Fractint for Windows

Help Topics

New Features in this Version

Version 3.1 bug-fixes for version 3.0 (oops):

L-Systems fractals are working again.

Fractal types 'ifs', 'ifs3d', 'formula', and 'julibrot' are activated again.

"printer=ps" in your SSTOOLS.INI file won't crater WINFRACT anymore.

Version 3.1 enhancements:

New Coordinate Box option from Mark Peterson

Version 3.0 enhancements:

"L-systems" fractal types (Docs for which, as usual, are in Fractint-for-DOS's FRACTINT.DOC).

"Copy to Clipboard" has been added to the "Files" menu.

Bug-fixes:

Low-memory conditions are checked for when the program first fires up, and the consequent error messages are a *tad* more informative.

Program crashes caused when Windows moves data segments on Winfract have (we think) been fixed (thanks to James Seidman, who tracked this one down).

Winfract should fire up in 256-color mode when it is run under 16-bit or 24-bit video drivers now (*without* color-cycling, unfortunatley).

It's not really a "bug fix", but we found out that the ATI (and other) 1024x768x16 video problems (Winfract was generating its images several inched *below* its window) were video driver problems - and the latest drivers from ATI fixed the problem.

Bugs and Limitations in this Version

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 is writing this, and he often has no idea what he's 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 (even with a 256-color driver twenty of the colors are "simulated" by Windows, although it's hard to tell that from the image). 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

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 as a GIF (version 89a) file, compatible with Fractint-for-DOS, your favorite DOS GIF viewers, and (of course) the "File Open" option. 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.

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.

Fractal Formula Selection

Using the "Options Formula" option, you can select any of over 60 fractal types (every fractal type that is available in version 14.0r 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:

mandel = 'Classic' Mandelbrot fractals using 32-bit integer math for speed.
z(0) = 0; z(n+1) = z(n)^2 + C, where C = Xcoord + i * Ycoord.
Two optional params: real and imaginary pertibations of z(0).

julia = 'Classic' Julia set fractals using 32-bit integer math for speed.
z(0) = Xcoord + i * Ycoord; z(n+1) = z(n)^2 + C.
Two params required: real and imaginary parts of C.

newton, newtbasin = Newton Domains-of-attraction (only the coloring schemes are different). First param: the power (from 3 to 10) of the eqn.
If param=4, the eqn is z(n+1) = (3*z(n)^4+1)/(4*z(n)^3).
Second param is flag to turn on alternating stripes in basin mode.

complexnewton, complexbasin = Newton's fractal type extended to complex numbers.
Calculates roots of z^a + b, where both 'a' and 'b' complex.

lambda = Classic Lambda fractal. z(0) = Xcoord + i * Ycoord;
z(n+1) = Lambda*z(n)*(1 - z(n)^2). Two params: real, imaginary parts of Lambda. 'Julia' variant of Mandellambda

mandellambda= 'Mandelbrot-Equivalent' for the lambda fractal. z(0) = .5;
lambda = Xcoord + i * Ycoord; z(n+1) = lambda*z(n)*(1 - z(n)^2).
Parameters are pertubations of z(0).

barnsleym1 = Alternative 'Mandelbrot'. z(0) = Xcoord + i * Ycoord;
z(n+1) = (z-1)*C if Real(z) >= 0, else = (z+1)*modulus(C)/C,
where C = Xcoord + i * Ycoord. Params are pertubations of z(0).

barnsleyj1 = 'Julia' corresponding to barnsleym1. z(0) = Xcoord + i * Ycoord;
z(n+1) = (z-1)*C if Real(z) >= 0, else = (z+1)*modulus(C)/C.
Two params required: real and imaginary parts of C.

barnsleym2 = Another alternative 'Mandelbrot'. z(0) = 0; z(n+1) = (z-1)*C
if Real(z)*Imag(C) + real(C)*imag(z) >= 0, else z(n+1) =
(z+1)*C, where C = Xcoord + i * Ycoord. Parameters are
pertubations of z(0).

barnsleyj2 = 'Julia' corresponding to barnsleym2. z(0) = Xcoord + i * Ycoord;
z(n+1) = (z-1)*C if Real(z)*Imag(C) + real(C)*imag(z) >= 0,
else = z(n+1) = (z+1)*C. Two params: real and imaginary parts of C.

barnsleym3, barnsleyj3 = Another Barnsley 'Mandelbrot', 'Julia' pair.
See Fractint-for-DOS's FRACTINT.DOC for details.

sierpinski = Sierpinski gasket - a Julia set that produces a 'Swiss cheese
triangle'. z(n+1) = (2*x,2*y -1) if y > .5; else (2*x-1,2*y)
if (x > .5); else (2*x,2*y). No parameters.

ifs = Barnsley IFS Fractal (a fern unless an alternate IFS map has
been defined using the 'ifs=' command-line option).

ifs3d = Barnsley 3D IFS Fractal (a fern unless an alternate IFS map has
been defined using the 'ifs3d=' command-line option).

marksmandel= A variant of the mandel-lambda fractal. z(0) = 0;
z(n+1) = ((Xcoord+i*Ycoord)^exp)*z(n) + (Xcoord+i*Ycoord).
Parameters are pertubations of z(0).

marksjulia = A variant of the julia-lambda fractal.
z(0) = Xcoord + i * Ycoord; z(n+1) = (z(0)^exp)*z(n) + z(0).

cmplxmarksjul, = The above two types generalized with 'exp' a complex rather
cmplxmarksmand than a real number
julibrot = 'Julibrot' 4-dimensional fractals. See Fractint-for-DOS's
  FRACTINT.DOC for an description of these fractals (and a
  description of the prompts involved in invoking them).
unity = Mark Peterson's 'Unity' fractal type. Truly Wierd -
  See Fractint-for-DOS's FRACTINT.DOC for a full description.
formula = Formula interpreter - write your own formulas as text files!
  See Fractint-for-DOS's FRACTINT.DOC for instructions on this one.
lambdafn = lambda-fn fractal. z(0) = Xcoord + i * Ycoord; z(n+1) =
  lambda * fn(z(n)). Two params: real, imag portions of lambda.
  'Julia' variant of Mandelfn. 'fn' is a variable function,
  and may be one of sin, cos, sinh, cosh, exp, log, or sqr.
mandelfn = 'Mandelbrot-Equivalent' for the lambda-fn fractal.
  z(0) = Xcoord + i * Ycoord; z(n+1) = z(0)*fn(z(n)). Parameters
  are pertubations of z(0). 'fn' is a variable function.
mandel4 = Fourth-power 'Mandelbrot' fractals using 32-bit integer math.
  z(0) = 0; z(n+1) = z(n)^4 + C, where C = Xcoord + i * Ycoord.
  Two optional params: real and imaginary parts of z(0) (if not 0).
julia4 = Fourth-power Julia set fractals using 32-bit integer math.
  z(0) = Xcoord + i * Ycoord; z(n+1) = z(n)^4 + C.
  Two params required: real and imaginary parts of C.
test = 'test' point letting us (and you!) easily add fractal types.
  Currently, the 'Distance Estimator' M'brot/Julia Set algorithm.
  two optional parameters - if none given, uses the M'brot Set"
plasma = plasma clouds - random, cloud-like formations. Requires four or
  more colors. One param: 'graininess' (.5 to 50, default = 2)
julfn+zsqrd= Julia Biomorph fractal. z(0) = Xcoord + i * Ycoord; z(n+1) =
  fn(z(n)) + z(n)^2 + C. Two params: real, imag portions of C.
manfn+zsqrd= 'Mandelbrot-Equivalent' for the Julfn+zsqrd fractal.
  z(0) = Xcoord + i * Ycoord0; z(n+1) = fn(z(n)) + z(n)^2 +
  (Xcoord + i * Ycoord). Parameters are pertubations of z(0).
manzpower = 'Mandelbrot-Equivalent' for julzpower. z(n+1) = z(n)^m + C.
  Parameters are real and imaginary pertubations, exponent m.
julzpower = Juliazpower fractal. z(0) = Xcoord + i * Ycoord; z(n+1) =
  z(n)^m + C. Three params: real, imag portions of C, exponent m.
manzzpwr = 'Mandelbrot-Equivalent' for the julzzpwr fractal. Use the
  Space-bar to select julzzpwr fractals a/la Mandel/Julia.
  z(n+1) = z(n)^z(n) + z(n)^m + C. Parameters are real pertubation,
  imaginary pertubation, and exponent m.
julzzpwr = z(0) = Xcoord + i * Ycoord; z(n+1) = z(n)^z(n) + z(n)^m + C.
  Three params: real, imag portions C, and the exponent m.
manfn+exp = 'Mandelbrot-Equivalent' for the julfn+exp fractal. Use the
  Space-bar to select julfn+exp fractals a/la Mandel/Julia.
  z(n+1) = fn(z(n)) + e^z(n) + C. Parameters are real pertubation,
  and imaginary pertubation of z(0).
julfn+exp = julia fn+exp fractal. z(0) = Xcoord + i * Ycoord; z(n+1) =
  fn(z(n)) + e^z(n) + C. Two params: real, imag portions C.
popcorn = orbits of x(n+1) = x(n) - h*sin(y(n) + tan(3*y(n)) and
  y(n+1) = y(n) - h*sin(x(n) + tan(3*x(n)) plotted for EACH
  screen pixel and superimposed.
popcornjul= Julia set from popcorn formula. z(0) = Xcoord + i * Ycoord.
  Orbit calculated as in popcorn.
bifurcation = 'Bifurcation' fractal. Pictoral representation of a population
  growth model. Let P = new population, p = oldpopulation,

r = growth rate. The model is: P = p + r*p*(1-p).
biflambda = Bifurcation variation: model is: P = r*p*(1-p)P.
bif+sinpi = Bifurcation variation: model is: P = p + r*sin(PI*p).
bif=sinpi = Vifurcation variation: model is: P = r*sin(PI*p).
lorenz = Lorenz attractor - orbit in three dimensions defined by:
xnew = x + (-a*x*dt) + (a*y*dt)
ynew = y + (b*x*dt) - (y*dt) - (z*x*dt)
znew = z + (-c*z*dt) + (x*y*dt)
Parameters are dt, a, b, and c.
lorenz3d = 3D Lorenz attractor with 3D perspective.
rossler3D = Orbit in three dimensions defined by:
xnew = x - y*dt - z*dt
ynew = y + x*dt + a*y*dt
znew = z + b*dt + x*z*dt - c*z*dt
Parameters are dt, a, b, and c.
henon = Orbit in two dimensions defined by:
xnew = 1 + y - a*x*x
ynew = b*x
Parameters are a and b
pickover = Orbit in three dimensions defined by:
xnew = sin(a*y) - z*cos(b*x)
ynew = z*sin(c*x) - cos(d*y)
znew = sin(x)
Parameters are a, b, c, and d.
gingerbread = Orbit in two dimensions defined by:
x <- 1 - y + |x|
y <- x
diffusion = Diffusion Limited Aggregation. Randomly moving points
accumulate. One param: border width (default 10)
fn(z*z) = z(0) = Xcoord + i * Ycoord; z(n+1) = fn(z(n)*z(n))
fn*fn = z(0) = Xcoord + i * Ycoord; z(n+1) = fn(n)*fn(n)
fn*z+z = z(0) = Xcoord + i * Ycoord; z(n+1) = p1*fn(z(n))+p2*z(n)
Parameters are real and imaginary components of p1 and p2.
fn+fn = z(0) = Xcoord + i * Ycoord; z(n+1) = p1*fn1(z(n))+p2*fn2(z(n))
Parameters are real and imaginary components of p1 and p2.
sqr(1/fn) = z(0) = Xcoord + i * Ycoord; z(n+1) = (1/fn(z(n))*(1/fn(z(n))
sqr(fn) = z(0) = Xcoord + i * Ycoord; z(n+1) = fn(z(n))*fn(z(n))
spider = c(0) = z(0) = Xcoord + i * Ycoord; z(n+1) = z(n)*z(n) + c(n);
c(n+1) = c(n)/2 + z(n+1)
manowar = c = z1(0) = z(0) = Xcoord + i*Ycoord; z(n+1) = z(n)^2 + z1(n) + c;
z1(n+1) = z(n);
tetrate = z(0) = c = Xcoord + i * Ycoord; z(n+1) = c^z(n)
kamtorus = series of orbits superimposed. x(0) = y(0) = 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. Parmeters
are a, step size, stop value for 'orbit', and points per orbit.
kamtorus3d= Same as kamtorus but in 3D with 'orbit' the z dimension.
magnetj1 = z(0) = Xcoord + i * Ycoord;

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

Parameters are real and imaginary parts of c.
magnetm1 = 'Mandelbrot' variant with c = Xcoord + i * Ycoord and z(0) = 0.
Orbit formula same as magnetj1

Parameters are pertubations of z(0).

magnetj2 = z(0) = Xcoord + i * Ycoord;

$$z(n+1) = \left| \frac{z(n)^3 + 3*(C-1)*z(n) + (C-1)*(C-2)}{3*(z(n)^2) + 3*(C-2)*z(n) + (C-1)*(C-2) - 1} \right|^2$$

Parameters are real and imaginary parts of c.

magnetm2 = 'Mandelbrot' variant with c = Xcoord + i * Ycoord and z(0) = 0.

Orbit formula same as magnetj2

Parameters are pertubations of z(0).

Image Sizing and Color selection

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

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 in either polar, rectangular, or pixel coordinates. If you are using polar coordinates, you have furthur options of selecting degrees or radians.

Zooming in on an Image, Mandelbrot/Julia Toggling

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.

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

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.

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.

Fractint can now also use "Hot-keys" for color-cycling
spacebar - toggles color-cycling on and off
left, right arrows - turn on color-cycling and set the "direction"
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, ...)

Getting a Status Report

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 menu bar. A "pop-up" message box will tell you about 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. It may be used personally or in a business - if you can do your job better by using Fractint for Windows, or use images from it, that's great! It may be given away with commercial products under the following conditions: It must be clearly stated that Fractint for Windows does not belong to the vendor and is included as a free give-away. It must be a complete unmodified release of Fractint, with documentation. There is no warranty of Fractint's suitability for any purpose, nor any acceptance of liability, express or implied.

Virtually all of the Fractint 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! Several of us can also be found on BIX in the GRAPHIC.DISP/FRACTALS area.

New versions of Fractint-for-DOS and Fractint-for-Windows are uploaded to the Compuserve and BIX networks, and make their way to other systems from those points. The latest version of Fractint-for-Windows can usually be found in the following locations:

WINFRA.ZIP - (Executable/Docs) Compuserve: COMART Lib 15 ("Fractals")
BIX: GRAPHIC.DISP/LISTINGS
WINSRC.ZIP - (Complete Source) Compuserve: COMART Lib 15 ("Fractals")
BIX: GRAPHIC.DISP/LISTINGS

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

A list of Fractint Authors

Fractint for Windows was 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 tracked down some of the bugs that, among other things, have been preventing Winfract from using the incremental linker and added the Coordinate Box options.

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

---------- Primary Authors (this changes over time) ----------
Bert Tyler - Compuserve (CIS) ID: [73477,433] BIX ID: btyler
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 - [73307,606] 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] compressed,sqrt log palette
Richard Finegold- [76701,153] 8/16/../256-Way Decomposition option
Mike Gelvin - [73337,520] Mandelbrot speedups
Lawrence Gozum - [73437,2372] Tseng 640x400x256 Video Mode
David Guenther - [70531,3525] Boundary Tracing algorithm
Mike Kaufman - [kaufman@eecs.nwu.edu] mouse support, other features
Adrian Mariano - [theorem@blake.acs.washington.edu] Diffusion fractal type
Chris Martin - Paintjet printer 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
Ethan Nagel - [70022,2552] Palette editor
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 - Tetrate, Spider, Mandelglass fractal types and more
Dean Souleles - [75115,1671] Hercules Support
Kurt Sowa - [73467,2013] Color Printer Support
Scott Taylor - [72401,410] KAM Torus, many trig function types
Paul Varner - [73237,411] Floating-point fractal algorithms
Dave Warker - Integer Mandelbrot Fractals concept
Phil Wilson - [76247,3145] Distance Estimator, Bifurcation fractals
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