated. The essential octonion parameters are randomly adjusted to point into an eight-dimensional formula H<sub>0</sub>-H<sub>9</sub>, and then the octonion Mandelbrot set is scanned to find an interesting area to zoom into. The ranges are reset, and the lighting is set for optimum viewing.

This option uses the octonion formulas H<sub>0</sub>-H<sub>9</sub>, with random dimensional switching (one of OE-OK for O<sub>i</sub>) to create octonion fractals that may extend to eight dimensions.

## 14.15 Random Lsystem

### Random LSystem (Demo menu)

A random lsystem fractal is generated. An .ls file is selected at random from the startup directory, the palette randomized, and then a random amount of mutation is applied. The lighting is set for optimal viewing, and the figure rotated a random amount on the x y and z-axis.

## 15 Growth menu

### Growth menu commands

The Growth menu offers the following commands, which illustrate various features of Fractal Imaginator:

<a href="#"><u>BBM 20</u></a>	Julius set of 441 images
<a href="#"><u>BBM 5</u></a>	Set of 25 images in one window
<a href="#"><u>BBM 1</u></a>	Display single image.
<a href="#"><u>Non-standard J-set</u></a>	Non-standard Julius set
<a href="#"><u>Next Set</u></a>	Display next BBM 20 or BBM 5 set
<a href="#"><u>Previous Set</u></a>	Display previous BBM 20 or BBM 5 set
<a href="#"><u>Show Orbits</u></a>	Show Julia set environment (escape rings)
<a href="#"><u>Cores</u></a>	Cores in Julia set
<a href="#"><u>J-set -- <math>z^2+c</math></u></a>	Julius set -- $z^{\text{power}}+c$
<a href="#"><u>J-set -- <math>z^{-2}+c</math></u></a>	Julius set -- $z^{(-\text{power})}+c$
<a href="#"><u>J-set -- <math>c*\sin(z)</math></u></a>	Julius set -- $c*\sin(z)$
<a href="#"><u>J-set -- <math>c*\cos(z)</math></u></a>	Julius set -- $c*\cos(z)$
<a href="#"><u>J-set -- <math>c*\tan(z)</math></u></a>	Julius set -- $c*\tan(z)$
<a href="#"><u>J-set -- <math>c*\exp(z)</math></u></a>	Julius set -- $c*\exp(z)$
<a href="#"><u>Basic Growth</u></a>	Growth of Basic Awareness
<a href="#"><u>Business Growth</u></a>	Growth of Business Awareness
<a href="#"><u>Global Growth</u></a>	Growth of Global Awareness
<a href="#"><u>Starting Business Growth</u></a>	Start of Business Awareness
<a href="#"><u>Business Core Growth</u></a>	Growth of the core of Business Awareness
<a href="#"><u>Ending Business Growth</u></a>	Growth of the end of Business Awareness
<a href="#"><u>Symmetric Growth</u></a>	Symmetric growth of the complex Sine
<a href="#"><u>Exp (Same Color Growth)</u></a>	Symmetric growth of the complex exponential function
<a href="#"><u>Basic X Growth</u></a>	Growth on the x-axis
<a href="#"><u>Basic Core Growth</u></a>	Growth near the core
<a href="#"><u>Dante Growth</u></a>	Dante model growth
<a href="#"><u>Model 25S</u></a>	Organizational 25S Model growth
<a href="#"><u>Automatic 8-Fold Symmetry</u></a>	Eight-fold symmetry
<a href="#"><u>Automatic 4-Fold Symmetry</u></a>	Four-fold symmetry
<a href="#"><u>Mito Growth</u></a>	Growth of DNA and Proteins
<a href="#"><u>Mito 25</u></a>	25 images of Mito Growth

### 15.1 BBM 20

#### BBM 20

This option displays the Julius Ruis set of 441 images in one window (BBM 20), without setting any of the default values that are done for the Awareness example options. So here you define your own ranges for Starting 'c', Orbits Limit and Set Limit, as well as incremental values.

## 15.2 **BBM 5**

### **BBM 5**

This option displays a set of 25 images in one window (BBM 5), without setting any of the default values that are done for other Awareness example options. So here you define your own ranges for Starting 'c', Orbits Limit and Set Limit, as well as incremental values.

## 15.3 **BBM 1**

### **BBM 1**

Use this option to display a single image from one of the Growth examples. Use the hot key 's' to save the complex c of the image while the Growth example is running. Then select this option to redisplay the image at any resolution and with any rendering style supported by Imaginator.

## 15.4 **Non-standard J-set**

### **Non-standard J-set**

Use this option to display a Julius set of 441 images that run consecutively instead of row by row. With a standard JR-set, ci is set to the "Starting ci" at the beginning of each row. Non-standard J-sets are most effectively displayed when one of the axis is set to zero, as when either "Increment cr" or "Increment ci" is zero.

## 15.5 **Next Set**

When the this command is issued, the Set Limit and Orbits Limit are multiplied by their incremental complements, Set Dec and Orb Inc, forming a new BBM 20 or BBM 5.

## 15.6 **Previous Set**

When the this command is issued, the Set Limit and Orbits Limit are divided by their incremental complements, Set Dec and Orb Inc, forming a new BBM 20 or BBM 5.

## 15.7 **Show Orbits**

### **Show Orbits**

Here you have the option to show escape rings around the Julia sets or exclude them to just show the inner Julia set. This is what BBM calls the Julia set "environment". The set is called the Julius Ruis set with environmentt or just Julius set without the enviroment.

## 15.8 **Cores**

### **Cores**

These options control how the inner core of the Julia set is revealed. With zero cores, the inner core will be solid. With one or two cores, depending on the value of

the Set Limit, there will be the displayed circular patterns inside the Julia cores, sort of like how the Level Curve options in the Render menu work.

### 15.9 J-set -- $z^2+c$

J-set --  $z^2+c$

The Julius Ruis Set of the complex polynomial ( $z'=z^2+c$ ) is displayed. This set is used for the presentation of BBM's Business Awareness routine.

### 15.10 J-set -- $z^{-2}+c$

J-set --  $z^{-2}+c$

The Julius Ruis Set of complex inverted polynomial ( $z'=z^{-2}+c$ ) is displayed. This set is used for the presentation of BBM's Basic Awareness routine.

### 15.11 J-set -- $c*\sin(z)$

J-set --  $c*\sin(z)$

Julius Ruis Set of the complex sine is displayed. This set is called a transcendental function and is used in BBM's Global Awareness routines.

### 15.12 J-set -- $c*\cos(z)$

J-set --  $c*\cos(z)$

Julius Ruis Set of the complex cosine is displayed. This set is called a transcendental function and is used in BBM's Global Awareness routines.

### 15.13 J-set -- $c*\tan(z)$

J-set --  $c*\tan(z)$

Julius Ruis Set of the complex tangent is displayed. This set is called a transcendental function and is used in BBM's Global Awareness routines.

### 15.14 J-set -- $c*\exp(z)$

J-set --  $c*\exp(z)$

Julius Ruis Set of the complex exponential function is displayed. This set is called a transcendental function and is used in BBM's Global Awareness routines.

## 15.15 Basic Growth

### Basic Growth

This example shows the growth of Basic Awareness. A Julia set ( $z' = z^{-z} + c$ ) is used, changing on the x-axis from  $x=-2$  until  $x=2$ .

## 15.16 Business Growth

### Business Growth

This example shows the growth of Business Awareness. A Julia set ( $z' = z^z + c$ ) is used, changing on the x-axis from  $x=-2$  until  $x=2$ .

## 15.17 Global Growth

### Global Growth

This example shows the growth of Global Awareness. A Julia set (complex sin) is used, changing on the x-axis from  $x=-0.2$  until  $x=0.2$ .

## 15.18 Starting Business Growth

### Starting Business Growth

This example shows the growth of the start of Business Awareness. A Julia set ( $z' = z^z + c$ ) is used, changing on the x-axis from  $x=-0.2$  until  $x=0.2$ .

## 15.19 Business Core Growth

### Business Core Growth

This example shows the growth of the core of Business Awareness. A Julia set ( $z' = z^0 + c$ ) is used, changing on the x-axis from  $x=0.504$  until  $x=0$ .

## 15.20 Ending Business Growth

### Ending Business Growth

This example shows the growth of the end of Business Awareness. A Julia set ( $z' = z^z + c$ ) is used, changing on the x-axis from  $x=0.25$  until  $x=0.35$ .

## 15.21 Symmetric Growth

### Symmetric Growth

Symmetric growth of the complex Sine: This example shows different colours in each of four quadrants, just for beauty and fun.



## 15.22 Exp (Same Color Quadrant)

### Exp (Same Color Growth)

Symmetric growth of the complex exponential function: This example shows the same colours for each of four quadrants, just for beauty and fun.

## 15.23 Basic X Growth

### Basic X Growth

This example shows 25 images in one window. Julia sets of the inverted function  $z' = z^{-x} + c$  are presented (changing x from x=-2, until x=1).

## 15.24 Basic Core Growth

### Basic Core Growth

This is an example of beauty of growth near the core of the inverted polynomial  $z' = z^{-x} + c$  (x=-0.01 and y=0). Characteristic for the complex inverted polynomials is the influence of the parameter "Orbits Limit" on the inner side of the picture (Orbits Limit =  $10^{10}$ ).

## 15.25 Dante Growth

### Dante Growth

The Dante model is used in the BBM to show the psychological process going through life-crisis, based on the story of La Divina Comedia written by Dante. 25 pictures in one window are presented, starting with full coherence, going through the area of low coherence (crisis or chaos, or Inferno), turning back into a stable zone (Garden of Eden) and arriving (for a moment) in Heaven.

## 15.26 Model 25S

### Model 25S

This example is used as the basis for the so called organizational 25S Model, an extension of the 7S Model of McKinsey, containing 25 aspects for describing people, teams, business and regions, all starting with the character 'S'. The picture is a Julia set of  $z' = z^6 + c$  starting at x=-1.23 and y=0; for each new picture the x increases with 0.01.

## 15.27 Automatic 8-Fold Symmetry

### Automatic 8-Fold Symmetry

This is a stand-alone part of the program just for creating beautiful pictures of Julia sets, which are build up as an eight-fold symmetry.

## 15.28 Automatic 4-Fold Symmetry

### Automatic 4-Fold Symmetry

This is a stand-alone part of the program just for creating beautiful pictures of Julia sets, which are build up as a four-fold symmetry.

## 15.29 Mito Growth

### Mito Growth

On the request of Dr. Andras Pellionisz this example has been created for the mitochondrial scientific society, to give an impression of fractal growth of DNA and Proteins.

## 15.30 Mito 25

### Mito 25

This example contains 25 images in one window, showing the possible growth of mitochondrial DNA and Proteins (made on the request of Dr. Andras Pellionisz).

## 16 Help menu

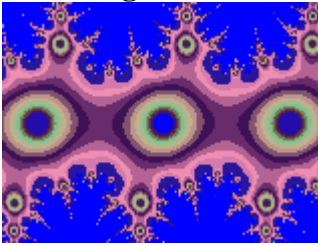
### Help menu commands

The Help menu offers the following commands, which provide you assistance with this application:

<a href="#">Getting Started</a>	Tutorial for new users of Fractal Imaginator.
<a href="#">Index</a>	Offers you an index to topics on which you can get help.
<a href="#">Hot Keys</a>	Quick reference to Fractal Imaginator's hot keys.
<a href="#">Parser Info</a>	Quick reference to Fractal Imaginator's parser variables and functions.
<a href="#">Built-in Formulas</a>	Quick reference to Fractal Imaginator's built-in formulas.
<a href="#">Bibliography</a>	Sources for fractal information and complex numbers.
<a href="#">About Fractal Imaginator</a>	Displays the version number and author info for this application.

## 16.1 Getting Started

### Getting Started



Welcome to Fractal Imaginator!

This is a short tutorial that will cover basic commands and background material necessary for a new user to create an initial picture with Fractal Imaginator. For help on any menu command, press F1 while the command is highlighted. For help on the Edit Formula or Parameters window, click on the Help button inside that window.

If the Auto-Alert option is enabled (on by default), Fractal Imaginator will give an audible sound when the plot is completed. The sound may be customized via the Sounds editor in the Windows 95 control panel. The plot-complete alert uses the Exclamation sound.

Jules Ruis developed the Awareness Governance Model "BBM" (in Dutch being the abbreviation of Bewustzijns Besturings Model), used in Training Interaction Management. The BBM-model is an organisational model that makes a distinction on three levels of awareness: Basic Awareness (on the level of each person individual), Business Awareness (on the level of the business or organisation in which that person works) and Global Awareness (on the level of the region/country in which that person lives).

BBM uses fractal images as a metaphor for the three levels of awareness. For each of these levels a group of mathematical functions is used to show a kind of integrated awareness. The inverted complex polynomials (e.g.  $z' = z^{-z} + c$ ) are used for the basic awareness. Complex polynomials (e.g.  $z' = z^z + c$ ) are used for the business awareness. Complex transcendental functions (e.g. the function for the complex sine, cosine, tan and exponential) are used for the global awareness. All functions are graphically presented in terms of growth. On the edge of all functions (the edge of chaos) the coherence between the different elements (pixels) is very weak. Going more to the inside of fractals the coherence becomes more stable. This characteristic feature of fractals (a stable core and permeable boundaries) makes a fractal very suitable to use as a metaphor for all kind of natural life and social living.

Fractal Imaginator enables you to discover all kind of interesting and beautiful pictures, and is not limited to just the formulas originally contained in BBM. It now includes the formula set of Fractal Zplot and FraSZle plus many of the rendering options found in those programs and extends Julius Ruis sets to 3-D in the form of quaternions and other quad fractal types.

We have programmed twenty examples of BBM to inspire you. Each of the examples in the Growth menu has default values for parameters and in the case of BBM a default formula.

When you select these options Growth/Default Function is automatically selected, so that default values for parameters necessary to these routines are set into play. You may want to change parameters or the formula in the example to something other than the default values, so using Edit/Formula or Edit/Parameters will deselect "Default Function". However, you can use this option to restore some of the essential values in the Formula and Parameter windows by re-selecting it. Some options that are turned on by the random options in the Demo menu, such as orbit-traps and biomorph, are not reset, so you may have to open a new window if the parameters/options have changed too radically and you get lost...

At the opening screen you see a Julius set of the  $z^2+c$  formula. This shows a general map of the Julia basins that make up the famous Mandelbrot set. Double click with the left-mouse button on any of the Julia sets shown and that image will be displayed at the resolution of the draw window. After the Julia set is finished drawing (or as much of it is finished that you want to zoom in on), select the Zoom In/Out command off the Image menu, or just point and click the left-mouse button over any area of the drawing. A box a quarter the size of the window will appear that you can move around with the mouse. Hold the left-mouse button down to shrink the box, or the right-mouse button down to expand the box. Move the box over the area you are zooming in on, size the box if necessary and when it includes the details you want, press the space bar. The plot will be redrawn at zoom scale. To zoom out, you need to draw a smaller size plot, then zoom using a box larger than the plot drawn, or use the Shift key to zoom out by 50% with each keystroke. You can also rotate the zoom box by using the left and right arrow keys. This effectively rotates the area being zoomed into in the reverse direction as the zoom box is rotated. When rotating the zoom box, think of the final image as being a horizontal version of the image in the zoom box.

If you want to zoom into a portion of the Julius set itself, right click with the mouse cursor on an area of the J-set. The cursor changes to a box which you can manipulate as if you were zooming into a single image as explained above.

Fractal Imaginator allows you to Undo the last command in most cases. However this is mostly a failsafe command, as it disables color-cycling and requires you to redraw the fractal to change colors or lighting variables.

This completes the Getting Started tutorial. Be sure to read the [hot keys](#) and [built-in formulas](#) sections for additional info. The [Bibliography](#) lists additional reference material for a better understanding of the fractal types and functions contained in Fractal Imaginator.

## 16.2 Index

### Index command (Help menu)

Use this command to display the opening screen of Help. From the opening screen, you can jump to step-by-step instructions for using Fractal Imaginator and various types of reference information.

Once you open Help, you can click the Contents button whenever you want to return to the opening screen.

## 16.3 Hot Keys

### Hot keys

Ctrl+F1-Ctrl+F9, Ctrl+F11, Ctrl+0-Ctrl+9 --- change to one of 21 color palettes -- useable during plotting.

Ctrl+F12 holds the palette of the most recently loaded function.

Tab --- Replaces the currently-selected palette with the palette in F11.

Useful when you want to make a palette file(.pl) from the palettes in a lot of individual bitmap files. Use the copy data and paste data commands to move the palette from another drawing window into F11. Select the palette(Ctrl+F1-Ctrl+F9, Ctrl+F12, Ctrl+0-Ctrl+9) you want to move Ctrl+F11 into, then press Tab.

up arrow --- forward cycle colors one step, including set color -- useable during plotting.

down arrow --- back cycle colors one step, including set color -- useable during plotting.

Shift-C -- clear the screen to the current background color.

's' -- save current 'c' to vcr and vci -- used with Growth examples and automatic symmetry to save a particular image from a rolling stream of images. In automatic symmetry, the stream of images is halted so you can resize and save a particular image you like. To resize an image after pressing 's' (during automatic symmetry operation) click inside the draw window to exit routine and select Edit/Size to resize the draw window.

'm' -- morph images -- used with automatic symmetry examples. Instead of random 'c', the 'c' value is morphed in a continuous manner.

't' -- enable continuous stream of images -- used with automatic 8-fold and 4-fold symmetry.

'k' -- kill morphing and continuous stream modes -- used with automatic 8-fold and 4-fold symmetry. If both morphing and continuous modes have been enabled, the first time 'k' is used the continuous mode is halted. The second time 'k' is pressed the morphing mode is exited.

Shift-T -- annotate a picture with text. Cursor changes to a crosshatch, which you position over the area where you want the text to start. Then click the left-mouse button to transfer any text (from the Edit/Text window) to the picture. Can be used with Undo. Use the Edit/Text command to change font, text color or format text into multiple lines. This is useful for adding copyright/author info to a finished picture.

## 16.4 Parser

### 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 Fractal Imaginator to represent standard function".

sine z ---  $\sin(z)$  or  $\text{SIN}(Z)$  ; where Z can be any complex expression

hyperbolic sine z ---  $\sinh(z)$  or  $\text{SINH}(Z)$

arcsine z ---  $\text{asin}(z)$  or  $\text{ASIN}(Z)$

cosine z ---  $\cos(z)$  or  $\text{COS}(Z)$

hyperbolic cosine z ---  $\cosh(z)$  or  $\text{COSH}(Z)$

arccosine z ---  $\text{acos}(z)$  or  $\text{ACOS}(Z)$

tangent z ---  $\tan(z)$  or  $\text{TAN}(Z)$

hyperbolic tangent z ---  $\tanh(z)$  or  $\text{TANH}(Z)$

arctangent z ---  $\text{atan}(z)$  or  $\text{ATAN}(Z)$

cotangent z ---  $\text{cotan}(z)$  or  $\text{COTAN}(Z)$

arccotangent z ---  $\text{acotan}(z)$  or  $\text{ACOTAN}(Z)$

$e^z$  ---  $\exp(z)$  or  $\text{EXP}(z)$  -- the exponential function

natural log of z ---  $\log(z)$  or  $\text{LOG}(Z)$

absolute value of z ---  $\text{abs}(z)$  or  $\text{ABS}(Z)$

square root of z ---  $\text{sqrt}(z)$  or  $\text{SQRT}(Z)$

z squared ---  $\text{sqr}(z)$  or  $\text{SQR}(Z)$

real part of z ---  $\text{real}(z)$  or  $\text{REAL}(Z)$

imaginary part of z ---  $\text{imag}(z)$  or  $\text{IMAG}(Z)$

modulus of z ---  $\text{mod}(z)$  or  $\text{MOD}(Z)$  or  $|z|$  --  $(x*x + y*y)$

conjugate of z --  $\text{conj}(z)$  or  $\text{CONJ}(z)$  --  $(x-yi)$

flip(z) ---  $\text{flip}(z)$  or  $\text{FLIP}(Z)$  -- exchange real and imaginary parts of z ( $y+xi$ )

polar angle of z --  $\text{theta}(z)$

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.  
 convergence limit --- cl# or CL# -- the constant entered in the Converge gadget, read-only.  
 cr -- the constant entered in the cr box in the Parameters window(use j# for parser)  
 ci -- the constant entered in the ci box in the Parameters window(use k# for parser)  
 e --- e or E --  $1e^1$  -- 2.71828, read/write.  
 i --- i or I -- square root of -1,read/write.  
 iteration --- iter# -- iteration loop counter  
 j --- j# or J# -- real part of the complex constant, read-only.  
 k --- k# or K# -- coefficient of the imaginary part of the complex constant, read-only.  
 Note: j and k are the actual values of the complex constant terms as they are used in the iteration process, so will vary when the Mandelbrot option is used.  
 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# -- real constant used in phoenix maps; uses the real part of the complex constant when the Phoenix option is chosen, read-only.  
 p1 -- the complex constant entered in the cr and ci gadgets, read-only.  
 pi --- pi or PI -- 3.14159, read/write.  
 q --- q# or Q# -- real constant used in phoenix maps; uses the imaginary part of the complex constant when the Phoenix option is chosen, 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.

## 16.5 Built-in Formulas

**Built-in Formulas** (enter the following prefix into the Function #1 or Function #2 edit boxes)

p0 --  $z^2+c$  --- the standard Mandelbrot or Julia set.  
 p1 --  $cz(1-z)$  --- the self-squared dragon set.  
 p2 --  $c(z-1/z)$  --- alternate Mandelbrot or Julia set.  
 p3 --  $cz^2-1$  --- alternate Mandelbrot or Julia set.  
 p4 --  $c^2/(c+z^2)$  --- alternate Mandelbrot or Julia set.

- p5 --  $z^3+c$  --- cubic Mandelbrot or Julia set.
- p6 --  $((z^2+c-1)/(2z+c-2))^2$  -- renormalization formula #1 for x-plane or q-plane pictures (Note 9).
- p7 --  $z^2+j+kzn$  --- Phoenix curve (Ushiki). Uses Fractint extensions for degree(>2 or <-3).
- p8 -- Julia/Mandelbrot set (modified from M. Barnsley) (Note 2).
- p9 --  $fn(z)-cfn(z)$  -- generalized frothy basin (J. Alexander.) (Note 7).
- r0 -- Newton/Halley map of  $z^3+conj(z)c$  -- exploratory function based on modified frothy basin.
- r1 --  $z^z+z^s+c$  --- Biomorphs, etc.; s may be complex using the si variable as its imaginary component.
- r2 --  $z^s-z+c$  --- Biomorphs, etc.; s may be complex using the si variable as its imaginary component.
- r3 --  $fn(z)+exp(z)+c$  -- Biomorphs, etc.
- r4 -- solves Newton/Halley transformation of  $(z^2-c)(z+1)$ . (Notes 3,4,5,11).
- r5 --  $cfn(z)$  -- transcendental Julia curve, etc.
- r6 --  $cexp(z)$  -- exponential Julia curve, etc. with additional plane checking when real value of Z exceeds 50. If  $\cos(\text{imag-Z}) \geq 0$ , point is considered part of Julia set.
- r7 --  $fn(z)+cfn(z)+1$  -- generalized form of t9.
- r8 -- foggy coastline #1 Mandelbrot IFS (M. Barnsley) (Note 14).
- r9 -- foggy coastline #2 Mandelbrot IFS (M. Barnsley) (Note 14).
- e0 -- solves Newton/Halley transformation of  $(z+j)(z+k)(z^2+1)$  for either Julia or Mandelbrot set.
- e1 -- solves Newton/Halley transformation of  $(z+j)(z^2+z+k)$  for Mandelbrot and Julia set.
- e2 -- solves Newton/Halley transformation of  $(z-1)(z^2+z+c)$  for either Julia or Mandelbrot set.
- e3 -- solves Newton/Halley transformation of  $(z+j)(z+k)(z+1)$  for either Julia or Mandelbrot set.
- e4 -- Chaos Game Julia IFS (M. Barnsley).
- e5 -- snowflake Julia IFS (as described in Fractals Everywhere by M. Barnsley).
- e6 -- solves Newton/Halley transformation of  $\log z-c$ .
- e7 -- solves Newton/Halley transformation of  $exp(z)-c$ .
- e8 -- solves Newton/Halley transformation of  $(z-c)(z+1)(z-1)$  for Mandelbrot or Julia set.
- e9 -- solves Newton/Halley transformation of  $(z-c)(z+c)(z^2+c^2)$  ---  $z^4-c^4$ .
- s0 -- solves Newton/Halley transformation of  $\sin z-c$ .
- s1 --  $sexpz+c$  -- transcendental Mandelbrot or Julia set.
- s2 --  $c(1+z^2)^2/(z^2-1)$  -- alternate Mandelbrot/Julia set.
- s3 -- solves Newton/Halley transform of  $\tan(z)-c$ .
- s4 -- IFS ( $x=sy+j, y=-sx+k$  ( $x>0$ )); else  $x=sy-j, y=-sx-k$  (modified from M. Barnsley) (An alternate version of this formula is executed when the limit value is non-integral.)
- s5 -- solves Newton/Halley transform of  $z^s-1$  (Julia set only).
- s6 -- composite function  $cz-c/z$  &  $z^2+c$  (C. Pickover).
- s7 -- transcendental function  $fn(z)+c$ .
- s8 --  $((z^3+3(c-1)z+(c-1)(c-2))/(3z^2+3(c-2)z+c^2-3c+3))^2$  -- renormalization formula #2 for x-plane or q-plane pictures .



- s9 -- Newton/Halley map of  $z(z^{\text{limit}-1})$  (Julia set only).
- t0 -- Newton/Halley map of  $z(z^{\text{limit}-c})$  (Julia or Mandelbrot set ; display method 2 default;  $\text{limit} \geq 1.0$  ; use 1.0 for initial  $z$  with Mandelbrot0 type, or use MandelbrotP type.).
- t1 -- Newton/Halley map of Chebyshev function  $\cos(n \cdot \arccos x)$ .
- t2 -- Newton/Halley map of Hermite polynomial:  $16x^4 - 48x^2 + 12$ .
- t3 -- alternate Newton/Halley map of  $\tan z - c$  (Julia or Mandelbrot set.) the twist is in the second derivative of the Halley type.
- t4 -- Newton/Halley map of  $z^{\text{limit}-c}$  (Julia or Mandelbrot set ; display method 2 default;  $\text{limit} \geq 1.0$  ; use 1.0 for initial  $z$  with Mandelbrot0 type, or use MandelbrotP type.).
- t5 --  $\text{fn}(\text{fn}(z)) + c$  -- user-defined complex set. When the first function is  $z^2$  and the second function is  $\text{conj}(z)$ , this becomes the  $z$ -conjugate set,  $zz^2 + c$ , the tricorn set. (Note 12).
- t6 -- Volterra-Lotka equations discretized by modified Huen method (from The Beauty of Fractals).
- t7 --  $c^z + c$  -- tetration of  $z$ .
- t8 --  $q^2 + c$  -- Quaternion set (from Computer, Pattern, Chaos and Beauty) (Note 8).
- t9 --  $z + cz + 1$  -- try with Newton's method applied. A buggy algorithm found this one.
- a0 -- spiral network -- C. Pickover.
- a1 --  $z - (1/z + c)$  -- try with renormalization applied. Sequel to t9.
- a2 --  $\text{fn}(z) - (\text{fn}(z) + c)$  -- generalized form of a1.
- a3 -- alternate Newton/Halley map of  $\sin z - c$ . see t3 variant.
- a4 -- user-defined complex set:  $\text{fn}(z) + \text{fn}(z) + c$ .
- a5 -- hypercomplex Newton/Halley map of  $h^3 + c$ .
- a6 -- Hypercomplex Newton/Halley map of  $\text{fn}(h) + c$ .
- a7 -- user-defined complex set:  $\text{fn}(z) + \text{fn}(c)$ .
- a8 --  $\text{fn}(z) + zn + c$  -- from Fractal Creations.
- a9 --  $q^3 + c$  -- cubic Quaternion set.
- b0 -- alternate Newton/Halley map of  $\exp(z) - c$ . as for t3 variant.
- b1 -- alternate Newton/Halley map of  $\log(z) - c$ .
- b2 -- Newton/Halley map of phoenix curve.
- b3 --  $\text{cfn}(z) + zn$  -- user-defined complex formula.
- b4 --  $\text{fn}(z) + kzn + j$  -- generalized phoenix curve formula.
- b5 --  $\text{fn}(z)$  a preformula for use with type 3 composite fractals. uses limit gadget to select function.
- b6 -- Newton/Halley map of  $\text{fn}(z) + \text{fn}(z) + c$ .
- b7 -- Newton/Halley map of  $\text{cfn}(z)$ .
- b8 --  $\text{fn}(z) * \text{fn}(z) + c$ .
- b9 -- Newton/Halley map of foggy coastline #1.
- c0 -- Newton/Halley map of foggy coastline #2.
- c1 -- Newton/Halley map of  $\text{fn}(\text{fn}(z)) + c$ .
- c2 --  $\text{cfn}'(z)$ , where  $\text{fn}'(z)$  = first derivative of user-defined function.
- c3 --  $\text{fn}(z) + \text{fn}'(z) + c$ .
- c4 --  $\text{fn}'(z) + \text{fn}(c)$ .
- c5 --  $\text{fn}(\text{fn}'(z))$ .

- c6 -- first order gamma function:  $(z/e)^z \sqrt{2\pi z} + c$ .
- c7 -- Newton/Halley map of fifth degree Legendre polynomial:  $1/8(63z^5 - 70z^3 + 15z)$ ; display method 2 default.
- c8 --  $(z^2 + e^{-z})/(z+1)$ : second-order convergence formula for finding root of  $ze^z - 1 = 0$ .
- c9 -- Newton/Halley map of  $fn(z) * fn(z) + c$ .
- d0 --  $z^s / \text{limit} + c$ : anti-derivative of  $z^n$ ; s may be complex using the si variable as its imaginary component.
- d1 -- Sterling expansion of gamma function:  $(z/e)^z \sqrt{2\pi/z} + c$ .
- d2 -- Newton map of  $fn'(z) - fn(z) + c$ : generalized first degree Laguerre polynomial. Newton map only. (Note 13).
- d3 --  $fn(1/(fn(z) + c))$ .
- d4 --  $z^2 - c$ ; where  $z_{\text{real}} = \text{abs}(z_{\text{real}})$  (Paul Carlson's "alien" Julia set).
- d5 --  $z^2$ ; where  $z_{\text{real}} = \text{abs}(z_{\text{real}}) - cr$ ,  $z_{\text{imag}} = z_{\text{imag}} - ci$  (Paul Carlson Julia set).
- d6 --  $cfn(z) + c$ .
- d7 -- Newton's method applied to  $(x^3 + y^2 - cr = 0$  and  $y^3 - x^2 + ci = 0)$ . Newton map only. from Sylvie Gallet and Fract19.par.
- d8 -- Newton's method applied to  $fn1(x) + fn2(y) - cr = 0$  and  $fn3 - fn4 + ci = 0$  (Note 15) Newton map only.
- d9 -- Bill13 from Bill Rossi via the Internet.
- f0 -- generalized form of Earl Hinrichs' sophomore sine function(ssin) --  $\text{limit} * fn(z) + s + si$ , where  $fn(z) = (fn1(x), fn2(y))$ .
- f1 --  $c^2 / (1 - cz^2)$  -- variant of p4.
- f2 -- Gallet-4-01, from Sylvie Gallet's extensive Internet collection (Note 16).
- f3 -- Gallet-4-02, from Sylvie Gallet.
- f4 -- Gallet-6-01, from Sylvie Gallet.
- f5 -- Gallet-6-02, from Sylvie Gallet.
- f6 -- Gallet-6-03, from Sylvie Gallet.
- f7 -- Gallet-6-04, from Sylvie Gallet.
- f8 -- Gallet-6-05, from Sylvie Gallet.
- f9 -- Gallet-7-01, from Sylvie Gallet.
- g0 --  $z^3 - 3c^2z + s$  -- cubic Mandelbrot (Note 17)
- g1 --  $z^3 - 3sz + c$  -- alternate cubic Mandelbrot
- g2 --  $z^3 - 3c^2z^2 + s$  -- cubic Mandelbrot variant
- g3 --  $z^3 - 3sz^2 + c$  -- alternate cubic Mandelbrot variant
- g4 --  $z^3 - 3c^2/z + s$  -- cubic Mandelbrot variant
- g5 --  $z^3 - 3s/z + c$  -- alternate cubic Mandelbrot variant
- g6 --  $z^3 - 3c^3 * z + s$  -- cubic Mandelbrot variant
- g7 --  $z^3 - 3s/z^2 + c$  -- alternate cubic Mandelbrot variant
- g8 --  $z^3 - 3c^2 + z + s$  -- cubic Mandelbrot variant
- g9 --  $z^3 - 3s + z + c$  -- alternate cubic Mandelbrot variant
- h0 --  $(O * C^{-1})(C * O) + c$  -- octonion set (Note 4)
- h1 --  $cO * CC(1 - C * O)$  -- octonion set
- h2 --  $O * C(O - CC * O) + c$  -- octonion set
- h3 --  $cO^2 + O^2 * CC - c$  -- octonion set

h4 --  $O^2*CC+O^2*C+c$  -- octonion set  
 h5 --  $O^3*CC+c$  -- octonion set  
 h6 --  $O^3*CC+O^3*C+c$  -- octonion set  
 h7 --  $O^4*CC+c$  -- octonion set  
 h8 --  $cO^3*CC+c$  -- octonion set  
 h9 --  $O^2*CC+O^3*C+c$  -- octonion set

i0 --  $2*z*c\#\cos(\pi/z)$  -- Godwin Vickers  
 i1 --  $2*z*c\#\sin(\pi/z)$  -- Godwin Vickers  
 i2 --  $2*z*c\#\tan(\pi/z)$  -- Godwin Vickers  
 i3 --  $1/z^3+\sin(z)*c^2$   
 i4 --  $\cosh(z)/c*z+c^2$   
 i5 --  $\exp(z)/z^3-c$   
 i6 --  $\cosh(z)*z^2-c^2$   
 i7 --  $1/z^2-cz-c$   
 i8 --  $\tan(z)-czc$   
 i9 --  $(\tan(z)-1/z^3)/c^2$

j0 --  $\cosh(z)*\cos(z)+1/c$   
 j1 --  $z^4/(z^3-c^2)$   
 j2;  $1/z^2-\tan(z)+c$   
 j3 --  $\tan(z)/z^3+c$   
 j4 --  $1/z^3-cz+1/c$   
 j5 --  $z^3+\tan(z)*c$   
 j6 --  $1/z^3-\tan(z)-c$   
 j7 --  $\tan(z)-\sin(z)+c$   
 j8 --  $1/z^2-\tan(z)+1/c$   
 j9 --  $z^4+\sin(z)/c$

k0 -- Mandelbrot set(sine variation)  
 k1 --  $z^{1.5}+c$  -- Godwin Vickers  
 k2 --  $(z^2-z^{(2-s)})/s+c$  -- Escher set by Roger Bagula  
 k3 --  $z^2+z/(|z|+c)$  -- Roger Bagula  
 k4 -- quantum set -- S.M. Ulam  
 k5 -- prey predator #1 -- Roger Bagula  
 k6 -- prey predator #2 -- Roger Bagula  
 k7 -- Klein group #1 -- Roger Bagula  
 k8 -- Klein group #2 -- Roger Bagula  
 k9 -- Klein group #3 -- Roger Bagula

L0 -- Loxodromic by Thomas Kroner (fixed type)  
 L1 -- squared loxodrome  
 L2 -- Gedatou by Thomas Kroner (fixed type)  
 L3 -- Ventri by Thomas Kroner (fixed type)  
 L4 -- squared gedatou  
 L5 --  $fn(z)-cfn(z)$  (Note 5)  
 L6 --  $fn(z)+fn(z)+c$

L7 --  $\text{cfn}(z)+c$ "

L8 --  $\text{fn}(z)+\text{cfn}(z)+1$ "

L9 --  $\text{fn}(z)+c$

m0 --  $\text{fn}(\text{fn}(z))+c$

m1 --  $\text{fn}(z)+\text{fn}(c)$

m2 --  $\text{fn}(z)+z^n+c$

m3 --  $\text{cfn}(z)+z^n$

m4 --  $\text{fn}(z)+kz^n+j$  -- generalized phoenix curve

m5 --  $\text{fn}(z)*\text{fn}(z)+c$

m6 --  $\text{fn}(1/(\text{fn}(z)+c))$

m7 --  $(1/\text{fn}(z))^2+c$

m8 --  $(1/\text{fn}(z))^3+c$

m9 --  $\text{fn}(z)/(1-\text{fn}(z))+c$

n0 -- Sinus by Thomas Kromer (fixed type)

n1 -- Sinus #2 by Thomas Kromer (fixed type) (Note 18)

n2 -- Rings of Fire by Thomas Kromer (fixed type) (Note 18)

n3 -- Teres by Thomas Kromer (fixed type)

n4 --  $z^2+y+c$

n5 --  $z^2+y[n+1]+c$

n6 --  $z^2+zi+c$

n7 --  $z^2+zi[n+1]+c$

n8 --  $zr^2+3zi+c$

n9 --  $zr^2+4zi+c$

#### FraSZle formulas

1 p0;  $z^2+c$

2 p1;  $cz(1-z)$

3 p2;  $z(z-1/z)+c$

4 p3;  $cz^2-c$

5 p4;  $z^2+cz^2+c$

6 p5;  $z^3+c$

7 p6;  $((z^2)*(2z+c))^2+c$

8 p7;  $z^2+j+kzn$

9 p8;  $(x^2-y^2-j, 2xy-k)$  when  $x>0$ ; else  $(x^2-y^2-c+jx, 2xy+kx-k)$  -- Barnsley (Note 2)

10 p9;  $z^2-cz^3+c$

11 r0;  $z^3+\text{conj}(z)c+c$

12 r1;  $z^z+z^3+c$

13 r2;  $z^3-z+c$

14 r3;  $z^2+\exp(z)+c$

15 r4;  $(z^2-c)(z+1)$

16 r5;  $cz^3+c$

17 r6;  $z^2+c\exp(z)+c$

18 r7;  $\sin(z)+cz^2+c$

19 r8;  $(z-1)/c$  when  $x>=0$ ; else  $(z+1)/cc$  -- Barnsley

20 r9;  $(z-1)/c$  when  $kx-jy>=0$ ; else  $(z+1)/c$  -- Barnsley

21	e0; $(z+j)(z+k)(z^2+1)+c$
22	e1; $(z+j)(z^2+z+k)+c$
23	e2; $(z-1)(z^2+z+c)$
24	e3; $(z+j)(z+k)(z+1)+c$
25	e4; chaos formula -- Barnsley
26	e5; snowflake IFS -- Barnsley
27	e6; $\cos(z)+c$
28	e7; $z^3+\exp(z)+c$
29	e8; $(z-c)(z+1)(z-1)+c$
30	e9; $z^4-c^4$
31	s0; $\sin(z)-c$
32	s1; $z^4+\exp(z)+c$
33	s2; $(c(z^2+1)^2)/(z^2-1)$
34	s3; $z^2+\tan(z)+c$
35	s4; strange attractor IFS ( $s=1.4142$ ) -- Barnsley
36	s5; $z^5+c$
37	s6; composite function $cz-c/z$ && $z^2+c$
38	s7; $1/z^2+c$
39	s8; $(z^3+3(c-1)z+(c-1)(c-2))$
40	s9; $z(z^5+c)$
41	t0; $z(z^6+c)$
42	t1; $z^7-z^5+z^3-z+c$
43	t2; $z^4-z^2+c$
44	t3; $z^3+\tan z+c$
45	t4; $z^2+z^c+c$
46	t5; $z^3+c/z+c$
47	t6; discretizes Volterra-Lotka equations via modified Heun algorithm
48	t7; $z^3+c^z+c$
49	t8; Quaternion set -- $q^2+c^2$
50	t9; $z^2+cz^2+c$
51	a0; spiral network -- C. Pickover
52	a1; $z+z^3/c+c$
53	a2; $\tan(z)-(z^2+c)$
54	a3; $z^2+\arcsin z-c$
55	a4; $\cos(z)+\csc(z)+c$
56	a5; $\cos(z^3)+c$
57	a6; $z^7+c$
58	a7; $z^3+\sin(c)$
59	a8; $z^3+zn+c$
60	a9; Quaternion set -- $q^3+c^3$
61	b0; $1/z^2+\exp(z)-c$
62	b1; $1/z^3+\log z-c$
63	b2; $z^3+j+kzn$
64	b3; $cz^2+zn+c$
65	b4; $\sin(z)+kzn+j$
66	b5; $\text{vers}(z)+c$
67	b6; $\text{vers}(z)+\text{covers}(z)+c$

- 68      b7;  $c \cdot \text{vers}(z) + c$   
69      b8;  $\text{vers}(z) \cdot \text{covers}(z) + c$   
70      b9; (foggy coastline #1)<sup>2</sup>+c -- Barnsley  
71      c0; (foggy coastline #2)<sup>2</sup>+c -- Barnsley  
72      c1;  $\sin(\text{vers}) + c$   
73      c2;  $\text{csec}(z^2) + c$   
74      c3;  $z^3 + \text{sech}(z^2) + c$   
75      c4;  $c \cdot z^{(c-1)} + z^2$   
76      c5;  $\cos(-1/w^2) + c$   
77      c6;  $(z/e)^z \cdot \text{sqr}(2 \cdot \pi \cdot z) + c$   
78      c7;  $1/8(63z^5 - 70z^3 + 15z) + c$   
79      c8;  $(z^2 + e^{(-z)}) / (z+1) + c$   
80      c9;  $z^2 \cdot \exp(z) + c$   
81      d0;  $(z^3) / c + c$   
82      d1;  $z^3 \cdot \text{sqr}(2 \cdot \pi / z) + c$   
83      d2;  $z^2 - \text{csc}(h) \cot(h) + c$   
84      d3;  $(1 / (\sin(z) + c))^3$   
85      d4;  $z^2 - c$ ; where  $x = \text{abs}(\text{real}(z))$ ,  $y = \text{imag}(z)$  -- Paul Carlson  
86      d5;  $z^2$ ; where  $x = \text{abs}(\text{real}(z)) - cr$ ,  $y = \text{imag}(z) - ci$  -- Paul Carlson  
87      d6;  $c \cdot \cos(z) + c$   
88      d7;  $z \cdot \cos(z) + c$   
89      d8;  $z^2 + z / \sin(z) + c$   
90      d9;  $z^2 + z^{\pi} + c$   
91      f0;  $z^2 + z^3 + \sin(c) + \cos(c) + c$   
92      f1;  $z^2 \cdot (1 - cz^2) + c$   
93      f2;  $1 / (z \cdot z / c) + c$   
94      f3;  $1 / (z \cdot z) - z + c$   
95      f4;  $(1 + c\#) / (z \cdot z) - z$   
96      f5;  $(1 + c) / (z \cdot z / c) + c$   
97      f6;  $(1/c) / (z \cdot z / c) + c$   
98      f7;  $1 / ((z \cdot z) / c) + (c / (1/z - c))$   
99      f8;  $1 / (z \cdot z \cdot z / c) + c$   
100     f9;  $1 / (z \cdot z \cdot z) - z + c$   
101     g0;  $z^3 - 3c^2z + s$  -- cubic Mandelbrot (Note 17)  
102     g1;  $z^3 - 3sz + c$  -- alternate cubic Mandelbrot  
103     g2;  $z^3 - 3c^2z^2 + s$  -- cubic Mandelbrot variant  
104     g3;  $z^3 - 3sz^2 + c$  -- alternate cubic Mandelbrot variant  
105     g4;  $z^3 - 3c^2/z + s$  -- cubic Mandelbrot variant  
106     g5;  $z^3 - 3s/z + c$  -- alternate cubic Mandelbrot variant  
107     g6;  $z^3 - 3c^3z + s$  -- cubic Mandelbrot variant  
108     g7;  $z^3 - 3s/z^2 + c$  -- alternate cubic Mandelbrot variant  
109     g8;  $z^3 - 3c^2 + z + s$  -- cubic Mandelbrot variant  
110     g9;  $z^3 - 3s + z + c$  -- alternate cubic Mandelbrot variant  
111     h0;  $(O \cdot C^{\wedge} - 1)(C \cdot O) + c$  -- octonion set (Note 19)  
112     h1;  $cO \cdot CC(1 - C \cdot O)$  -- octonion set  
113     h2;  $O \cdot C(O - CC \cdot O) + c$  -- octonion set  
114     h3;  $cO^2 + O^2 \cdot CC - c$  -- octonion set

- 115 h4;  $O^2*CC+O^2*C+c$  -- octonion set  
 116 h5;  $O^3*CC+c$  -- octonion set  
 117 h6;  $O^3*CC+O^3*C+c$  -- octonion set  
 118 h7;  $O^4*CC+c$  -- octonion set  
 119 h8;  $cO^3*CC+c$  -- octonion set  
 120 h9;  $O^2*CC+O^3*C+c$  -- octonion set  
 121 i0;  $2*z*c\#\cos(\pi/z)$  -- Godwin Vickers  
 122 i1;  $2*z*c\#\sin(\pi/z)$  -- Godwin Vickers  
 123 i2;  $2*z*c\#\tan(\pi/z)$  -- Godwin Vickers  
 124 i3;  $1/z^3+\sin(z)*c^2$   
 125 i4;  $\cosh(z)/c*z+c^2$   
 126 i5;  $\exp(z)/z^3-c$   
 127 i6;  $\cosh(z)*z^2-c^2$   
 128 i7;  $1/z^2-cz-c$   
 129 i8;  $\tan(z)-czc$   
 130 i9;  $(\tan(z)-1/z^3)/c^2$   
 131 j0;  $\cosh(z)*\cos(z)+1/c$   
 132 j1;  $z^4/(z^3-c^2)$   
 133 j2;  $1/z^2-\tan(z)+c$   
 134 j3;  $\tan(z)/z^3+c$   
 135 j4;  $1/z^3-cz+1/c$   
 136 j5;  $z^3+\tan(z)*c$   
 137 j6;  $1/z^3-\tan(z)-c$   
 138 j7;  $\tan(z)-\sin(z)+c$   
 139 j8;  $1/z^2-\tan(z)+1/c$   
 140 j9;  $z^4+\sin(z)/c$   
 141 k0; Mandelbrot set(sine variation)  
 142 k1;  $z^{1.5}+c$  -- Godwin Vickers  
 143 k2;  $(z^2-z^2(2-s))/s+c$  -- Escher set by Roger Bagula  
 144 k3;  $z^2+z/(|z|+c)$  -- Roger Bagula  
 145 k4; quantum set -- S.M. Ulam  
 146 k5; prey predator #1 -- Roger Bagula  
 147 k6; prey predator #2 -- Roger Bagula  
 148 k7; Klein group #1 -- Roger Bagula  
 149 k8; Klein group #2 -- Roger Bagula  
 150 k9; Klein group #3 -- Roger Bagula  
 151 L0; Loxodromic by Thomas Kroner (fixed type)  
 152 L1; squared loxodrome  
 153 L2; Gedatou by Thomas Kroner (fixed type)  
 154 L3; Ventri by Thomas Kroner (fixed type)  
 155 L4; squared gedatou  
 156 L5;  $fn(z)-cfn(z)$  (Note 7)  
 157 L6;  $fn(z)+fn(z)+c''$   
 158 L7;  $cfn(z)+c''$   
 159 L8;  $fn(z)+cfn(z)+1''$   
 160 L9;  $fn(z)+c$   
 161 m0;  $fn(fn(z))+c$

162	m1; $fn(z)+fn(c)$
163	m2; $fn(z)+zn+c$
164	m3; $cfn(z)+zn$
165	m4; $fn(z)+kzn+j$ -- generalized phoenix curve
166	m5; $fn(z)*fn(z)+c$
167	m6; $fn(1/(fn(z)+c))$
168	m7; $(1/fn(z))^2+c$
169	m8; $(1/fn(z))^3+c$
170	m9; $fn(z)/(1-fn(z))+c$
171	n0; Sinus by Thomas Kromer (fixed type)
172	n1; Sinus #2 by Thomas Kromer (fixed type) (Note 18)
173	n2; Rings of Fire by Thomas Kromer (fixed type) (Note 18)
174	n3; Teres by Thomas Kromer (fixed type)
175	n4; $z^2+y+c$
176	n5; $z^2+y[n+1]+c$
177	n6; $z^2+zi+c$
178	n7; $z^2+zi[n+1]+c$
179	n8; $zr^2+3zi+c$
180	n9; $zr^2+4zi+c$

Note 1: all pertinent menu flags must be set for built-in functions to work as described.

Note 2: For further info on Michael Barnsley's formulas, see his "Fractals Everywhere".

Note 3: Halley map requires the Newton flag to be set. This is another numerical approximation method for finding complex roots. For all Newton/Halley functions, the Newton map is the default. The Halley option is specified through the Arg Gadget, the second character being set to 'h', after the display method(1-9). E.g. '1hr' would designate a relaxed Halley map with display method 1.

Note 4: Halley and Newton maps can use one of nine display methods:

#1 (the default mode, except for functions using  $\sin z$ ,  $\exp z$ ,  $\log z$  and  $\tan z$ , or  $fn(z)$ , which default to method 2, and don't use methods 1,4 or 5): ---colors represent the root(the zero) which a point converges to.

#2 (if the Arg Gadget is set to 2, or for functions of  $\sin z$ ,  $\tan z$ ,  $\log z$ ,  $\exp z$ , and  $fn(z)$ ): ---colors represent the number of iterations a point takes to converge.

#3 (if the Arg Gadget is set to 3) -- colors represent the number of iterations a point takes to converge according to an alternate formula described by C. Pickover in *Computers, Pattern, Chaos and Beauty*.

#4 (if the Arg Gadget is set to 4) -- a merging of methods 1 and 3. After the point converges according to the alternate formula #3, its roots are colored according to #1.

#5 (if the Arg Gadget is set to 5) -- a variation of method 1, with double-convergence checking inside the loop.

#6, #7 and #8-- alternate convergent formulas.

#9 -- a variation of method 3, with double-convergence checking.

Note 5: An additional third argument that affects the convergence speed of Newton/Halley



maps may be one of the following six methods:

'r': relaxed Newton method uses the formula  $z = z - sf(z)/f'(z)$ .

'm': modified Newton transformation uses the formula:  $z - (f(z)/(f'(z)+si))$ . Note: Si here references the s variable \* i, not the complex variable s+si.

'd': relaxed modified Newton method uses the formula:  $z - (sf(z)/f'(z)+si)$ .

'p': premodified Newton transform uses the formula:  $sz - (f(z)/f'(z))$ .

'c': complex Newton transform uses the formula :  $z - (f(z)/(f'(z)+c))$ , where c is the complex constant.

'n': Nova variation by Paul Derbyshire,  $z - (f(z)/(f'(z))+c)$ .

The s constant is entered via the S gadget.

Note 7: the term 'fn(w)' represents any one of 47 user-defined functions chosen through the f1-f4 gadgets:

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<sub>3</sub>(w): 3rd degree Laguerre polynomial.      36: gamma(w): first order gamma function.  
 37: G(w): Gaussian probability function -- (1/sqr(2pi))\*e^(.5w^2).  
 38: c^(s+si).      39: zero.      40: w^(s+si). 41: |(wx)|+|(wy)|\*i(abs).  
 42: wy+wx\*i(flip).      43: conj(cos(w))--cosxx.      44: theta(w) -- polar angle(w).  
 45: real(w).      46: imag(w).

When only fun#1 or fun#2 is used and a single user-defined function is involved, the function is taken from f1. When two user-defined functions appear in a function, the f2 gadget supplies the second function type, except as noted below. For Newton/Halley maps involving  $z^z$ , the first derivative is defined as  $z^z*(1+\ln(z))$ . An alternate derivative formula ( $z^z*(z-1)$ ) is used when a non-integral value is entered as an arg limit (e.g.: 0.1). (This produces interesting effects, though mathematically inaccurate.) For plots that use both fun#1 and fun#2 (type 2 or 3, etc), fun#1 takes its functions from f1 and f2 and fun#2 takes its functions from f3 and f4.

Note 8: The quaternion and hypercomplex functions use the complex c gadgets to input cr, ci, cj and ck. These may be zero when generating a Mandelbrot-like set of these functions. Julia sets may then be mapped by grabbing points (cr,ci) from interesting areas near this set. Cj and ck must be entered manually for Julia sets. The hj and hk gadgets are used to input the z and w coefficients of the j and k planes. Use small amounts to start for these variables (0-1.0.) Values of 0 for hj, hk, cj and ck result in a two-dimensional slice that matches the standard (non-hypercomplex) type. Higher values of z and w (as well as cj and ck) produce more pronounced asymmetry in the complex mapping.

Note 9: Renormalization functions use the Arg Gadget for plotting options (1-4,6-8) as follows:

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

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

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

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

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

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

Note 10: Most of the built-in functions (except for real Newtons and the Gallet formulas) have hypercomplex extensions when values of cj, ck, hj or hk are non-zero.

Note 11: Hypercomplex Newton/Halley maps use only type 2 and type 3 convergence tests.

Note 12: The default version of hypercomplex conjugate is defined as  $\text{conjugate}(h) = h_r - h_i - h_j + h_k$ . A variant of the hypercomplex conjugate uses an arg limit with a non-integral value (e.g.: 2.1.) This makes all imaginary components of h negative, such that  $\text{conjugate}(h) = h_r - h_i - h_j - h_k$ .

Note 13: The formula for a first degree Laguerre polynomial is  $e^{t(d/dt(t/e^t))} = d/dt(t) \cdot t$ .

Note 14: With a non-integral value entered in the limit gadget, an alternate (Fractint) version of this formula is executed.

Note 15: For real Newtons, the function selected from the f1-f4 boxes is a real function. For a real conjugate, the negation of the real term is used.

Note 16: formulas by Sylvie Gallet have been modified to allow both Mandelbrot and Julia sets to be drawn from them. Except for Gallet-6-02, the bailout is set with the built-in variable zlimit in the Parameters window. Gallet-6-02 uses the complex constant p3 (limit and converge) for bailout.

Note 17: For the traditional cubic Mandelbrot (example in The Science of Fractal Images), hj, hk, cj and ck (hypercomplex components of z and c) should be set to zero when used with the quaternion type.

Cubic Mandelbrots quaternions use the S and Si variables (in the New Formula window) to set the initial value and range of the 3rd dimension. Starting from an initial value of 'S', Si is normally set to  $2 \cdot \text{abs}(S)$ , when the Type is MandelbrotP or Julia Tower. Si can also be set to center the 3rd dimension on something besides 0.0. Si has no affect on Mandelbrot0 or Julia types, height fields or in 2D mode. (In 2D mode, the S variable acts as one of the two fixed

dimensions, along with the Limit variable.) Note: the previous version of FZ (1.19b) used  $S_i$  as an increment to  $S$ , instead of the range of the 3rd dimension. Assuming Steps was 200 in the Quaternion window, you need to multiply  $S_i$  by 200 to retain compatibility with cubic pictures done with version 1.19b. The current version allows you to change the Steps variable (for smoother pictures) without having to change  $S_i$  also.

The Limit variable (NF window) points to the fourth dimension (in the Quaternion window the 4th Dim. variable points to  $hk$ .)

In addition, the Arg value (entered in NF window) has the following affect on cubic Mandelbrots:

- 0 -- compute  $M_+$ , using  $hj$  for  $z$  space
- 1 -- compute  $M_+$ , using greater of  $S$  or  $hj$  for  $z$  space
- 2 -- compute  $M_-$ , using greater of  $S$  or  $hj$  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 quaternion-type cubic Mandelbrots, while args 0-2 can be used in 2D mode, or with height fields.

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(for compatibility with alpha versions of Fractal Imaginator 1.19)

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

Note 18: For the loxodromic functions, Sinus #2 and Rings of Fire, the Arg Limit variable (in the Edit Formula/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.

Note 19: 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:

- $OI$  -- rotate OE-OK to OI-OE
- $OJ$  -- rotate OE-OK to OJ-OI
- $OK$  -- 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 Hyperion.) Hyper-octonions use alternate definitions of the basic octonion multiplication tables. This is similar to the difference between hypercomplex quaternions (hyperions) and quaternions. The algebra for octonion and hyper-octonions differ in how they conform to (or fail in) the associative and commutative laws.

## 16.6 Lsystem info

### Lsystem info

Fractal Imaginator contains an extended version of the 0L-system (string rewriting) described in *The Science of Fractal Images* (edited by Pietgen and Saupe.) The basic algorithm has been expanded and modified extensively, and now supports the 3-D syntax of Laurens Lapre's LParser. In addition an expanded set of rules that allow looping and 3D formula mapping (lissajous curves) have been added. To enable the program to run at higher depth levels without extra megabytes of memory, an alternate method of string interpretation has been implemented, using recursion. String size is kept to a minimum, while the axiom is interpreted character by character.

For those who are not familiar with L-systems, a discussion of their basic features and how they differ from other turtle graphics is a good starting point. Lsystems have as their base an axiom, which is a string of character commands, or a single character. The character can be a command in itself, such as 'F' for forward, or it may be a production rule, which is a string of commands also. The axiom and production rules are created and edited with separate editors. When the axiom is 'run', the axiom is scanned for commands and/or rules. Recursion is used to substitute where necessary a production rule string for a single character, to the depth specified at runtime. In 0L-systems, quite long character strings can be built this way and then executed. Once the string is built, it is then stripped of non-commands and executed without displaying the commands on the screen. The size of the drawing object is thus established and it is then scaled and drawn on the screen. The first significant difference from other turtle graphics systems is now apparent: self-scaling. The object(s) will never overrun the screen boundaries. Objects can be displayed to the limits of the window, regardless of window size. Since object size is always relative to the window size, a scaling factor is used to create similar objects of different size. Other turtle systems would vary the length of an object's sides.

L-systems have a stack available to keep track of turtle position and variables whenever it is necessary to return to a particular point in the drawing. This is useful for producing tree structures and other branching patterns. Fractal Imaginator puts on its stack the turtle position, headings, line style, and color information.

The third significant difference between L-systems and other turtle graphics interpreters is in its handling of recursion. In L-systems, recursion is a built-in feature. You only need to

specify the level of recursion, and the lsystem does its string interpretation to that level. Fractal Imaginator uses recursion to call its own interpretation routines, and thus eliminates the need for lengthy strings to interpret. The price for the above built-in features is speed. Lsystems can never be as fast drawing a dragon curve, for instance, as a dedicated dragon-drawing program. On the other hand, fractal generators have trouble matching the speed of L-systems where branching patterns are involved. There is no need to recurse backwards to reach a branching node in L-systems: just pop the stack, and you're there. Fractal Imaginator is capable of rendering polygons in three dimensions, but cannot match the speed of a dedicated 3-D modeler. Hidden-line removal is done via a z-buffer, with simple ray-tracing to enhance the view. L-systems offer the user a chance to experiment with a very different approach to graphics/fractal production.

### Syntax for Fractal Imaginator's Lsystem (default and extended)

The default syntax follows Laurens Lapre's LParser implementation. The initial heading is (0,1,0), which will move the turtle up in the direction of the y-axis, when no rotation is applied to the axis (or a rotation of 0,0,0). All turning commands are three-dimensional. The following is extracted from Laurens Lapre's LParser package.

---

#### Turtle Orientation commands

---

+ turn left around up vector  
 +(x) turn x left around up vector  
 - turn right around up vector  
 -(x) turn x right around up vector  
 & pitch down around left vector  
 &(x) pitch x down around left vector  
 ^ pitch up around left vector  
 ^(x) pitch x up around left vector  
 < roll left (counter clockwise) around forward vector  
 <(x) roll x left around forward vector  
 > roll right (clockwise) around forward vector  
 >(x) roll x right around forward vector

---

#### Special Orientation commands

---

| turn 180 deg around up vector  
 % roll 180 deg around forward vector  
 \$ roll until horizontal  
 ~ turn/pitch/roll in a random direction  
 ~(x) " in a random direction with a maximum of x degrees  
 t correction for gravity with 0.2  
 t(x) correction for gravity with x

---

#### Movement commands

when { } active

---

```

F  move forward and draw full length  record vertex
F(x) move x forward and draw          record vertex
Z  move forward and draw half length  record vertex
Z(x) move x forward and draw          record vertex
f  move forward with full length      record vertex
f(x) move x forward                   record vertex
z  move forward with half length      record vertex
z(x) move x forward                   record vertex
g  move forward with full length      don't record vertex
g(x) move x forward                   don't record vertex
.  don't move                          record vertex

```

---

#### Structure commands

---

```

[  push current state
]  pop current state
{  start polygon shape
}  end polygon shape

```

---

#### Inc/Dec commands

---

```

"  increment length with 1.1
'  decrement length with 0.9
"(x) multiply length with x also '(x)
;  increment angle with 1.1
:  decrement angle with 0.9
:(x) multiply angle with x also :(x)
?  increment thickness with 1.4
!  decrement thickness with 0.7
?(x) multiply thickness with x also !(x)

```

---

#### Additional commands

---

```

c  increment color index
c(x) set color index to x

```

---

#### Extended command syntax:

r(n) -- repeat the execution of the following commands by the number specified by the following character or characters. The default is 1, or one repetition. For example: 'r(5)' would repeat a loop 5 times.

`rv` -- repeat the execution of the following commands until the current heading is reached again. Used to create turtle graphics subroutines that stop when the the total turning is a multiple of 360 degrees.

`rp` -- repeat the execution of the following commands until the current position(x,y,z) is reached again. Used for mapping lissajous curves.

`@` -- end of repeat loop. Repeat loops can be nested to a level of 10. You should begin and end a repeat loop inside of the same production rule or axiom, to avoid potential problems with the interpretation of that loop. Since recursion is used to interpret a production rule character by character, the '@' can only send the interpreter back to the beginning of that loop. Trying to get back to an axiom's repeat loop using a '@' from a production rule will trigger a syntax error.

`'s(n)'` -- number of sides in line. 0 sides is default for a tubular line. `s(3)` sets a wedge-shaped line, while `s(4)` sets a cubic or square line shape. Increasing the number of sides makes the line more rounded, but increases draw time. The sides command only works when the line width is greater than 1. (A narrower line can be rounded with fewer sides, to speed up drawing.)

3D curve(lissajous) commands:

`F(Ln)` -- draw segment of 3D curve, centered around current position.

`f(Ln),g(Ln)` -- move as segment of 3D curve, no drawing.

`Z(Ln)` -- draw half segment or n-length segment of 3D curve.

`z(Ln)` -- move as half-segment or n-length segment of 3D curve, no drawing.

`+(Ln)` -- rotates 3D curve's x and z (heading) vectors by n degrees.

`-(Ln)` -- rotates 3D curve's x and z (heading) vectors by -n degrees.

`&(Ln)` -- rotates 3D curve's x and y (heading) vectors by n degrees.

`^(Ln)` -- rotates 3D curve's x and y (heading) vectors by -n degrees.

`<(Ln)` -- rotates 3D curve's y and z (heading) vectors by n degrees.

`>(Ln)` -- rotates 3D curve's y and z (heading) vectors by -n degrees.

`b(n)` -- set boundary limit for 3D curve (default is 0). Used preceding a 'rp' command, to allow the repeat loop more flexibility in deciding when the beginning position has been reached. Some 3D curves require a loosening of the boundary for the rp loop to work. Higher boundary values make the rp loop more effective, though too high a value can result in premature bailout. To eliminate endless rp loops, a failsafe bailout is set at 50000 reps.

`i(n)` -- skip n segments in 3D curve (default 0.) Use to speed up drawing by decreasing segment resolution. Sharp angles may appear as disks rather than a continuous form when segments are skipped.

`l(n)` -- set starting angle for 3D curve (default is 0.) Using 'l' and `r(n)` different part of the 3D curve can be drawn. The 'rp' command automatically sets l to 0.

P -- set center position for 3D curve. Uses the current turtle position as a center point for 3D curves

x{formula} -- set x formula for 3D curve. Can use any trig function, plus the angles, 'x', 'y' and 'z' and the loop variable 'd'. The default x formula is "sin(x\*d)\*cos(y\*d)".

y{formula} -- set y formula for 3D curve. Can use any trig function, plus the angles, 'x', 'y' and 'z' and the loop variable 'd'. The default y formula is "sin(x\*d)\*sin(y\*d)"

w{formula} -- set z formula for 3D curve. Can use any trig function, plus the angles, 'x', 'y' and 'z' and the loop variable 'd'. The default z formula is "cos(x\*d)".

u(n) -- sets the x angle for 3D curve.

v(n) -- sets the y angle for 3D curve.

k(n) -- sets the z angle for 3D curve.

### Notes:

The color index is initialized at 2. Each index of the current palette is used by the ray-tracer as a separate palette, so up to 236 different objects can be modeled using up to 236 different colors. This works best when the index is a dark color. The color will be spread from dark to light when it is used during final ray-tracing.

The angles used in turning commands may be negative, as +(-25).

When F or a like drawing/movement command is followed by an argument, as in F(5), the turtle draws the absolute distance specified, in turtle units. This may be different from FFFFF, since the initial length of the draw vector is 100. An equivalent (and faster) draw command for FFFFF would therefore be F(500), assuming the length vector was not changed by one of the inc/dec commands. When the length vector of a line is decreased or increased, the width of the line is proportionately changed also (to retain compatibility with LParser files.)

Under the extended rule set, when line width is changed during drawing, Fractal Imaginator attempts to smooth out width changes by introducing a gradient, or gradual width change, while drawing. If two or more consecutive commands are introduced that change the line width before drawing, only the last width change will have this gradient.

Syntax errors and assignment errors that occur during runtime are displayed in a message box, and the run is halted. The message box contains the characters that caused the syntax error, to aid troubleshooting.

## 16.7 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

Burton, 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.

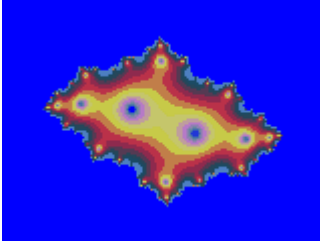
Wegner, Tim and Tyler, Bert: "Fractal Creations", Second Edition, Waite Group Press, Corte Madera, CA, 1993.

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

## 16.8 About Fractal Imaginator

### About Fractal Imaginator

>>>> Fractal Imaginator™ v3.064 ©2004, 2005 by Terry W. Gintz and Jules Ruis



Fractal Imaginator graphs formulas based on 4-D complex number planes. Fractal Imaginator currently supports the Mandelbrot set, Julia sets, and Phoenix curves, with millions of mapping variations. Also included are midpoint displacement routines to enhance 3D plots. 3D plot types include quaternion, hypernion and other quad types and octonions. 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, including the gamma and Laguerre functions.

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 the epsilon-cross method (Clifford A. Pickover), renormalization and IFS (Michael Barnsley).

Since the formula parser is an interpreter, with its inherent lack of speed, 180 'popular' and unusual formulas have been hard-coded to reduce graphing time by 40 to 60 percent (plus the FraSZle formula set too!) Several of the built-in formulas are capable of producing graphs that the program could not generate through the parser alone, such as applying the Mandelbrot set to Newton's method for solving quadratic equations. Also included in the built-in functions are the Quaternion and hypercomplex sets of four dimensions. Hypercomplex extensions (as described in Fractal Creations) have been incorporated into most of the Mandelbrot and Julia functions except for the Quaternion set (this is itself a special 4D version of the Mandelbrot set  $z^2+c$ .), and the real Newton formulas and formulas f3-f9 by Sylvie Gallet.

Fractal Imaginator requires a true-color video adapter for best results. It may work in 16-bit (high color), but this hasn't been tested thoroughly.

Memory requirements for Fractal Imaginator vary with the size of the drawing area Fractal Imaginator opens on, ranging from approximately 3 megabytes memory for a 640X480 area to 48 megabytes for a 2048X1536 area. Special routines have been added to reduce memory requirements for large bitmaps (up to 14400X10800) by writing these directly to a file instead of using a memory bitmap.

Acknowledgements: The Growth menu in Fractal Imaginator and associated variables are

based on the algorithms in Jules Ruis's BBM8 program. Many thanks to Paul Carlson for providing me his algorithms for 3D-like fractals, and allowing me to incorporate his ideas into Fractal Imaginator. Also, special thanks to Ron Barnett for his help in setting up the animation routines, to Earl Hinrichs for sharing his unique programming methods on the fractal art and programmer's lists, 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 3D insights. A special mention for Stig Pettersson, for his ground breaking work in cubic Mandelbrots, giving me clues to an historic fractal world I hardly knew existed. The multi-windowing interface in Fractal Imaginator is courtesy of that extraordinary and prolific fractal programmer, Steven C. Ferguson.

For a short history of programs leading to this one, see [Chronology](#).

## 16.8.1 Chronology

### Chronology

History of the programs:

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 3D 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 3D 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 -- Dofu 1.0 -- supports new Ferguson/Gintz plug-in spec.

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

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

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

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

April 1999 -- Split Dofu-Zon Elite into two programs: Fractal Zplot using built-in formulas and rendering methods, and Dofu-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 3D 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-3D 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.

October 2001 -- FraSZle 1.0, QuaSZ formula and algebra compatible version of Fractal Zplot

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 -- Pod ME 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.

# Index

## - B -

break: biomorph 54  
 break: biomorph off 54  
 break: decomposition 57  
 break: decomposition off 58  
 break: orbit traps 54  
 button: [ ] 19  
 button: ||||| 18  
 button: > 19  
 button: abort 13  
 button: batch 12  
 button: bmp 18  
 button: channel c1 16  
 button: channel c2 16  
 button: channel cj 16  
 button: channel es 15  
 button: channel J1 14  
 button: channel J2 15  
 button: channel LS 17  
 button: channel o1 17  
 button: channel o2 17  
 button: channel Q1 15  
 button: channel Q2 16  
 button: channel SB 15  
 button: color 12  
 button: draw 13  
 button: fvr 13  
 button: help 13  
 button: jpg 18  
 button: load 18  
 button: new 12  
 button: png 18  
 button: rend 13  
 button: save 17  
 button: size 12  
 button: undo 12  
 button: V 19  
 button: view 13

## - C -

color: blue edit box 44

color: blue slider 44  
 color: cancel button 43  
 color: copy button 42  
 color: divide by eight palette 65  
 color: divide by four palette 65  
 color: divide by one palette 65  
 color: divide by two palette 65  
 color: edit palette 40  
 color: green edit box 43  
 color: green slider 43  
 color: h/r button 42  
 color: map button 42  
 color: neg button 42  
 color: okay button 43  
 color: pixel 62  
 color: rand button 44  
 color: red edit box 43  
 color: red slider 43  
 color: reset button 43  
 color: reverse button 42  
 color: spread button 42  
 color: srb button 43  
 color: srg button 42  
 colorscaling: background 64  
 colorscaling: continuous potential 63  
 colorscaling: escape 62  
 colorscaling: graded palette 64  
 colorscaling: level 63  
 colorscaling: set only 64  
 colorscaling: use level curve 64

## - D -

demo: batch mode 71  
 demo: random coctonion 75  
 demo: random cubic julia 75  
 demo: random cubic mandelbrot 74  
 demo: random escher 73  
 demo: random julia 72  
 demo: random julia2 72  
 demo: random lsystem 75  
 demo: random newton 73  
 demo: random octonion 75  
 demo: random quaternion 73  
 demo: random quaternion2 74  
 demo: random render 71  
 demo: random stalks 73

**- E -**

edit: copy 31  
edit: copydata 32  
edit: cubic values 39  
edit: formula 32  
edit: fractal variables 35  
edit: lsystem axiom 37  
edit: lsystem rules 38  
edit: octonion parameters 40  
edit: parameters 36, 37  
edit: paste 32  
edit: pastedata 32  
edit: preferences 44  
edit: ray-tracing variables 40  
edit: size 40  
edit: text 44  
edit: undo 31  
exit 30

**- F -**

files: load jpeg 24  
files: load lsystem 26  
files: load palette 26  
files: load palettes 24  
files: load par 23  
files: load parameters 23  
files: load png 24  
files: managing 21, 22, 30  
files: save lsystem 26  
files: save lsystem to dxf 29  
files: save lsystem to obj 29  
files: save palette 26  
files: save palettes 25  
files: save par 24  
files: save parameters 24  
files: save q polygon 27, 28  
files: save quasz parameters 26  
files: set max vertices 29  
files: simplify mesh 27  
files: smooth 28  
files: write jpeg 25  
files: write png 25

**- H -**

help: about fractal imaginators 105  
help: bibliography 103  
help: built-in formulas 86  
help: channels 14  
help: chronology 106  
help: hot keys 84  
help: parser info 84  
help: remote 12  
help: tutorial 82

**- I -**

image: abort 48  
image: auto alert 47  
image: auto remote 47  
image: auto time 48  
image: clear 47  
image: clone 49  
image: continue draw 48  
image: dive 49  
image: draw 45  
image: draw lsystem 45  
image: mesh setup 30  
image: pilot 49  
image: redraw 47  
image: reset 50  
image: show picture 49  
image: new view on zoom 49

**- P -**

palettes: selecting 65  
pixel: phoenix 60  
pixel: solid guessing 61  
pixel: symmetry 61

**- R -**

render: atan coloring 59  
render: bof60 coloring 59  
render: coloring filter 58  
render: level curve 56  
render: orbit trap values 55  
render: potential coloring 60

render: texture scale 60

## - S -

status bar 67

## - T -

toolbar 66

type: 2d complexified quaternion 53

type: 2d cubic 53

type: 2d hypercomplex 53

type: 2d quaternion 53

type: complexified quaternion 53

type: cubic 52

type: hypernion 51

type: julia 51

type: lsystem 53

type: mandelbrot 51

type: quaternion 51

## - V -

video: avi object 70

video: avi wrl 70

video: close avi stream 69

video: open avi stream 69

video: view avi 70

video: write frames 69