fraktaler-3-3.1

Claude Heiland-Allen

2025-12-15

Contents

Fraktaler 3

Fast deep hybrid escape time fractals.

https://fraktaler.mathr.co.uk

Fraktaler 3 is a cross-platform program (Linux, Windows, Android, Web) for fast deep zooming of hybrid escape-time 2D fractals. It has a graphical explorer using SDL2, OpenGLES and Dear ImGUI, and a batch mode for high resolution images and zoom sequences, with optional export of raw data in EXR format compatible with Kalles Fraktaler 2 + and zoomasm.

See images and videos made with Fraktaler 3.

Live

Try Fraktaler 3 live online in your web browser.

https://fraktaler.mathr.co.uk/live/latest

Requires support for `SharedArrayBuffer`, among other web APIs.

Performance is significantly slower than native versions, which are available for download below.

Download

https://fraktaler.mathr.co.uk/download/latest

Source code `fraktaler-3-VERSION.7z`

Documentation `fraktaler-3-VERSION.pdf`

Windows (EXE) `fraktaler-3-VERSION-windows.7z`

Android (APK), requires Android 5 (Lollipop, API 21, 2014) or above `uk.co.mathr.fraktaler.v3-VERSION.ap`

Run

Output of `fraktaler-3 --help`:

```
usage:
  fraktaler-3 [mode] [flags ...] [inputfile [inputfile ...]]
modes of operation:
  -h, --help              print this message and exit
  -V, --version           print version information and exit
  -i, --interactive       interactive graphical user interface
  -b, --batch             command line batch processing
  -W, --generate-wisdom   generate initial hardware configuration
  -B, --benchmark-wisdom  benchmark hardware for optimal efficiency
  -S, --export-source     export this program's source code
flags:
  -v, --verbose           increase verbosity
  -q, --quiet             decrease verbosity
  -p, --persistence file  path to persist state
  -P, --no-persistence    don't persist state
  -w, --wisdom file       path to wisdom
input files are merged in command line order
```

The help text will list the default locations for persistence and wisdom files on your system, as well as the file name for the `--export-source` option.

Multiple parameter files may be specified on the command line. After persistence is loaded, they are merged in order (later files override earlier files). This allows you to keep different aspects of parameters in different files.

Wisdom

Fraktaler 3 can use regular CPU-based code, and OpenCL-based code for CPU and GPU devices. All of these can be used simultaneously. The speed of each device for each number type are stored in a "wisdom" file, along with some other metadata like precision and range of each type, and hardware groupings.

OpenCL may be faster depending on hardware. OpenCL with CPU is typically faster than the regular CPU code, possibly apart from zoom depths between 1e300 and 1e4920 or so where the regular CPU code can use the `long double` number type (on x86/x86_64 hardware).

Fraktaler 3 uses wisdom to automatically choose the best number type and devices to use for each location. If wisdom is not enumerated and benchmarked for your hardware, placeholder defaults are used, which may be suboptimal (for example, OpenCL will not be used).

Wisdom GUI The wisdom dialog in the user interface allows enumeration of hardware. Check the unlock box to the left of the enumerate button, then click the enumerate button.

To benchmark the hardware, check the unlock box to the left of the benchmark button, then click the benchmark button. It takes a couple of minutes per device. Speeds are reported logarithmically: an increase of 1 unit corresponds to a factor 2 increase in speed.

Multiple backends may use the same underlying hardware (for example PoCL and the builtin CPU implementation both use the same processor). To prevent contention, and use only the fastest, hardware can be grouped. For example, enumeration may give cpu0 and cpu1, if these use the same hardware (as is likely) then rename them all to cpu0.

The text fields below the hardware groups are labels for convenience, and do not affect how wisdom is used.

Each backend can be individually enabled or disabled using the check boxes. The currently used backends are highlighted.

Wisdom can be saved to disk. If the file is saved in the default location with the name `wisdom.toml` it will be loaded automatically.

Wisdom CLI To enumerate and benchmark wisdom without the graphical user interface you can run these two steps manually:

```
./fraktaler-3 --generate-wisdom
./fraktaler-3 --benchmark-wisdom
```

The wisdom has two main parts, the `type` map, and the `hardware` map. If a particular (platform, device, numbertype) causes problems when benchmarking wisdom, disable them in the `type` map before benchmarking. If a particular (platform, device) causes problems when benchmarking wisdom, disable them in the `hardware` map before benchmarking.

After benchmarking wisdom, edit the `hardware` map to ensure each (platform, device) is in a group corresponding to the real hardware device: typically this would involve putting the regular CPU code without OpenCL (platform -1) into the same group as CPU with OpenCL (for example the PoCL implementation). You can also rename the devices if desired.

You can specify an alternative wisdom file with the `--wisdom` flag.

Run GUI

See above for details on wisdom for optimal operation.

```
./fraktaler-3 --interactive
```

You need support for recent OpenGLES. If you don't have it, the program window may appear briefly before closing without any error messages visible, or a dialog may appear with an error message.

On Microsoft Windows, if your GPU drivers do not support it you can install Mesa 3D and the Vulkan Runtime from:

- https://github.com/pal1000/mesa-dist-win/releases
- https://vulkan.lunarg.com/sdk/home#windows

Use the `mesa-dist-win` per-app deployment script.

On Linux, with recent Mesa on old hardware, you can try setting the environment variable `LIBGL_ALWAYS_SOFTWARE=1`.

State is remembered between runs, which causes problems with multiple concurrent sessions. To use a different store for this state, you can specify an alternative file with the `--persistence` flag, or disable persistence completely with the `--no-persistence` flag.

Run CLI

See above for details on wisdom for optimal operation.

```
./fraktaler-3 --batch
```

Run Web

Configure web server with headers:

```
Cross-Origin-Embedder-Policy: require-corp
Cross-Origin-Resource-Policy: same-origin
Cross-Origin-Opener-Policy: same-origin
```

Make sure `*.wasm` is served with MIME type `application/wasm`

Serve the `live/` sub-folder. Needs httpS for non-localhost domains.

You must serve the corresponding source code to comply with the license.

Run Android

Install the APK, then click the icon on your app menu.

See above for details on wisdom for optimal operation.

Parameters, colouring shaders, colouring parameters, and images can be saved and loaded, but only in a location private to the app.

Parameters can be exported and imported via the system clipboard, see the Input/Output Window for copy and paste buttons.

GUI

Launch with the `-i` flag (`--interactive`).

Navigation

The F10 key toggles the graphical user interface windows, so you can focus on exploring the fractal. If you do not have a keyboard, you can manually close all the windows by deselecting their checkboxes in the main Fraktaler 3 window, which can also be collapsed by clicking on the small triangle in the top left.

The fractal can be explored with a mouse. Left mouse down on the desired new image center and drag to set the new image size; a rectangle is highlighted during the gesture. Release the left mouse button to confirm the new view, or press the right mouse button (while the left button is still held) to cancel the action. Alternatively use the scroll wheel to zoom in and out around the mouse cursor position. The middle mouse button centers the view on the click location.

The fractal can be explored with a keyboard. Numeric keypad keys 1-9 zoom to different quadrants of the view (1 is bottom left, 9 is top right, 4 is middle left, and so on, as per usual layout). The 0 key zooms out. Page Up and Page Down also zoom keeping the center fixed. Numeric keypad keys + and - adjust the maximum iteration count (doubling and halving respectively), which can also be set in the Bailout window.

The fractal can be explored with multi-touch. One touch translates the view. Two touches zoom and rotate. Three touches enables stretching or skewing the image. If you have no multi-touch device, but do have a mouse and keyboard, you can use multi-touch emulation. Hold Ctrl+Shift and the left mouse button to add or move a touch point. Hold Ctrl+Shift and press the right mouse button to delete a touch point. Delete all touch points to finish the gesture and confirm the action.

Fraktaler 3 Window

This window has toggles to open/close all the other subwindows.

Input/Output Window

This has a Home button to zoom out to the original view. You must set the checkbox to the left to enable this to avoid accidents. There are also buttons to Load and Save, which can be as parameter file text (suggested extension `.f3.toml`) or images with embedded metadata (EXR, PNG, JPEG). Clicking the Load or Save buttons opens a file browser dialog.

There is limited partial supported for reading Kalle's Fraktaler 2 parameters from `.kfr` files amd images wtih embedded KFR metadata. So far only some basic formulas (no hybrids) are recognized, and no palette information is loaded. Location and transformation are supported, so the shapes should look the same even if colours differ. KFR parameters (copied from Kalle's Fraktaler 2 with the Ctrl-X keyboard shortcut) can also be pasted into Fraktaler 3.

Formula Window

The default formula is the Mandelbrot set, with one line with |X|, |Y|, -X, -Y all unchecked and P=2. This corresponds to the familiar formula (X+iY)^2 + C. If you check both |X| and |Y| then you get the Burning Ship (|X|+i|Y|)^2 + C, if instead you check -Y you get the Mandelbar (aka Tricorn) (X-iY)^2 + C. The + button on the right lets you add more than one formula, which can be edited independently. These are iterated in an interleaved fashion, one line after the other in a loop, creating hybrid escape time fractals. Note: reference orbit processing and memory requirements increase with each line (N lines need N times the amounts total as 1 line);.

Clicking the advanced button converts the formula to a list of opcodes:

store store the current Z value for later use by mul.

sqr square the current Z value.

mul multiply the current Z value by the stored value.

absx make the real part of the current Z value positive.

absy make the imaginary part of the current Z value positive.

negx negate the real part of the current Z value.

negy negate the imaginary part of the current Z value.

rot{degrees} rotate the Z value anti-clockwise by the given angle in degrees.

add add C to the current Z value (implicit, always last).

Clicking the button between opcodes inserts a new one. Dropdowns allow changing opcode or deleting.

Switching back to simple mode will generally give a different formula.

Status Window

Shows various progress bars to show how rendering is proceeding. There is also a timer.

Location Window

Shows the coordinates and magnification of the view.

Reference Window

Shows the coordinates and period (if any) of the reference (which is usually the image center).

Bailout Window

Adjust maximum iteration count if there are solid regions that look out of place. Some images may need the iteration limit to be millions.

For complex images increase the perturbation limit (Max Ptb Iters) if increasing the first limit does not fix the issue. A perturbation limit of a few thousand is usually sufficient.

Similarly, the BLA step limit can be increased if necessary. A BLA step limit of a few thousand is usually sufficient. A too-low BLA step limit can lead to glitchy images.

Note: excessively increasing the perturbation and BLA step limits can lead to GPU timeouts, or at least increased calculation time.

The escape radius is adjusted at the bottom, it is automatically clamped to a safe range so that no unsightly rings appear around the fractal (due to numeric overflow) for high power formulas. The radius can be less than 2 for formulas with all powers greater than 2 (“low bailout”).

Transform Window

Adjust image transformation, including reflection (useful if your Burning Ship is upside down), rotation, and stretch. The Auto Stretch (DE) button uses histograms of iteration data to unstretch features.

The exponential map feature is not so useful in the graphical program, but can be used for rendering a zoom out sequence for later assembly into a video using zoomasm (https://mathr.co.uk/zoomasm) or other tools.

The vertical flip setting is useful if your video assembler expects exponential map images the other way up (only affects saved images, not visible appearance in the user interface).

Algorithm Window

Contains advanced algorithm tuning options. Be careful if you adjust these as sometimes bad images can result. See algorithm parameters section below.

Information Window

Displays histograms of iteration data. At least one sample must have been completed for this to be updated. If the checkboxes are activated, the histograms are scaled logarithmically.

The top three histograms can be used to check bailout limits: if there is a peak at the right hand edge then the limit is too low.

The distance histogram can be used to diagnose distorted or stretched images, which can be compensated by Auto Stretch (DE) button in the Transform window.

Quality Window

The base image size can be set in pixels and physical dimensions. Lock the print resolution DPI to make changing physical dimensions change the pixel size instead of the DPI. Click the Apply button to confirm image size changes, or Cancel to discard changes.

The Super control multiplies the base image size, useful for quickly changing size for high resolution prints (for example).

The Sub control divides the pixel size while keeping the displayed size the same, useful for reducing quality to speed up interactive use.

The Samples control allows increasing quality by computing many slightly different versions of the image and averaging them. Setting this to 0 will compute indefinitely, allowing you to stop when the quality gets high enough for you, but can’t be used when rendering. Raw iteration EXR export requires 1 sample.

Render Window

The render window can be used to render zoom out sequences.

The filename stem will be appended with a frame number and file type extension; you can use an absolute path if you want to save outside the current working directory.

When saving a zoom out sequence for assembly into a video with zoomasm, use zoom out factor 2, and remember to enable Exponential Map in the Transform window with a very wide image aspect ratio in the Quality window.

The Start Frame and Frame Count options can be used to render part of a zoom out sequence. The PRNG Seed sets the pseudo-random number generator seed, which might be useful in rare circumstances.

Multiple file formats can be saved for each frame. EXR output can be either RGB (any number of samples) or raw (only 1 sample supported, set in Quality window).

To render, unlock the Render button at the top (only displayed if it is possible to render) before clicking it. You can tell Fraktaler 3 to exit when finished by activating the Quit When Done checkbox before clicking the Render button.

To cancel rendering, unlock the Stop button below the progress bar and click it.

To resume a zoom sequence:

1. load the last successfully saved key frame (either as `.f3.toml` or as metadata embedded in an image);
2. note the sequence number in the name of this file (e.g. 00001234);
3. in the Render window, set Numbering Offset to this number (e.g. 1234);
4. in the Render window, set Start Frame to 1 to resume rendering with the next key frame after the loaded parameter;
5. resume rendering with the Render button.

Colours Window

The top widget sets the background colour surrounding the fractal.

If the colouring uses `getHistogram()`, you need to enable the Uses Histogram checkbox (enabled by default), otherwise colouring will be incorrect. Leaving it checked is always safe but might be slightly slower. Technically, when the checkbox is enabled, the first sample's tiles will be recoloured once the histogram is available (after the first sample has finished calculating).

Unlock the Reset button and click it to reset colouring shader to the default, which is a monochrome distance estimator iwth brightness, contrast, exposure and gamma controls (gamma 0 looks bad, set it to 1 instead).

Below the settable variables is a compilation control, Auto Compile makes it update on every key press which can get laggy, if disabled you can Compile by hand.

The colouring is done by an OpenGL Shading Language (GLSL) source code snippet. Any active uniforms can be set as variables above. Sometimes unused uniforms are eliminated by the shader compiler so might not be shown.

You need to mplement this, which should return linear RGB (*not* sRGB):

```
vec3 colour(void);
```

Inside your `colour()` function you can get current pixel raw data:

```
uint  getN0 (void);
uint  getN1 (void);
float getNF (void);
float getT  (void);
float getDEX(void);
float getDEY(void);
vec2  getDE (void);
flaot getHistogram(void);
vec2  getCoord(void);     // in range (0,0) to getImageSize()
```

Histogram is a quantile (between 0 and 1) of `N` over the image.

You can also get image parameters:

```
ivec2 getImageSize(void); // size of whole image in pixels
float getTime(void);      // seconds since program launch
float getZoomLog2(void);  // log2 of zoom depth
```

There are also some colour space conversion utility functions:

```
float linear2sRGB(float c);
vec3  linear2sRGB(vec3 c);
float sRGB2linear(float c);
vec3  sRGB2linear(vec3 c);
vec3  hsv2sRGB(vec3 c);
vec3  hsv2rgb(vec3 c);
```

Shader compilation errors are shown at the bottom of the window.

You can export colouring shaders by choosing `.glsl` extension when saving in the Input/Output window. You can import colouring shaders by loading in the Input/Output window with Load Colouring Only enabled.

Postprocessing Window

The Postprocessing window allows quick basic colour manipulations: Brightness, Contrast, Gamma and Exposure. For higher quality results these should be applied in the Colours shader code.

RGB histograms are displayed both before and after the postprocessing colour adjustments.

Newton Zooming Window

Zooms automatically to mini-sets or embedded Julia sets deep in the fractal. Set the options (each action includes the ones above), then select the activate checkbox and left-click in the image where you want to zoom. Remember to deselect the activate checkbox if you want to use the left mouse zooming feature. This happens automatically unless the Keep Active checkbox is enabled.

The Capture button sets the starting zoom level for relative Newton zoom. Auto Capture activates Capture after each successful Newton Zoom.

The Power value sets how far to zoom from the captured zoom to the mini. The Factor value sets a fine-tune scale factor. The Power presets adapt to the degree of the formula: the first formula line determines doubling (2 4 8… for degree 2) vs tripling (3 9 27… for degree 3) etc behaviour when approaching the mini-set.

Wisdom Window

Configures wisdom, see discussion above.

Preferences Window

UI Scale makes the interface widgets bigger, from 100% (default) up to 1000%.

Maximum image size in Megapixels can be limited to avoid denial of service from extreme parameters.

Maximum iteration limits can be set (or even disabled if you're living on the edge) to avoid denial of service from extreme parameters. This is especially important for some GPU drivers where timeouts due to long-running calculations can exit the desktop session, losing unsaved data in any open applications.

About Window

Displays version information and software licenses.

CLI

Launch with the `-b` flag (`--batch`).

Parameters

Parameters are stored in TOML format (suggested filename extension `.f3.toml`). Parameters that are unchanged from the default values are omitted from files saved by Fraktaler 3. Metadata is also stored in EXR image files (viewable with the `exrheader` program).

Loading parameters saved from old versions into a new version of Fraktaler 3 should always work. Loading parameters saved from new versions of Fraktaler 3 into an old version may behave unexpectedly.

Location Parameters

Location parameters are strings to store more range and/or precision than TOML double precision floating point.

```
location.real = "0"
location.imag = "0"
location.zoom = "1"
```

Zoom 1 (without Transform) corresponds to vertical axis from -2 to $+2$, zoom 2 to -1 to $+1$, and so on.

When saving from the GUI, long strings are broken across multiple lines.

Reference Parameters

The reference defaults to the location (i.e. center of image).

```
reference.real = location.real
reference.imag = location.imag
reference.period = 0
```

Setting an inaccurate period is a good way to get corrupt images; use Newton zooming dialog to find correct period. 0 means unknown, don't use period for anything.

When saving from the GUI, long strings are broken across multiple lines.

Bailout Parameters

Iterations can be set arbitrarily high without too much slowdown. Maximum reference iterations should normally be set to the same as the iterations setting, setting it too low can lead to corrupt images. Maximum perturb iterations can be left at a few 1000 usually, increase it if you get blobby spiral centers or if mini-sets are not sharp enough for your taste. Maximum BLA steps can be a few 1000 too, increase it if you get glitchy areas.

Escape radius and inscape radius do not usually need to be changed, if you get strange iteration bands with high powers then reduce the escape radius (this is due to overflow of floating point range).

```
bailout.iterations = 1024
bailout.maximum_reference_iterations = 1024
bailout.maximum_perturb_iterations = 1024
bailout.maximum_bla_steps = 1024
bailout.escape_radius = 625.0
bailout.inscape_radius = 9.765625e-4
```

Transform Parameters

Angles are in degrees, stretch amount is in cents. Usually you would adjust these interactively in the GUI using multitouch (or multitouch emulation), or via Newton zooming dialog, or the Autostretch DE button.

Reflect flips the imaginary axis direction.

Exponential map is useful for zoom out sequences, not currently very usable in the user interface.

Vertical flip is sometimes useful for output (does not affect visible image in the GUI, only saved files).

```
transform.reflect = false
transform.rotate = 0.0
transform.stretch_angle = 0.0
transform.stretch_amount = 0.0
transform.exponential_map = false
transform.vertical_flip = false
```

Image Parameters

Sets output image dimensions in pixels. Increasing subsampling reduces image size by that factor. Increasing subframes increases quality (antialiasing samples per pixel). When subframes is more than 1, output EXR files contain only `RGB` data. When subframes is 1, output EXR files in batch mode also contain raw calculation data so will be large.

```
image.width = 1024
image.height = 576
image.subsampling = 1
image.subframes = 1
```

Render Parameters

The filename will have `.exr` appended, or `.########.exr` appended for zoom out sequences (where `########` is the frame number). A frame count of 0 means the zoom out sequence continues until fully zoomed out (zoom $< 1.0/65536.0$).

```
render.filename = "fraktaler-3"
render.zoom_out_sequence = false
render.zoom_out_factor = 2.0
render.numbering_offset = 0
render.start_frame = 0
render.frame_count = 0
```

See the documentation above (Render Window section) for information about resuming zoom sequence rendering.

Newton Parameters

These parameters set the defaults in the GUI.

```
newton.action = 3
newton.domain = false
newton.absolute = false
newton.power = 0.5
newton.factor = 4.0
```

Algorithm Parameters

If the period of the reference is known (for example after Newton zooming), then locking the maximum reference iterations to the period can give a good speedup for high iteration areas. When used with a bad period, bad images can result (for example, moving too far from the valid area can cause pixelation artifacts).

When rendering zoom out sequences, the same reference can be reused instead of being recomputed, saving time. The reference will be recalculated at each number type change. Reusing bilinear approximation is not generally applicable at the present time (it depends on zoom depth, so will be less efficient when zooming in, and cause problems zooming out).

```
algorithm.lock_maximum_reference_iterations_to_period = false
algorithm.reuse_reference = false
algorithm.reuse_bilinear_approximation = false
```

The number types could be restricted in earlier versions, but this is no longer implemented here: use wisdom settings for this instead.

OpenCL Parameters

Increase tile size as much as reasonable without hitting operating system timeouts (bad images will result in that case, or even crashes of your desktop session, potentially losing unsaved data). For example in one test location, the default `128x128` took 3 minutes, while `960x1080` took 1m34s, which was only a fraction slower than `7680x4320` (one tile for the whole image). Making sure there aren't small fragments of tiles at image edges is important too, otherwise effective parallelism is reduced.

```
opencl.tile_width = 128
opencl.tile_height = 128
```

The tile size also affects regular CPU rendering without OpenCL. It's then best to set the tile so that the image size is a common multiple of the tile size and core count. If using multiple GPUs something similar applies: set the tile size so that the image size is common multiple of the tile size and number of GPUs. This is more important when using the graphical user interface as it renders one subframe at a time (and so if the tile size is the image size only one device will be used at a time): batch mode renders subframes all at once and does not have this limitation.

Tile size can be adjusted in the Algorithm dialog.

Formula Parameters

After all the other parameters, multiple formula blocks corresponding to each line in the formula dialog.

```
[[formula]]
abs_x = false
abs_y = false
neg_x = false
neg_y = false
power = 2
```

Recommended Parameters for Zoomasm

Zoomasm https://mathr.co.uk/zoomasm is a tool for assembling zoom out sequences containing raw iteration data in exponential map format, into zoom videos. Fraktaler 3 can save in a compatible format.

You may need to increase the location zoom so that the first frame's bottom edge is completely interior (or exterior), otherwise the end of the zoom may look strange in zoomasm.

Render exponential map zoom out sequence keyframes with raw data included (subframes 1):

```
image.width = 12288
image.height = 1360
image.subframes = 1
transform.exponential_map = true
render.zoom_out_sequence = true
algorithm.reuse_reference = true
opencl.tile_width = 768
opencl.tile_height = 680
```

Make the tile size smaller if problems occur.

If reference period is known:

```
reference.period = ...
algorithm.lock_maximum_reference_iterations_to_period = true
```

Depending on your zoom assembler, you may want to add

```
transform.vertical_flip = true
```

Zoomasm has an option to flip on load, but others might not.

Source

You can clone the source code repository from:

```
git clone https://code.mathr.co.uk/fraktaler-3.git
```

Build

Source Dependencies

```
git clone https://github.com/ocornut/imgui.git
git clone https://github.com/AirGuanZ/imgui-filebrowser.git
git clone https://github.com/epezent/implot.git
git clone https://github.com/martijnberger/clew.git
git clone https://github.com/Armchair-Software/emscripten-browser-clipboard.git
git clone https://github.com/Armchair-Software/emscripten-browser-file.git
git clone https://code.mathr.co.uk/fraktaler-3.git
```

Tested with versions as of 2025-11-16:

- imgui v1.92.4-84-gc254db763 with `fraktaler-3/src/imgui-sdl2-text-input.patch` applied
- imgui-filebrowser 47a1884
- implot v0.16-42-g285df95
- clew 0.10-28-g50751dd
- emscripten-browser-clipboard bcba48f
- emscripten-browser-file ffd21d1 with `fraktaler-3/src/emscripten-browser-file-pthread-fix.patch` applied

clew is only used when cross-compiling for Windows.

emscripten-browser-clipboard and emscripten-browser-file are only used when cross-compiling for Web.

Optional Features

All features are enabled by default. You can disable them by adding variables to the `make` command line, for example:

```
make STDCXX=c++14 CL= TOML11=3 FLOAT128= EXR=0 FS= DEBUG=
```

will use C++14 instead of C++17, without OpenCL acceleration, with toml11 version 3 instead of 4, without _Float128 (quadruple precision) support, without OpenEXR image saving support, without C++ filesystem support, and with debug symbol generation disabled.

Build For Debian

Bookworm or newer is recommended. These instructions are for Trixie, other releases may need adaptations.

```
sudo apt install \
  build-essential \
  cpio \
  git \
  icoutils \
  libglm-dev \
  libmpfr-dev \
  libmpfrc++-dev \
  libopenexr-dev \
  libsdl2-dev \
  libtoml11-dev \
  ocl-icd-opencl-dev \
  opencl-headers \
  p7zip \
  pkg-config \
  pocl-opencl-icd \
  xxd
```

Debian gcc

```
make
```

Debian clang

```
make SYSTEM=native-clang
```

Debian Buster The OpenEXR library in Debian Buster does not support C++17. C++17 is required for its filesystem module. There are three options:

1. Compile without OpenEXR support:

```
make EXR=0
```

This will mean you cannot export EXR image files at all.

2. Compile without filesystem support:

```
make STDCXX=c++14 FS=
```

This will mean no file dialogs in the graphical user interface. You can use the persistence mechanism to extract parameters from the GUI and render them using the command line interface.

3. Backport a newer OpenEXR library version to your system.

This may break other software you have installed.

Build For Windows

For cross-compilation from Debian.

```
sudo dpkg --add-architecture i386
sudo apt update
sudo apt install \
  build-essential \
  cpio \
  git \
  icoutils \
  mingw-w64 \
  p7zip \
  wine32 \
  wine64 \
  wine-binfmt \
  xxd
sudo update-alternatives --set x86_64-w64-mingw32-g++ /usr/bin/x86_64-w64-mingw32-g++-posix
sudo update-alternatives --set x86_64-w64-mingw32-gcc /usr/bin/x86_64-w64-mingw32-gcc-posix
sudo update-alternatives --set x86_64-w64-mingw32-gfortran /usr/bin/x86_64-w64-mingw32-gfortran-posix
sudo update-alternatives --set x86_64-w64-mingw32-gnat /usr/bin/x86_64-w64-mingw32-gnat-posix
sudo update-alternatives --set i686-w64-mingw32-g++ /usr/bin/i686-w64-mingw32-g++-posix
sudo update-alternatives --set i686-w64-mingw32-gcc /usr/bin/i686-w64-mingw32-gcc-posix
sudo update-alternatives --set i686-w64-mingw32-gfortran /usr/bin/i686-w64-mingw32-gfortran-posix
sudo update-alternatives --set i686-w64-mingw32-gnat /usr/bin/i686-w64-mingw32-gnat-posix
```

Use build-scripts to download and build dependencies for your architecture. For example,

```
LIBS="clew deflate gmp imath jpeg mpfr mpreal openexr3 opengl glm sdl2 toml11 zlib png"
./BUILD.sh download "$LIBS"
./BUILD.sh x86_64-w64-mingw32 "$LIBS"
```

Windows i686

```
make SYSTEM=i686-w64-mingw32
```

Batch mode works in Wine on my system. GUI works in Wine on my system. OpenCL did not work in Wine on my system. Microsoft Windows is untested.

Windows x86_64

```
make SYSTEM=x86_64-w64-mingw32
```

Batch mode works in Wine on my system. GUI works in Wine on my system. OpenCL works in Wine on my system. Microsoft Windows is untested.

Windows armv7 You need `llvm-mingw` because `gcc-mingw` does not support Windows on ARM: https://github.com/mstorsjo/llvm-mingw

Note: `-lopengl32` is not supported upstream yet, so the GUI won't be compiled.

Note: Wine is untested. Microsoft Windows is untested.

```
make SYSTEM=armv7-w64-mingw32
```

Windows aarch64 You need `llvm-mingw` because `gcc-mingw` does not support Windows on ARM: https://github.com/mstorsjo/llvm-mingw

Note: `-lopengl32` is not supported upstream yet, so the GUI won't be compiled.

Note: Wine is untested. Microsoft Windows is untested.

```
make SYSTEM=aarch64-w64-mingw32
```

Build For Web

Emscripten Dependencies Use build-scripts to download and build dependencies for Emscripten. For example,

```
LIBS="emsdk deflate gmp imath jpeg mpfr mpreal openexr3 opengl glm sdl2 toml11 zlib png"
./BUILD.sh download "$LIBS"
./BUILD.sh emscripten "$LIBS"
```

Emscripten Build

```
emmake make SYSTEM=emscripten FLOAT128=
```

Build For Android

You need Android command line tools with SDK and NDK. Tested with these versions:

```
$ ${ANDROID_HOME}/cmdline-tools/latest/bin/sdkmanager --list_installed
[=======================================] 100% Fetch remote repository...
Installed packages:
  Path                                     | Version            | Description                        | Loca
  -------                                  | -------            | -------                            | ----
  build-tools;30.0.2                       | 30.0.2             | Android SDK Build-Tools 30.0.2     | buil
  build-tools;33.0.1                       | 33.0.1             | Android SDK Build-Tools 33.0.1     | buil
  cmdline-tools;latest                     | 14.0.0             | PLACEHOLDER                        | cmdl
  emulator                                 | 34.2.14            | Android Emulator                   | emul
  ndk;25.1.8937393                         | 25.1.8937393       | NDK (Side by side) 25.1.8937393    | ndk/
  ndk;25.2.9519653                         | 25.2.9519653       | NDK (Side by side) 25.2.9519653    | ndk/
  ndk;26.3.11579264                        | 26.3.11579264      | NDK (Side by side) 26.3.11579264   | ndk/
  ndk;27.0.11718014                        | 27.0.11718014 rc1  | NDK (Side by side) 27.0.11718014   | ndk/
  platform-tools                           | 35.0.1             | Android SDK Platform-Tools         | plat
  platforms;android-21                     | 2                  | Android SDK Platform 21            | plat
  platforms;android-31                     | 1                  | Android SDK Platform 31            | plat
  system-images;android-21;default;x86_64 | 5                  | Intel x86_64 Atom System Image     | syst
```

Use the `android.sh` script to download and build dependencies for Android. Set environment variables to configure, for example:

```
ANDROID_HOME=/opt/android-sdk
ANDROID_NDK_HOME=${ANDROID_HOME}/ndk/25.2.9519653
PATH="${ANDROID_HOME}/tools:${PATH}"
PATH="${ANDROID_HOME}/platform-tools:${PATH}"
PATH="${ANDROID_NDK_HOME}:${PATH}"
./build/android.sh prepare
./build/android.sh debug
```

Default is a debug build (runs slow). Release build requires signing.

Build Documentation

Needs `pandoc`. Built as part of release.

Build Release

Builds all architectures and documentation ready for release. Does not yet include Android.

```
./build/release.sh clean
./build/release.sh DEBUG= EXR=2
```

Theory

See https://mathr.co.uk/web/deep-zoom.html.

Alternatives

Other fractal deep zoom software that also uses bilinear approximation (BLA) for acceleration includes:

fractalshades Fractalshades is a Python package for creating static and interactive visualisations of 2d fractals. It targets Windows and Unix operating systems and implements efficient algorithms for very-deep exploration of the Mandelbrot and the Burning_Ship sets (1.e-2000 scale and beyond).

FractalShark A Mandelbrot Set renderer optimized for use with NVIDIA GPUs.

Fractal Zoomer An application that lets you render some of the most known fractal functions. It comes with alot of options to further enhance your fractal experience! Its easy to use and does not require installation. A java version higher than 1.8 is required to be installed.

Imagina Imagina is a fast fractal renderer & explorer.

The project is being rewritten. This repository may be renamed and replaced by the rewritten version when it's available.

Mandelbrot Perturbator GTK A Mandelbrot set explorer using GTK, with many features for annotating with (pre)periods, rays, regions, etc.

. . . Get in touch if you know of other software (closed or open source, payware or gratis) that is comparable and I'll add it to the list!

Bugs

For an up-to-date bug list see https://mathr.co.uk/web/fraktaler.html#Bugs-In-Version-3-1.

History

Version 0

2021-12-10 : project started.

Version 1

2023-03-13 : version 1 released. 423 git commits since version 0.

Version 1.1

2023-03-20 : version 1.1 released. 18 git commits since version 1.

- fix smooth iteration calculation for powers other than 2.
- fix perturbation glitches for high powers.
- fix too-bright EXR export from GUI.
- fix vertically flipped EXR export from GUI.
- fix auto stretch with reflection enabled.
- fix Newton transform with reflection enabled.
- fix reflection intuition in transformation GUI.
- fix rotation intuition in transformation GUI.
- fix crash when zooming out too far.
- fix progress reporting (done tiles vs started tiles).
- fix crash when loading bad wisdom.
- fix cancelling tiles takes too long with CPU backend.
- fix for multiple parameters on command line.
- fix non-central references.
- fix image dimensions in GUI (now correctly locked to window size).
- fix wisdom benchmarking (now easier to disable devices).
- fix misleading `number_types` setting (was nonfunctional, now deleted).

- fix iteration band glitches for high powers; a partial fix with workarounds still be required:
  - lower bailout escape radius (changes appearance with some colourings);
  - disable single precision float (low range type) in wisdom (may be slower).

Version 2

2023-03-31 : version 2 released. 59 git commits since version 1.1.

- formula engine rewritten using opcode model:
  - faster computationally,
  - more robust mathematically,
  - more flexible artistically.
- wisdom configuration via the graphical user interface.
  - note: wisdom file format is incompatible with earlier versions, wisdom will need to be regenerated.
- preview transformations for keyboard navigation.
- clipboard copy/paste in Input/Output dialog.
- new BLA steps limit in Bailout dialog reduces computation time.
- lock reference iterations to iterations to prevent bad images.
- tile size can be set in Algorithm dialog.
- fix Newton zoom dialog custom size entry.
- fix wisdom hardware grouping logic.

Version 1.2

2023-07-13 : version 1.2 released. 10 git commits since version 1.1.

- fix bad rendering to the left of the needle.
- fix typo in wisdom stopping.
- documentation improvements.
- use build-scripts for third-party dependencies.
- upgrade third-party dependencies to latest versions.

Version 1.2.1

2023-07-14 : version 1.2.1 released. 2 git commits since version 1.2.

- fix build for web with current emscripten versions.

Version 2.1

2023-07-14 : version 2.1 released. 14 git commits since version 2.

- incorporate changes from versions 1.2 and 1.2.1.
- fix typo preventing setting bla steps by text entry.

Version 3

2025-06-10 : version 3 released. 225 git commits since version 2.

- flexible colouring algorithms using GLSL:
  - editable shader source;
  - parameters (shader uniform variables) adjustable in user interface;
  - tiles cached for recolouring without recomputing fractal.
- image size decoupled from window size with full screen mode.
- image export with metadata, and metadata import from images:
  - PNG (RGB8);
  - JPEG (with RGB32F -> YUV8 conversion for higher quality);
  - EXR (RGB32F).
- improved interactive zooming experience (no longer replaces image with grey when first tile arrives).
- graphical interface for zoom sequence rendering
- information window:
  - iteration count histograms to inspect limits;
  - distance estimate histogram to inspect skew.
- post-processing window:
  - RGB histogram displays;
  - basic colour adjustments.
- new program icon.
- optimisations:
  - BLA skip levels setting reduces memory requirements;
  - reference calculation has periodicity check.
- Android build system overhauled, for dependencies now use build-scripts.
- vertical flip option for image export.
- support low bailout (escape radius less than 2 for higher powers).
- Newton zoom improvements:
  - zoom power presets adapt to degree of first formula line;
  - only auto capture new zoom depth if Newton succeeded.
- fix pasto in negy reference (thanks FractalLion).
- remove low-zoom reference 0 hack (thanks Microfractal).
- requires OpenGLES 3 / WebGL 2 / Android 21.

Version 3.1

2025-12-15 : version 3.1 released. 28 git commits since version 3.

- feature: histogram colouring: shaders can use `getHistogram()` which reports the quantile in 0 to 1 of `N` for the current image; Uses Histogram checkbox can be disabled for a small speed boost if histogram colouring is not in use.
- feature: example histogram colouring with SunsetColors palette.
- feature: partial support for reading KFR parameters exported from Kalle's Fraktaler 2 (most formula locations including transformation, but not hybrid formulas or colouring).
- feature: the web version's canvas uses the full size of the browser window.
- feature: doubleexp number type (double precision floatexp); use when single precision floatexp is not precise enough.
- feature: float128 number type (quadruple precision); disabled automatically on unsupported architectures.
- feature: wisdom reports sizes in bytes; re-enumeration and benchmarking is required.
- fix: correct matching between appearance of CPU and OpenCL backends; thanks to JWM for reporting.
- fix: compatibility with toml11 v4.
- fix: builds and runs in Termux + Termux X11 on Android
- fix: no 'quit when done' option on Web or Android.
- fix: correct status when pressing ESC after rendering is complete.
- internal: build system improvements.

Future

For an up-to-date todo list see https://mathr.co.uk/web/fraktaler.html#Future.

Legal

---

https://mathr.co.uk