

## Fractint for Windows (Winfract)

Copyright (C) 1990-92 The Stone Soup Group. Fractint for Windows may be freely copied and distributed, but may not be sold.

### Help Topics

[New Features in this Version](#)

[Bugs and Limitations in this Version](#)

[File Menu](#)

[Fractal Formula Selection](#)

[Image Sizing and Color selection](#)

[Coordinate Box options](#)

[Zooming in on an Image, Mandelbrot/Julia Toggling](#)

[Options and Doodads](#)

[Color-Cycling](#)

[Getting a Status Report](#)

[Distribution Policy and Procedures, Contacting the Authors](#)

[A list of Fractint Authors](#)

## New Features in this Version

Version **17.1** is just version 17.0 with a few bug-fixes and a few minor improvements that we managed to squeeze into the week that occurred between the two releases:

- Fixed a bug caused by two mis-compiled assembler modules that resulted in PCs without an FPU locking up when attempting to run some fractal types (mostly type=formula).
- Fixed the color-cycling hot-keys (right/left arrow, spacebar, ENTER) so that they once again work all of the time.
- Added Fractint-style color-cycling speed-control with the Up/Down arrow keys - Up-Arrow speeds up the cycling, Down-Arrow slows it down.
- Added new Fractint-style hot-keys that bring up the appropriate Winfract menus - the newly-added keys are T(Fractal Type), R(estore), S(ave), 3(D), O(verlay), P(rint), TAB (status), DEL (image size), X, Y, and Z (which all bring up the same "doodads" menu) and C(olor-cycling).
- The listboxes that display filenames now display the filenames in standard sorted order, rather than whatever passed for the default Windows ordering.
- The Winfract title-bar now displays the word "(calculating)" while a fractal calculation is in progress.

Version **17.0** is just a "quick-release" of Winfract synched with the fractal engine from Fractint's version 17.0. Our goal is to significantly improve Winfract's user interface for the next release, which means that we're likely to have a developers' test version that doesn't run at all for some time <grin>, so this "quick-release" seemed like a good idea. Other than the use of Fractint 17.0's fractal engine, Winfract 17.0 is just version 16.51 with a few bug-fixes (both courtesy of Mark Peterson):

- The Julibrot parameter prompts now display correctly.
- The "Save As" logic now correctly saves 16-color BMPs even if you're using a 256-color Video driver.

## Bugs and Limitations in this Version

(This section of the documentation has not been modified since version 16.0)

Fractint for Windows is (and will continue to be) a "port" of Fractint for DOS to the Windows environment, retaining the Fractint for DOS fractal and GIF engines, but replacing its front-end and graphics interface with a Windows engine. As such, its development will always "trail" that of Fractint for DOS, except where the Windows version adds functionality simply by virtue of the Windows interface.

There are four main causes behind the various bugs and limitations in this program. For future brevity, these causes are tagged FFD (problems related to the fact that much of the code is and will continue to be from Fractint For Dos), WLIM (limitations caused by Windows, or some Windows video drivers that aren't limitations under MS-DOS), NWP (a Novice Windows Programmer write the original program, and he often had no idea what he was doing), and NYI (Not Yet Implemented - hey, we're working on it!). FFD and WLIM limitations are probably permanent - hopefully NWP and NYI problems are less so.

- You can run only one "instance" of Fractint for Windows. FFD: Fractint for DOS is riddled with initialized FAR data ("char far myvalue = 0;"), and the Windows SDK silently but firmly tags any program containing code like that as a single-instance program.

- Palette support is limited. WLIM: The "stock" VGA driver supplied with Win 3.0 doesn't support palette-modification by applications at all (most third-party 256-color SuperVGA drivers do, though). "Plasma Clouds" look Godawful using anything less than a 256-color Windows driver. Color-cycling is limited to video drivers which support palette-manipulation (and, for 16-color drivers, color-cycling affects the background windows as well). Note that 16-bit and 24-bit "true color" Windows video drivers don't \*have\* a palette to manipulate, so we can't color-cycle them.

- Fractint for Windows is not as "background process" friendly as it should be when it updates the screen image - if it is updating a large image in its entirety, it can grab the machine for seconds at a time. NWP - we're making attempts to update the screen in periodic "chunks", but we need to improve some more in this area.

- You cannot "abort" the File Print operations (File Save and File Open can be aborted simply by selecting another option). NWP, NYI: We're performing the entire print operation with a single 'StretchDIBits()' call. For now, just fire up a print operation, sit back, and wait for the results. Also, don't expect to run a background process (like a file transfer) while Fractint is printing. Finally, the File Print option only works on images up to 800x600 on my 386 with 6MB of RAM and LaserJet II, for some reason.

- There are lots of places where this program doesn't thoroughly check that Windows-specific procedures that it invokes ran correctly, and it crashes on occasion because of that. NWP, NYI: Basically, if we haven't hit a particular bug yet, we haven't adjusted the program to look for it.

- Lots of little doodads aren't in this version yet. NYI.

- There are \*far\* too many places in this document that say "Read Fractint-for-DOS's FRACTINT.DOC file for an explanation". NYI.

## File Menu

(This section of the documentation has not been modified since 16.51)

File **Open** loads either Fractint-for-DOS-style or "generic" GIF files. Note that the correct palette isn't completely displayed on the screen. Windows "reserves" the first twenty palettes for its own use, and "adjusts" Fractint's images accordingly - and for 16-color drivers like the VGA driver supplied with Windows that don't support palette-manipulation by applications programs, cheerfully ignores our attempts at palette-manipulation. If the restored image is resumable (a feature added with version 14.0 of FFD and version 1.2 of FFW), the image will resume calculating as soon as it is loaded.

File **Save** saves your current image (by default, as a GIF (version 89a) file, compatible with Fractint-for-DOS, your favorite DOS GIF viewers, and (of course) the "File Open" option, but optionally as a GIF87a or a BMP file). Again, the palette you see on the screen may not be the one that the program is "using" (and gets saved to the disk file). If the saved image is resumable (a feature added with version 14.0 of FFD and version 1.2 of FFW), the image will resume calculating as soon as it is finished saving. Note that Winfract can only open GIF89a-format images as fractal images, and cannot open BMP-format images at all.

File **3D Open** and File **3D Overlay** load in your image using "3D" transformations. The 3D Open option clears your image first and then loads the new one, while the Overlay option leaves your original image intact and adds the new image over it. For a full explanation of the zillion or so 3D parameters, read Fractint-for-DOS's FRACTINT.DOC file.

File **Print** does the best job it can sending the image to your printer. The program does not "dither" for black-and-white printers - it just "candy stripes" adjacent colors the same way that Fractint-for-DOS does.

**Read Color-Map** and **Write Color-Map** load and save external color-maps in the same manner as Fractint-for-DOS (within Windows' limitations).

File **Copy to Clipboard** copies the currently-displayed image to the Windows clipboard (in "Device-Independent Bitmap" format) where your other Windows programs can collect it. Note that, for 256-color video drivers and 256-color images, the colors in the clipboard won't visibly match the colors in Winfract's window until you change your "focus" to the clipboard's window. That's because Winfract has warned the Clipboard that it is reserving the right to color-cycle its colors, so the Clipboard is just using Windows' default 20 colors to display its colors while Winfract has control of the screen.

GIF and "Graphics Interchange Format" are trademarks of CompuServe Incorporated, an H&R Block Company.

## Fractal Formula Selection

Using the Options **Formula** option, you can select any of over 80 fractal types (every fractal type that is available in version 17.0 of Fractint-for-DOS). After selecting a fractal type, a dialogue box pops up and prompts you for any formula parameters and the screen corners (all with reasonable default values).

A brief list of Fractal types (and their formulas) included in this release:

### barnsleyj1

$z(0) = \text{pixel};$   
 $z(n+1) = (z-1)*c$  if  $\text{real}(z) \geq 0$ , else  
 $z(n+1) = (z+1)*\text{modulus}(c)/c$   
Two parameters: real and imaginary parts of  $c$

### barnsleyj2

$z(0) = \text{pixel};$   
if  $\text{real}(z(n)) * \text{imag}(c) + \text{real}(c) * \text{imag}(z(n)) \geq 0$   
 $z(n+1) = (z(n)-1)*c$   
else  
 $z(n+1) = (z(n)+1)*c$   
Two parameters: real and imaginary parts of  $c$

### barnsleyj3

$z(0) = \text{pixel};$   
if  $\text{real}(z(n) > 0$  then  $z(n+1) = (\text{real}(z(n))^2 - \text{imag}(z(n))^2 - 1)$   
 $+ i * (2*\text{real}(z(n)) * \text{imag}(z(n)))$  else  
 $z(n+1) = (\text{real}(z(n))^2 - \text{imag}(z(n))^2 - 1 + \text{real}(c) * \text{real}(z(n))$   
 $+ i * (2*\text{real}(z(n)) * \text{imag}(z(n)) + \text{imag}(c) * \text{real}(z(n)))$   
Two parameters: real and imaginary parts of  $c$ .

### barnsleym1

$z(0) = c = \text{pixel};$   
if  $\text{real}(z) \geq 0$  then  
 $z(n+1) = (z-1)*c$   
else  
 $z(n+1) = (z+1)*\text{modulus}(c)/c$ .  
Parameters are perturbations of  $z(0)$

### barnsleym2

$z(0) = c = \text{pixel};$   
if  $\text{real}(z)*\text{imag}(c) + \text{real}(c)*\text{imag}(z) \geq 0$   
 $z(n+1) = (z-1)*c$   
else  
 $z(n+1) = (z+1)*c$   
Parameters are perturbations of  $z(0)$

### barnsleym3

$z(0) = c = \text{pixel};$   
if  $\text{real}(z(n) > 0$  then  $z(n+1) = (\text{real}(z(n))^2 - \text{imag}(z(n))^2 - 1)$   
 $+ i * (2*\text{real}(z(n)) * \text{imag}(z(n)))$  else  
 $z(n+1) = (\text{real}(z(n))^2 - \text{imag}(z(n))^2 - 1 + \text{real}(c) * \text{real}(z(n))$   
 $+ i * (2*\text{real}(z(n)) * \text{imag}(z(n)) + \text{imag}(c) * \text{real}(z(n)))$   
Parameters are perturbations of  $z(0)$

### bifurcation

Pictorial representation of a population growth model.  
Let  $P$  = new population,  $p$  = oldpopulation,  $r$  = growth rate  
The model is:  $P = p + r*p*(1-p)$ .  
Two parameters: Filter Cycles and Seed Population.

### **bif+sinpi**

Bifurcation variation: model is:  $P = p + r*\sin(\pi*p)$ .  
Two parameters: Filter Cycles and Seed Population.

### **bif=sinpi**

Bifurcation variation: model is:  $P = r*\sin(\pi*p)$ .  
Two parameters: Filter Cycles and Seed Population.

### **biflambda**

Bifurcation variation: model is:  $P = r*p*(1-p)P$ .  
Two parameters: Filter Cycles and Seed Population.

### **bifstewart**

Bifurcation variation: model is:  $P = (r*p*p) - 1$ .  
Two parameters: Filter Cycles and Seed Population.

### **Circle**

Circle pattern by John Connett  
 $x + iy = \text{pixel}$   
 $z = a*(x^2 + y^2)$   
 $c = \text{integer part of } z$   
 $\text{color} = c \text{ modulo}(\text{number of colors})$

### **cmplxmarksjul**

A generalization of the marksjulia fractal.  
 $z(0) = \text{pixel}$ ;  
 $z(n+1) = (c^{\text{exp}})*z(n) + c$ .  
Four parameters: real and imaginary parts of  $c$  and  $\text{exp}$ .

### **cmplxmarksmand**

A generalization of the marksmandel fractal.  
 $z(0) = c = \text{pixel}$ ;  
 $z(n+1) = (c^{\text{exp}})*z(n) + c$ .  
Four parameters: real and imaginary parts of  
perturbation of  $z(0)$  and  $\text{exp}$ .

### **complexnewton, complexbasin**

Newton fractal types extended to complex degrees. Complexnewton  
colors pixels according to the number of iterations required to  
escape to a root. Complexbasin colors pixels according to which  
root captures the orbit. The equation is based on the newton  
formula for solving the equation  $z^p = r$   
 $z(0) = \text{pixel}$ ;  
 $z(n+1) = ((p - 1) * z(n)^p + r)/(p * z(n)^{(p - 1)})$ .  
Four parameters: real & imaginary parts of degree  $p$  and root  $r$

### **diffusion**

Diffusion Limited Aggregation. Randomly moving points  
accumulate. One parameter: border width (default 10)

### **fn+fn(pix)**

$c = z(0) = \text{pixel};$   
 $z(n+1) = \text{fn1}(z) + p * \text{fn2}(c)$   
Six parameters: real and imaginary parts of the perturbation of  $z(0)$  and factor  $p$ , and the functions  $\text{fn1}$ , and  $\text{fn2}$ .

#### **fn(z\*z)**

$z(0) = \text{pixel};$   
 $z(n+1) = \text{fn}(z(n)*z(n))$   
One parameter: the function  $\text{fn}$ .

#### **fn\*fn**

$z(0) = \text{pixel}; z(n+1) = \text{fn1}(n)*\text{fn2}(n)$   
Two parameters: the functions  $\text{fn1}$  and  $\text{fn2}$ .

#### **fn\*z+z**

$z(0) = \text{pixel}; z(n+1) = p1*\text{fn}(z(n))*z(n) + p2*z(n)$   
Six parameters: the real and imaginary components of  $p1$  and  $p2$ , and the functions  $\text{fn1}$  and  $\text{fn2}$ .

#### **fn+fn**

$z(0) = \text{pixel};$   
 $z(n+1) = p1*\text{fn1}(z(n))+p2*\text{fn2}(z(n))$   
Six parameters: The real and imaginary components of  $p1$  and  $p2$ , and the functions  $\text{fn1}$  and  $\text{fn2}$ .

#### **formula**

Formula interpreter - write your own formulas as text files!

#### **gingerbread**

Orbit in two dimensions defined by:  
 $x(n+1) = 1 - y(n) + |x(n)|$   
 $y(n+1) = x(n)$   
Two parameters: initial values of  $x(0)$  and  $y(0)$ .

#### **henon**

Orbit in two dimensions defined by:  
 $x(n+1) = 1 + y(n) - a*x(n)*x(n)$   
 $y(n+1) = b*x(n)$   
Two parameters:  $a$  and  $b$

#### **Hopalong**

Hopalong attractor by Barry Martin - orbit in two dimensions.  
 $z(0) = y(0) = 0;$   
 $x(n+1) = y(n) - \text{sign}(x(n))*\text{sqrt}(\text{abs}(b*x(n)-c))$   
 $y(n+1) = a - x(n)$   
Parameters are  $a$ ,  $b$ , and  $c$ .

#### **IFS**

Barnsley IFS fractals.

#### **julfn+exp**

A generalized Clifford Pickover fractal.  
 $z(0) = \text{pixel};$   
 $z(n+1) = \text{fn}(z(n)) + e^z(n) + c.$   
Three parameters: real & imaginary parts of  $c$ , and  $\text{fn}$

**julfn+zsqr**

$$z(0) = \text{pixel};$$

$$z(n+1) = \text{fn}(z(n)) + z(n)^2 + c$$

Three parameters: real & imaginary parts of c, and fn

**julia**

Classic Julia set fractal.

$$z(0) = \text{pixel}; z(n+1) = z(n)^2 + c.$$

Two parameters: real and imaginary parts of c.

**julia4**

Fourth-power Julia set fractals, a special case of julzpower kept for speed.

$$z(0) = \text{pixel};$$

$$z(n+1) = z(n)^4 + c.$$

Two parameters: real and imaginary parts of c.

**julibrot**

'Julibrot' 4-dimensional fractals.

**julzpower**

$$z(0) = \text{pixel};$$

$$z(n+1) = z(n)^m + c.$$

Three parameters: real & imaginary parts of c, exponent m

**julzpw**

$$z(0) = \text{pixel};$$

$$z(n+1) = z(n)^z(n) + z(n)^m + c.$$

Three parameters: real & imaginary parts of c, exponent m

**kamtorus, kamtorus3d**

Series of orbits superimposed.

3d version has 'orbit' the z dimension.

$$x(0) = y(0) = \text{orbit}/3;$$

$$x(n+1) = x(n)*\cos(a) + (x(n)*x(n)-y(n))*\sin(a)$$

$$y(n+1) = x(n)*\sin(a) - (x(n)*x(n)-y(n))*\cos(a)$$

After each orbit, 'orbit' is incremented by a step size.

Parameters: a, step size, stop value for 'orbit', and points per orbit.

**lambda**

Classic Lambda fractal. 'Julia' variant of Mandellambda.

$$z(0) = \text{pixel};$$

$$z(n+1) = \text{lambda}*z(n)*(1 - z(n)^2).$$

Two parameters: real and imaginary parts of lambda.

**lambdafn**

$$z(0) = \text{pixel};$$

$$z(n+1) = \text{lambda} * \text{fn}(z(n)).$$

Three parameters: real, imag portions of lambda, and fn

**lorenz, lorenz3d**

Lorenz two lobe attractor - orbit in three dimensions.

In 2d the x and y components are projected to form the image.

$$z(0) = y(0) = z(0) = 1;$$

$$x(n+1) = x(n) + (-a*x(n)*dt) + ( \quad a*y(n)*dt)$$



$$y(n+1) = y(n) + (b*x(n)*dt) - (y(n)*dt) - (z(n)*x(n)*dt)$$

$$z(n+1) = z(n) + (-c*z(n)*dt) + (x(n)*y(n)*dt)$$

Parameters are dt, a, b, and c.

### lorenz3d1

Lorenz one lobe attractor - orbit in three dimensions.  
 The original formulas were developed by Rick Miranda and Emily Stone.  
 $z(0) = y(0) = z(0) = 1$ ;  $norm = \sqrt{x(n)^2 + y(n)^2}$   
 $x(n+1) = x(n) + (-a*dt-dt)*x(n) + (a*dt-b*dt)*y(n)$   
 $+ (dt-a*dt)*norm + y(n)*dt*z(n)$   
 $y(n+1) = y(n) + (b*dt-a*dt)*x(n) - (a*dt+dt)*y(n)$   
 $+ (b*dt+a*dt)*norm - x(n)*dt*z(n) - norm*z(n)*dt$   
 $z(n+1) = z(n) + (y(n)*dt/2) - c*dt*z(n)$   
 Parameters are dt, a, b, and c.

### lorenz3d3

Lorenz three lobe attractor - orbit in three dimensions.  
 The original formulas were developed by Rick Miranda and Emily Stone.  
 $z(0) = y(0) = z(0) = 1$ ;  $norm = \sqrt{x(n)^2 + y(n)^2}$   
 $x(n+1) = x(n) + (-a*dt+dt)*x(n) + (a*dt-b*dt+z(n)*dt)*y(n)/3$   
 $+ ((dt-a*dt)*(x(n)^2-y(n)^2)$   
 $+ 2*(b*dt+a*dt-z(n)*dt)*x(n)*y(n))/(3*norm)$   
 $y(n+1) = y(n) + ((b*dt-a*dt-z(n)*dt)*x(n) - (a*dt+dt)*y(n))/3$   
 $+ (2*(a*dt-dt)*x(n)*y(n)$   
 $+ (b*dt+a*dt-z(n)*dt)*(x(n)^2-y(n)^2))/(3*norm)$   
 $z(n+1) = z(n) + (3*x(n)*dt*x(n)*y(n)-y(n)*dt*y(n)^2)/2 - c*dt*z(n)$   
 Parameters are dt, a, b, and c.

### lorenz3d4

Lorenz four lobe attractor - orbit in three dimensions.  
 The original formulas were developed by Rick Miranda and Emily Stone.  
 $z(0) = y(0) = z(0) = 1$ ;  
 $x(n+1) = x(n) + (-a*dt*x(n)^3$   
 $+ (2*a*dt+b*dt-z(n)*dt)*x(n)^2*y(n) + (a*dt-2*dt)*x(n)*y(n)^2$   
 $+ (z(n)*dt-b*dt)*y(n)^3) / (2 * (x(n)^2+y(n)^2))$   
 $y(n+1) = y(n) + ((b*dt-z(n)*dt)*x(n)^3 + (a*dt-2*dt)*x(n)^2*y(n)$   
 $+ (-2*a*dt-b*dt+z(n)*dt)*x(n)*y(n)^2$   
 $- a*dt*y(n)^3) / (2 * (x(n)^2+y(n)^2))$   
 $z(n+1) = z(n) + (2*x(n)*dt*x(n)^2*y(n) - 2*x(n)*dt*y(n)^3 - c*dt*z(n))$   
 Parameters are dt, a, b, and c.

### Isystem

Using a turtle-graphics control language and starting with an initial axiom string, carries out string substitutions the specified number of times (the order), and plots the resulting.

### lyapunov

Derived from the Bifurcation fractal, the Lyapunov plots the Lyapunov Exponent for a population model where the Growth parameter varies between two values in a periodic manner.

### magnet1j

$$z(0) = pixel;$$

$$z(n+1) = \left| \frac{z(n)^2 + (c-1)}{2*z(n) + (c-2)} \right|^2$$

Parameters: the real and imaginary parts of c

### magnet1m

$$z(0) = 0; c = \text{pixel};$$
$$z(n+1) = \sqrt{\frac{z(n)^2 + (c-1)}{2z(n) + (c-2)}}$$

Parameters: the real & imaginary parts of perturbation of z(0)

### magnet2j

$$z(0) = \text{pixel};$$
$$z(n+1) = \sqrt{\frac{z(n)^3 + 3(C-1)z(n) + (C-1)(C-2)}{3z(n)^2 + 3(C-2)z(n) + (C-1)(C-2) - 1}}$$

Parameters: the real and imaginary parts of c

### magnet2m

$$z(0) = 0; c = \text{pixel};$$
$$z(n+1) = \sqrt{\frac{z(n)^3 + 3(C-1)z(n) + (C-1)(C-2)}{3z(n)^2 + 3(C-2)z(n) + (C-1)(C-2) - 1}}$$

Parameters: the real and imaginary parts of perturbation of z(0)

### mandel

Classic Mandelbrot set fractal.

$$z(0) = c = \text{pixel};$$
$$z(n+1) = z(n)^2 + c.$$

Two parameters: real & imaginary perturbations of z(0)

### mandel4

Special case of mandelzpower kept for speed.

$$z(0) = c = \text{pixel};$$
$$z(n+1) = z(n)^4 + c.$$

Parameters: real & imaginary perturbations of z(0)

### mandelfn

$$z(0) = c = \text{pixel};$$
$$z(n+1) = c * \text{fn}(z(n)).$$

Parameters: real & imaginary perturbations of z(0), and fn

### Martin

Attractor fractal by Barry Martin - orbit in two dimensions.

$$z(0) = y(0) = 0;$$
$$x(n+1) = y(n) - \sin(x(n))$$
$$y(n+1) = a - x(n)$$

Parameter is a (try a value near pi)

### mandellambda

$$z(0) = .5; \text{lambda} = \text{pixel};$$
$$z(n+1) = \text{lambda} * z(n) * (1 - z(n)^2).$$

Parameters: real & imaginary perturbations of z(0)

### manfn+exp

'Mandelbrot-Equivalent' for the julfn+exp fractal.

$$z(0) = c = \text{pixel};$$
$$z(n+1) = \text{fn}(z(n)) + e^{z(n)} + C.$$

Parameters: real & imaginary perturbations of  $z(0)$ , and  $fn$

#### **manfn+zsqr**

'Mandelbrot-Equivalent' for the Julfn+zsqr fractal.

$$z(0) = c = \text{pixel};$$

$$z(n+1) = fn(z(n)) + z(n)^2 + c.$$

Parameters: real & imaginary perturbations of  $z(0)$ , and  $fn$

#### **manowar**

$$c = z1(0) = z(0) = \text{pixel};$$

$$z(n+1) = z(n)^2 + z1(n) + c;$$

$$z1(n+1) = z(n);$$

Parameters: real & imaginary perturbations of  $z(0)$

#### **manowarj**

$$z1(0) = z(0) = \text{pixel};$$

$$z(n+1) = z(n)^2 + z1(n) + c;$$

$$z1(n+1) = z(n);$$

Parameters: real & imaginary perturbations of  $z(0)$

#### **manzpower**

'Mandelbrot-Equivalent' for julzpower.

$$z(0) = c = \text{pixel};$$

$$z(n+1) = z(n)^{\text{exp}} + c; \text{ try exp} = e = 2.71828\dots$$

Parameters: real & imaginary perturbations of  $z(0)$ , real & imaginary parts of exponent  $exp$ .

#### **manzppwr**

'Mandelbrot-Equivalent' for the julzppwr fractal.

$$z(0) = c = \text{pixel}$$

$$z(n+1) = z(n)^{z(n)} + z(n)^{\text{exp}} + C.$$

Parameters: real & imaginary perturbations of  $z(0)$ , and exponent

#### **marksjulia**

A variant of the julia-lambda fractal.

$$z(0) = \text{pixel};$$

$$z(n+1) = (c^{\text{exp}}) * z(n) + c.$$

Parameters: real & imaginary parts of  $c$ , and exponent

#### **marksmandel**

A variant of the mandel-lambda fractal.

$$z(0) = c = \text{pixel};$$

$$z(n+1) = (c^{\text{exp}}) * z(n) + c.$$

Parameters: real & imaginary perturbations of  $z(0)$ , and exponent

#### **marksmandelpwr**

The marksmandelpwr formula type generalized (it previously had  $fn=sqr$  hard coded).

$$z(0) = \text{pixel}, c = z(0) ^ (z(0) - 1):$$

$$z(n+1) = c * fn(z(n)) + \text{pixel},$$

Parameters: real and imaginary perturbations of  $z(0)$ , and  $fn$

#### **newtbasin**

Based on the Newton formula for finding the roots of  $z^p - 1$ .

Pixels are colored according to which root captures the orbit.

$$z(0) = \text{pixel};$$

$$z(n+1) = ((p-1)*z(n)^p + 1)/(p*z(n)^{(p-1)}).$$

Two parameters: the polynomial degree  $p$ , and a flag to turn on color stripes to show alternate iterations.

### newton

Based on the Newton formula for finding the roots of  $z^p - 1$ . Pixels are colored according to the iteration when the orbit is captured by a root.

$$z(0) = \text{pixel};$$

$$z(n+1) = ((p-1)*z(n)^p + 1)/(p*z(n)^{(p-1)}).$$

One parameter: the polynomial degree  $p$ .

### pickover

Orbit in three dimensions defined by:

$$x(n+1) = \sin(a*y(n)) - z(n)*\cos(b*x(n))$$

$$y(n+1) = z(n)*\sin(c*x(n)) - \cos(d*y(n))$$

$$z(n+1) = \sin(x(n))$$

Parameters:  $a$ ,  $b$ ,  $c$ , and  $d$ .

### plasma

Random, cloud-like formations. Requires 4 or more colors.

A recursive algorithm repeatedly subdivides the screen and colors pixels according to an average of surrounding pixels and a random color, less random as the grid size decreases.

Three parameters: 'graininess' (.5 to 50, default = 2), old/new algorithm, seed value used.

### popcorn

The orbits in two dimensions defined by:

$$x(0) = \text{apixel}, y(0) = \text{ypixel};$$

$$x(n+1) = x(n) - h*\sin(y(n) + \tan(3*y(n)))$$

$$y(n+1) = y(n) - h*\sin(x(n) + \tan(3*x(n)))$$

are plotted for each screen pixel and superimposed.

One parameter: step size  $h$ .

### popcornjul

Conventional Julia using the popcorn formula:

$$x(0) = \text{apixel}, y(0) = \text{ypixel};$$

$$x(n+1) = x(n) - h*\sin(y(n) + \tan(3*y(n)))$$

$$y(n+1) = y(n) - h*\sin(x(n) + \tan(3*x(n)))$$

One parameter: step size  $h$ .

### rossler3D

Orbit in three dimensions defined by:

$$x(0) = y(0) = z(0) = 1;$$

$$x(n+1) = x(n) - y(n)*dt - z(n)*dt$$

$$y(n+1) = y(n) + x(n)*dt + a*y(n)*dt$$

$$z(n+1) = z(n) + b*dt + x(n)*z(n)*dt - c*z(n)*dt$$

Parameters are  $dt$ ,  $a$ ,  $b$ , and  $c$ .

### sierpinski

Sierpinski gasket - Julia set producing a 'Swiss cheese triangle'

$$z(n+1) = (2*x, 2*y-1) \text{ if } y > .5;$$

$$\text{else } (2*x-1, 2*y) \text{ if } x > .5;$$

$$\text{else } (2*x, 2*y)$$

No parameters.

**spider**

$c(0) = z(0) = \text{pixel};$   
 $z(n+1) = z(n)^2 + c(n);$   
 $c(n+1) = c(n)/2 + z(n+1)$

Parameters: real & imaginary perturbation of  $z(0)$

**sqr(1/fn)**

$z(0) = \text{pixel};$   
 $z(n+1) = (1/fn(z(n)))^2$

One parameter: the function  $fn$ .

**sqr(fn)**

$z(0) = \text{pixel};$   
 $z(n+1) = fn(z(n))^2$

One parameter: the function  $fn$ .

**test**

'test' point letting us (and you!) easily add fractal types via the `c` module `testpt.c`. Default set up is a mandelbrot fractal. Four parameters: user hooks (not used by default `testpt.c`).

**tetrate**

$z(0) = c = \text{pixel};$   
 $z(n+1) = c^z(n)$

Parameters: real & imaginary perturbation of  $z(0)$

**tim's\_error**

A serendipitous coding error in `marksmandelpwr` brings to life an ancient pterodactyl! (Try setting  $fn$  to `sqr`.)

$z(0) = \text{pixel}, c = z(0) ^ (z(0) - 1);$   
 $\text{tmp} = fn(z(n))$   
 $\text{real}(\text{tmp}) = \text{real}(\text{tmp}) * \text{real}(c) - \text{imag}(\text{tmp}) * \text{imag}(c);$   
 $\text{imag}(\text{tmp}) = \text{real}(\text{tmp}) * \text{imag}(c) - \text{imag}(\text{tmp}) * \text{real}(c);$   
 $z(n+1) = \text{tmp} + \text{pixel};$

Parameters: real & imaginary perturbations of  $z(0)$  and function  $fn$

**unity**

$z(0) = \text{pixel};$   
 $x = \text{real}(z(n)), y = \text{imag}(z(n))$   
 $\text{One} = x^2 + y^2;$   
 $y = (2 - \text{One}) * x;$   
 $x = (2 - \text{One}) * y;$   
 $z(n+1) = x + i*y$

No parameters.

## **Image Sizing and Color selection**

(This section of the documentation has not been modified since version 16.0)

Using the Options **Image Size** option, you can select either a pre-defined image size, or make up your own (up to an internal limit of 2048x2048) and select a two-color, 16-color, or 256-color image. Those of you with 24-bit color video cards will have to hold yourself back to 256-color images for the moment - sorry.

If you have selected the "Window Sizing" option, your image window will automatically re-size to the image size you have just selected.

If your image size is larger than your window, you can use the scroll bars in the window to scroll around your image.

## Zooming in on an Image, Mandelbrot/Julia Toggling

(This section of the documentation has not been modified since version 16.51)

With this release of Winfract, there are two different methods of generating and using "Zoom-Boxes".

If you do not have Mark Peterson's new "Zoom-Box" option enabled, Winfract's zoom box is activated in the same fashion as it was in earlier releases:

To zoom in on a fractal image, just move your mouse pointer to the upper-left corner of the "zoom-box" area, push and hold down the left mouse-button, move the mouse pointer to the lower-right corner of the "zoom-box" area, and let go of the mouse button. The program will immediately begin a new image using the "zoom-box" coordinates you selected.

As you are moving your mouse with the left-button down, the "zoom-box" area will be displayed as an X-ORed rectangle. If you change your mind about zooming as you are moving the mouse around ("No, No, Not \*There\*!"), just make one of the rectangular coordinates less than five pixels across - the program will "forget" about the zoom-box if you let go of the left mouse button and that is the case.

If the aspect ratio of your "zoom-box" is reasonably close to that of the full image size (within 33 percent or so), the program adjusts the "zoom-box" to match the aspect ratio exactly before performing the zoom.

If you have enabled Mark Peterson's new "Zoom-Box" option, a vertical Zoom-Box scroll-bar is displayed in addition to your fractal image. The Zoom-Box defaults to the mid-position, in which your Zoom-Box exactly covers the area of your fractal image. Moving the scroll-button up shrinks your zoom-box (and makes it visible on the fractal image). You can then move the zoom-box around by moving your cursor inside the zoom-box and then holding down your left mouse-button and moving the cursor and zoom-box as a single unit. Double-clicking the left mouse button will cause your image to be redrawn using your current zoom-box coordinates.

You can also "Zoom Out" by moving the zoom-box scroll bar below the midpoint. When you do this, the zoom-box that is displayed is actually showing you where (and how small) your currently displayed image would be if you double-clicked the left mouse button at that point. Moving the zoom-box and double-clicking to perform the zoom is done the same way as when you are "zooming in".

You can switch from a "Mandelbrot Set" image to its "Julia Set" at the location of the mouse cursor by clicking on the right mouse button. When the corresponding "Julia Set" image is on the screen, click the right mouse button again to get the "Mandelbrot Set" image back. The terms "Mandelbrot Set" and "Julia Set" are in quotes because many fractal types (mandel4 and julia4, for instance) have this "Mandelbrot Set"/"Julia Set" relationship.

## **Options and Doodads**

(This section of the documentation has not been modified since version 16.0)

You can select and of a number of options and doodads using the "Options Options-and-Doodads" menu. The list of doodads currently includes various algorithms (single/dual pass, solid-guessing, boundary tracing), biomorphs, decomposition, inside color, outside color, maximum iterations, etc - all flagrantly stolen from Fractint-for-DOS.

Once you select any of these options, the program will begin re-drawing your image with the new options in effect.



## Color Cycling

You start and stop Color-cycling mode using the Color-Cycling menu. This menu also lets you determine the direction of the cycling (the designations "in" and "out" were arbitrarily picked based on how the effect looked on a single Mandelbrot image), whether to just rotate the existing colors or generate new ones randomly, and (for random color-generation) whether the colors are to change with a low, medium, or high frequency.

You can also use the following **"Hot-keys"** for color-cycling

- spacebar - toggles color-cycling on and off

- left, right arrows - turn on color-cycling and set the "direction"

- up, down arrows - speed up or slow down the color-cycling speed

- ENTER - initiate random color-cycling with all new colors

(uhh, note that Windows takes a second or so to "fire up" color-cycling, so be patient when you press one of these keys. Breathe in, breathe out, ...)

Sorry, but at the current time color-cycling is restricted to display devices whose Windows drivers are capable of palette manipulation - and the "stock" VGA driver distributed with Win 3.0 doesn't have it! Also, color-cycling affects the background windows as well unless your video driver is capable of displaying 256 or more simultaneous colors. If Windows finds itself running with a 16-color video driver, it color-cycles only when its window has the input focus, and returns to the "stock" palette values when it loses this focus.

## Getting a Status Report

(This section of the documentation has not been modified since version 16.0)

You can find out the status of your current image (which formula you are using, its parameters and screen corners, and - most important - whether or not it's still being generated or has finished) by selecting the **Status** option from the Options menu. A "Pup-up" message will tell you about the status of the current image. If the image is still being generated, the program will continue with it when you press the "OK" button.

## Distribution Policy and Procedures, Contacting the Authors

The Stone Soup Group is a loosely associated group of fanatic programmers. See Fractint-for-DOS's FRACTINT.DOC for the story behind the 'Stone Soup' name. Fractint for Windows, like Fractint-for-DOS, is freeware, and our "contribution policy" is the same: **Don't want money. Got money. Want admiration.**

### Conditions on use:

**Fractint for Windows, like Fractint-for-DOS, may be freely copied and distributed but may not be sold** (a nominal distribution fee may be charged for media and handling by freeware and shareware distributors.) Winfract may be used personally or in a business - if you can do your job better by using Winfract, or using images from it, that's great! It may not be given away with commercial products without explicit permission from the Stone Soup Group.

There is no warranty of Winfract's suitability for any purpose, nor any acceptance of liability, express or implied.

Virtually all of the Fractint and Winfract authors can be found on the **Compuserve** network in the **COMART** ('COMputer ART') forum in S 15 ('Fractals'). Fractint development occurs in the COMART forum - most of the authors have never met except there. New members are always welcome!

New versions of Fractint-for-DOS and Fractint-for-Windows are uploaded to the Compuserve network, and make their way to other systems from that point. The latest version of Fractint-for-Windows can be found in the "Fractals" library of COMART as **WINFRA.ZIP** (Executable/Docs) and **WINSRC.ZIP** (Complete Source). If you're not a Compuserve subscriber, but wish to get more information about Compuserve and its graphics forums, feel free to call their 800 number (800-848-8199) and ask for operator number 229.

If you don't have access to Compuserve, many other sites tend to carry these files shortly after their initial release (although sometimes using different naming conventions). In particular...

If you speak Internet and FTP, SIMTEL20 and its various mirror sites tend to carry new versions of Winfract shortly after they are released. Look in the PD:<MSDOS.GRAPHICS> and PD:<MSDOS.WINDOWS3> directories for these files .

Your favorite local BBS may carry these files as well (although perhaps not the latest versions) using naming conventions like WINFRA\*.ZIP. One BBS that *\*does\** carry the latest version is the "Ideal Studies BBS" (508)757-1806, 1200/2400/9600HST. Peter Longo is the SYSOP and a true fractal fanatic. There is a very short registration, and thereafter the entire board is open to callers on the first call.

Many Shareware/Freeware library services will ship you diskettes containing the latest versions of Fractint for a nominal fee that basically covers their cost of packaging and a small profit that we don't mind them making. One in particular is the Public (Software) Library, PO Box 35705, Houston, TX 77235-5705, USA. Their phone number is 800-242-4775 (outside the US, dial 713-524-6398). Ask for item #9112 for five 5.25" disks, #9113 for three 3.5" disks. Cost is \$6.99 plus \$4 S&H in the U.S./Canada, \$11 S&H overseas.

...

(What's the latest version? Well, THIS one was, way back when we uploaded it!)

## A list of Fractint Authors

Fractint for Windows was originally ported from Fractint-for-DOS by Bert Tyler. This is the first Windows program that Bert ever wrote, which goes a long way towards explaining a lot of the bugs. Bert's task was made a lot easier by Pieter Branderhorst, who separated the MSDOS-specific code from Fractint-for-DOS's fractal generator modules, making a Windows port of the package possible. Mark Peterson helped track down and eliminate a lot of the bugs that were in the original program, and has since added many of the new options which take advantage of the capabilities of the Windows environment.

Fractint for Windows is based heavily on (and uses the fractal generator engines straight out of) Fractint-for-DOS. A partial list of the authors of Fractint-for-DOS includes:

----- Primary Authors (this changes over time) -----

Bert Tyler - CompuServe (CIS) ID: [73477,433]  
Timothy Wegner - CIS ID: [71320,675] Internet: twegner@mwunix.mitre.org  
Mark Peterson - CIS ID: [70441,3353]  
Pieter Branderhorst - CIS ID: [72611,2257]

----- Contributing Authors -----

Michael Abrash	360x480x256, 320x400x256 VGA video modes
Joseph Albrecht	Tandy video, CGA video speedup
Kevin Allen	Finite attractor and bifurcation engine
Steve Bennett	restore-from-disk logic
Rob Beyer	[71021,2074] Barnsley IFS, Lorenz fractals
Mike Burkey	376x564x256, 400x564x256, and 832x612x256 VGA video modes
John Bridges	[75300,2137] superVGA support, 360x480x256 mode
Brian Corbino	[71611,702] Tandy 1000 640x200x16 video mode
Lee Crocker	[73407,2030] Fast Newton, Inversion, Decomposition..
Monte Davis	[71450,3542] Documentation
Chuck Ebbert	[76306,1226] cmprsd & sqrt logmap, fpu speedups
Richard Finegold	[76701,153] 8/16/..256-Way Decomposition option
Frank Fussenegger	Mandelbrot speedups
Mike Gelvin	[73337,520] Mandelbrot speedups
Lawrence Gozum	[73437,2372] Tseng 640x400x256 Video Mode
David Guenther	[70531,3525] Boundary Tracing algorithm
Norman Hills	[71621,1352] Ranges option
Richard Hughes	[70461,3272] "inside=", "outside=" coloring options
Mike Kaufman	[kaufman@eecs.nwu.edu] mouse support, other features
Wesley Loewer	fast floating-point Mandelbrot/Julia logic
Adrian Mariano	[adrian@u.washington.edu] Diffusion & L-Systems
Charles Marslett	[75300,1636] VESA video and IIT math chip support
Joe McLain	[75066,1257] TARGA Support, color-map files
Bob Montgomery	[73357,3140] (Author of VPIC) Fast text I/O routines
Bret Mulvey	plasma clouds
Roy Murphy	[76376,721] Lyapunov Fractals
Ethan Nagel	[70022,2552] Palette editor, integrated help/doc system
Jonathan Osuch	[73277,1432] IIT detect
Marc Reinig	[72410,77] Lots of 3D options
Kyle Powell	[76704,12] 8514/A Support
Matt Saucier	[72371,3101] Printer Support
Herb Savage	[71640,455] 'inside=bof60', 'inside=bof61' options
Lee Skinner	Tetrade, Spider, Mandelglass fractal types and more
Dean Souleles	[75115,1671] Hercules Support
Kurt Sowa	[73467,2013] Color Printer Support

Hugh Steele	cyclorange feature
Chris Taylor	Floating&Fixed-point algorithm speedups, Tesseral Option
Scott Taylor	[72401,410] (DGWM18A) PostScript, Kam Torus, many fn types.
Bill Townsend	Mandelbrot Speedups
Paul Varner	[73237,441] Extended Memory support for Disk Video
Dave Warker	Integer Mandelbrot Fractals concept
Phil Wilson	[76247,3145] Distance Estimator, Bifurcation fractals
Nicholas Wilt	Lsystem speedups
Richard Wilton	Tweaked VGA Video modes

Byte Magazine	...	Tweaked VGA Modes
MS-Kermit		Keyboard Routines
PC Magazine		Sound Routines
PC Tech Journal		CPU, FPU Detectors



## Coordinate Box options

The **Coordinate Box** is courtesy of Mark Peterson. When you have the Coordinate Box checked, a small "coordinate window" constantly displays the current position of your mouse pointer.

From the **Options** menu you can select the coordinates to display in rectangular (default), polar, or pixel coordinates.

**Rectangular** coordinates correspond to Cartesian plane. The coordinates displayed are in absolute units relative to the origin. Fractint uses these coordinates to form a complex number. This complex number initializes one or more variables in the iterative calculation. The  $x$  coordinate is used as the *real* portion of the complex number and the  $y$  coordinate as the *imaginary* portion.

**Pixel** coordinates display the position of a point in terms of the number of pixels, or color dots, relative to the pixel in the upper left-hand corner of the image. For example, if the image size is 200 x 150, then the pixel in the lower right-hand corner of the image is coordinate (199, 149).

**Polar** coordinates display the position of a point in terms of its distance and angle relative to the origin. The angle can be in units of **degrees** (default), **radians**, or **grads**. Most people are familiar with degrees which divide the circle into 360 *degrees*. Grads divide the circle into 400 *grads*. Radians divide the circle into units of  $2\pi$  *radians*.