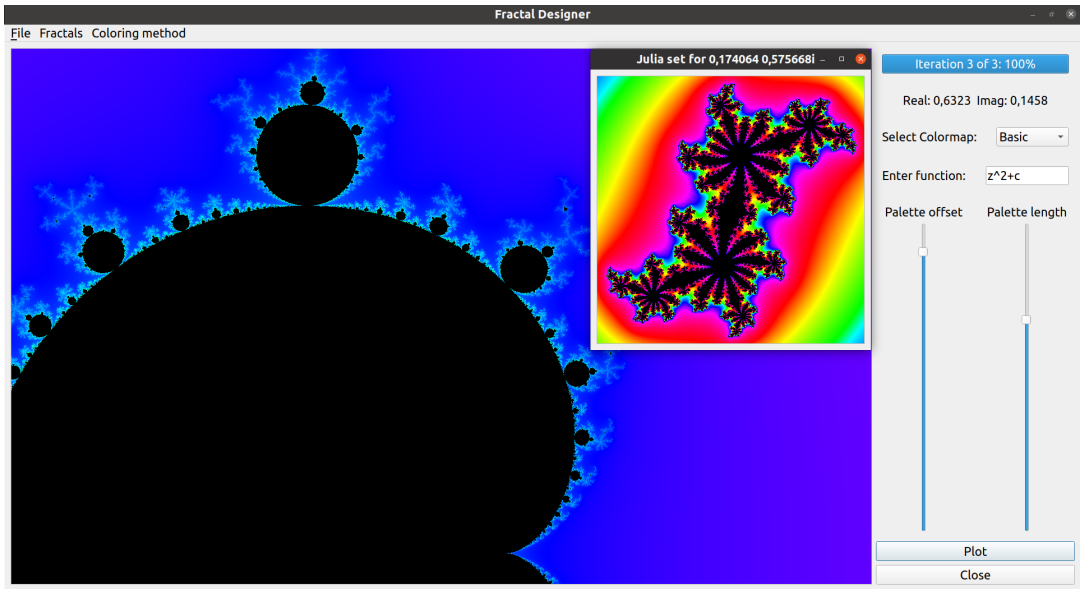
Fractal Designer Documentation

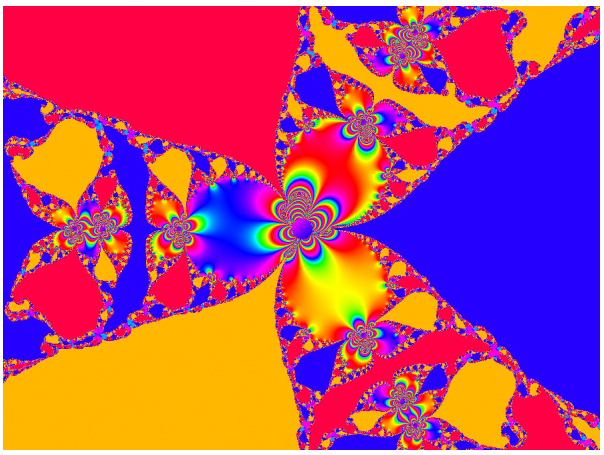
Spirin Ivan, Vinogradov Vasiliy

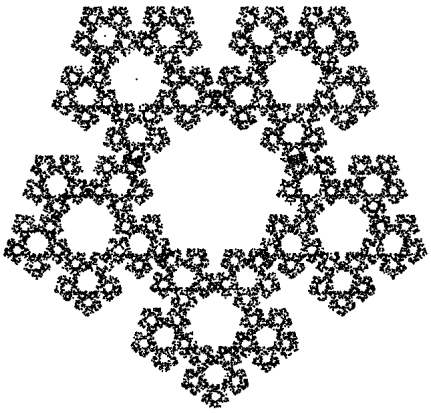
22th May 2021

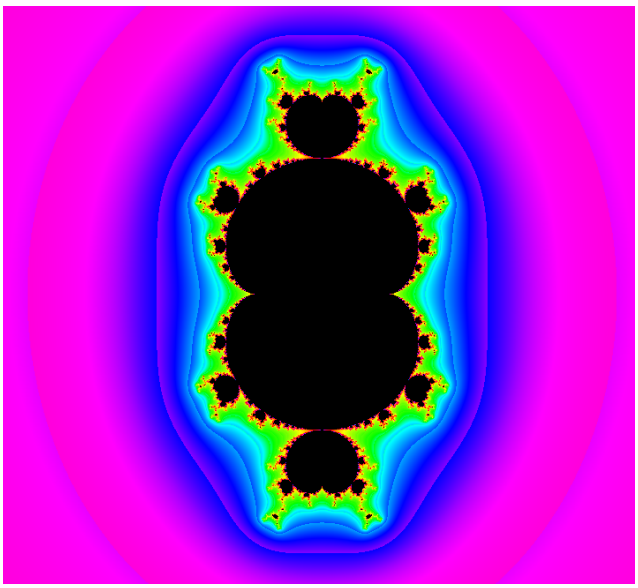
Product Overview

Fractal Designer is a program wich allows you to interact with a complex plane to build, modify and explore fractals. You can zoom in and out, chage construction formulas, set beautiful colormaps e.t.c. Here are some screenshots and fractals:









Controls and features

Basic controls

In Fractal Designer you can use your mouse and keyboard to interact with fractal. Here are some basic controls:

Action	Buttons
Position on plane	Mouse drag while left mouse button pressed
Zoom in	Mouse wheel / E
Zoom out	Mouse wheel / Q
Save image	Ctrl + S
Fast Julia mode	Right mouse button (in Mandelbrot only)
Plot image	Enter
Open documentation	Ctrl+D

Choosing fractal

To switch between fractals go to *Menu→Fractals* and choose the one you want.
The list of fractals available:

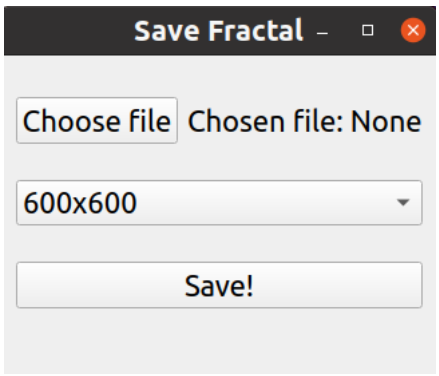
1. Mandelbrot Set
2. Julia Set
3. Newton Fractal (+ its generalization and Nova fractal)
4. Fractals related to secant method
5. IFS fractals
6. L–system fractals

Fast Julia mode

When in Mandelbrot Set mode you are able to click on the image with the right mouse button and a new window will pop up. It will contain a Julia Set connected to the point on the complex plane you have clicked.

Saving an image

Fractal designer allows you to save generated image. In most modes you can also choose the resolution of the image. To save rendered fractal go to *File→Save* or just press Ctrl+S on the keyboard. A window like this



will appear. Click *Choose file* to select where to save the image. Then pick a resolution from the drop–down list. Finally, click *save* and you're done!

Changing formulas

Here is an example of a user defined formula for Mandelbrot Set:

$$f(z, c) = z\text{^}2 + (2 + 5i) * c \ / \ 2 - pi * log(e)$$

User defined formulas support following features:

1. Math constants: pi, e.
2. Elementary functions (to check the full list visit this page):

Function	Description
$abs(z)$	Absolute value for real numbers, module for complex numbers.
$acos(z)/asin(z)$ $atan(z)$	Arc-cosine/arc-sine/arc-tangent of z in radians.
$acosh(z)/asinh$ $atanh(z)$	Same as $acos/asin/atan$ but for hyperbolic functions.
$cos(z)/sin(z)$ $tan(z)/cot(z)$	Cosine/sine/tangent/cotangent of z.
$conj(z)$	Complex conjugate of z.
$exp(z)$	Base 2 exponential of z.
$ceil(z)/int(z)/floor(z)$	Rounds z up/to closest integer/down. Note: $int(-2,5) = -3,\ int(2.5) = 3$
$log(z)/log2(z)/log10(z)$	Base e/base 2/base 10 logarithm of z.
$pow(z_1, z_2)$	Computes $z_1^{z_2}$.
$real(z)/imag(z)$	Returns real/imaginary part of z.
$sqrt(z)/cbrt(z)$	Square/cube root of z.

3. Math operators: $+$, $-$(unary and binary), $*$, $/$, ^ (power).

There are some **rules** for correct work with user defined formulas:

1. The main variable in formulas should be named z. The second variable in Mandelbrot Set formula should be named c.

2. Floating point numbers should have a comma as the separator between integer and fractional parts.

3. Complex numbers should be written as $a \pm bi$ where a and b are real numbers. Note that parentheses can be used to avoid precedence problems: $(1 + 1i) * z$ is not the same as $1 + 1i * z$ (the second expression is equivalent to $1 + (1i * z)$).

4. To avoid problems one may not use whitespaces between expressions and symbols in formula.

To parse formulas we used FunctionParser and to see full syntax constraints you can visit this page.

Palette and coloring

Palette is a function which associates a certain color to a real number. This number could be, in case of Mandelbrot Set, an iteration count required to exceed a mark of module 2 for a complex point in iterative sequence. In most modes you are able to move two sliders: *palette length* slider and *palette offset* slider. Palette is a periodic function, so the first slider controls this period and the second slider changes an offset within one-period section. Pre-defined palettes can be chosen in a corresponding drop-down list.

While zooming in a Mandelbrot set, iteration count used for coloring grows exponentially. This is why sometimes it can be useful to take the logarithm beforehand, so that colors are evenly distributed on a rendered picture. Such option can be enabled in a conforming menu.

Newton fractal

The standard formula for Newton fractals is:

$$z_{n+1} = z_n - \frac{p(z_n)}{p'(z_n)}$$

But Fractal Designer uses generalized formula with Nova fractal available. Therefore you can set different values for constants a and c in the following formula:

$$z_{n+1} = z_n - a\frac{p(z_n)}{p'(z_n)} + c$$

The default values are: $a = 1$, $c = 0$ – to match standard Newton Fractal. If you want to work with Mandelbrot version of Nova fractal just type z in the field related to the real part of the constant c and leave the imaginary part equal to zero.

In this mode you have to enter the derivative of the main formula. Moreover, it is available to select:

1. Type of coloring: by root or by converging speed.

2. Tolerance which is used to check whether the sequence converges.

3. Maximum number of iterations used to check whether sequence converges.

IFS fractals

Iterated function systems are used to plotting fractals. A text field in the right menu is designed for assigning points and variables, which may be used for function implementation. Use the following format: {point_name}={(x,y)}, or {variable_name}={z}, where x,y and z represent floating point numbers.

You can also add/remove/change functions in the list of functions used to create fractal. Linear version of function has the following pattern:

$$f(X) = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot X + A = \begin{pmatrix} ax_1 + bx_2 + a_1 \\ cx_1 + dx_2 + a_2 \end{pmatrix}$$

where X and A are points which are interpreted as two dimentional vectors to calculate the result. The default values are: $a = d = r,\ b = c = 0$, where r is specified by user.

L–system fractals

L–system fractals are constructed iteratively. The rules should be typed in a text field with a format {variable name}:{sequence} with no whitespaces in between. Rules are separated by any kind of whitespace characters. In a final sequence each variable represents an action:

Abbreviation	**Action**
F	Move straight and draw a line
f	Move straight without drawing a line
+	Turn right on an angle
-	Turn left on an angle
Empty	Do nothing

Angle is specified within text line as well as with slider.