fraktaler-3-2.1

Claude Heiland-Allen

2023-07-14

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. On Android, the save and load buttons operate only with this file.

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 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 (EXR format, extension `.exr`). Clicking the Load or Save buttons opens a file browser dialog. Note: parameters saved as metadata in EXR image files cannot yet be reloaded.

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 adbanced 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.

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, decrease it for high power formulas if unsightly rings appear around the fractal.

Transform Window

Adjust image transformation, including reflection (useful if your Burning Ship is upside down), rotation, and stretch. The exponential map feature is not so useful in the graphical program, but can be used in the command line version for rendering a zoom out sequence for later assembly into a video using zoomasm (https://mathr.co.uk/zoomasm).

Algorithm Window

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

Quality Window

Control image quality. Increasing top value decreases quality (but increases speed) by subsampling the image. Increasing the bottom value increases quality by computing many versions of the image and averaging them. Setting the bottom value to 0 will compute more subframes indefinitely, allowing you to stop when the quality gets high enough for you.

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.

Wisdom Window

Configures wisdom, see discussion above.

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.

```
transform.reflect = false
transform.rotate = 0.0
transform.stretch_angle = 0.0
transform.stretch_amount = 0.0
transform.exponential_map = 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.start_frame = 0
render.frame_count = 0
```

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
```

Source

You can browse the source code repository at:

https://code.mathr.co.uk/fraktaler-3

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/ToruNiina/toml11.git
git clone https://github.com/martijnberger/clew.git
git clone https://code.mathr.co.uk/fraktaler-3.git
```

Tested with versions as of 2023-07-13:

- imgui `v1.89.7-18-g77eba4d0d`
- imgui-filebrowser `fbafb08`
- toml11 `v3.7.1-79-g1340692`
- clew `0.10-28-g50751dd`

clew is only used when cross-compiling for Windows.

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= EXR=0 FS= DEBUG=
```

will use C++14 instead of C++17, without OpenCL acceleration, without OpenEXR image saving support, without C++ filesystem support, and with debug symbol generation disabled.

Build For Debian

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

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

Debian gcc

```
make headers
make
```

Debian clang

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

Build For Windows

For cross-compilation from Debian.

```
sudo dpkg --add-architecture i386
sudo apt update
sudo apt install \
  build-essential \
  git \
  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 desired architecture. For example:

```
git clone https://code.mathr.co.uk/build-scripts.git
cd build-scripts
./BUILD.sh download "gmp mpfr mpreal glm sdl2 zlib imath openexr3"
./BUILD.sh x86_64-w64-mingw32 "gmp mpfr mpreal glm sdl2 zlib imath openexr3"
```

Windows i686

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

Emscripten Dependencies

Use build-scripts to download and build dependencies. For example:

```
git clone https://code.mathr.co.uk/build-scripts.git
cd build-scripts
./BUILD.sh download "emdsk gmp mpfr mpreal glm sdl2 zlib imath openexr3"
./BUILD.sh emscripten "emsdk gmp mpfr mpreal glm sdl2 zlib imath openexr3"
```

Emscripten web

```
make headers
make SYSTEM=emscripten
```

Build For Android

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

```
claude@eiskaffee:~/opt/android$ ./cmdline-tools/tools/bin/sdkmanager --list_installed
Installed packages:=====================] 100% Fetch remote repository...
  Path                 | Version      | Description                     | Location
  -------              | -------      | -------                         | -------
  build-tools;21.1.2   | 21.1.2       | Android SDK Build-Tools 21.1.2  | build-tools/21.1.2
  build-tools;30.0.2   | 30.0.2       | Android SDK Build-Tools 30.0.2  | build-tools/30.0.2
  emulator             | 32.1.12      | Android Emulator                | emulator
  ndk;21.4.7075529     | 21.4.7075529 | NDK (Side by side) 21.4.7075529 | ndk/21.4.7075529
  patcher;v4           | 1            | SDK Patch Applier v4            | patcher/v4
  platform-tools       | 34.0.1       | Android SDK Platform-Tools      | platform-tools
  platforms;android-21 | 2            | Android SDK Platform 21         | platforms/android-21
  platforms;android-31 | 1            | Android SDK Platform 31         | platforms/android-31
```

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

```
ANDROID_HOME=${HOME}/opt/android
ANDROID_NDK_HOME=${ANDROID_HOME}/ndk/21.4.7075529
PATH="${ANDROID_HOME}/tools:${PATH}"
PATH="${ANDROID_HOME}/platform-tools:${PATH}"
PATH="${ANDROID_NDK_HOME}:${PATH}"
./build/android.sh prepare
./build/android.sh
```

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=3
```

Theory

References:

perturbation technique http://www.science.eclipse.co.uk/sft_maths.pdf

rebasing and BLA https://fractalforums.org/f/28/t/4360

distance estimates https://mathr.co.uk/helm

interior detection https://fractalforums.org/f/28/t/4802

The Mandelbrot Set

High precision reference orbit:

$$Z_{m+1} = Z_m^2 + C$$

m starts at 0 with $Z_0 = 0$.

Perturbation

Low precision deltas relative to high precision orbit. Pixel orbit $Z_m + z_n$, $C + c$.

$$z_{n+1} = 2Z_m z_n + z_n^2 + c$$

m and n start at 0 with $z_0 = 0$.

Rebasing

Rebasing to avoid glitches: when

$$|Z_m + z_n| < |z_n|$$

replace z_n with $Z_m + z_n$ and reset the reference iteration count m to 0.

Bivariate Linear Approximation

When Z is large and z is small, the iterations can be approximated by bivariate linear function;

$$z_{n+l} = A_{n,l} z_n + B_{n,l} c$$

This is valid when the non-linear part of the full perturbation iterations is so small that omitting it would cause fewer problems than the rounding error of the low precision data type.

Single Step BLA

Approximation of a single step by bilinear form is valid when

$$
\begin{aligned}
|z_n^2| &<< |2Z_n z_n + c| \\
&\Uparrow \quad \text{definition of } A_{n,1}, B_{n,1} \text{ for single step} \\
|z_n^2| &<< |A_{n,1} z_n + B_{n,1} c| \\
&\Uparrow \quad \text{definition of } \epsilon \text{ (for example, } \epsilon = 2^{-24}) \\
|z_n^2| &< \epsilon |A_{n,1} z_n + B_{n,1} c| \\
&\Uparrow \quad \text{triangle inequality} \\
|z_n^2| &< \epsilon |A_{n,1} z_n| - \epsilon |B_{n,1} c| \\
&\Uparrow \quad \text{algebra} \\
|z_n|^2 - \epsilon |A_{n,1}||z_n| + \epsilon |B_{n,1}||c_n| &< 0 \\
&\Uparrow \quad \text{quadratic formula} \\
|z_n| &< \frac{\epsilon |A_{n,1}| + \sqrt{(\epsilon |A_{n,1}|)^2 - 4\epsilon |B_{n,1}||c|}}{2} \\
&\Uparrow \quad \text{linear Taylor polynomial (**approximation**)} \\
|z_n| &< \epsilon |A_{n,1}| - \frac{|B_{n,1}|}{|A_{n,1}|} |c| =: R_{n,1}
\end{aligned}
$$

For single step of Mandelbrot set:

$$
\begin{aligned}
A_{m,1} &= \frac{\partial Z_{m+1}}{\partial Z_m} = 2Z_m \\
B_{m,1} &= \frac{\partial Z_{m+1}}{\partial C} = 1 \\
R_{m,1} &= \max \left\{ 0, \epsilon 2 |Z_m| - \frac{|c|}{2|Z_m|} \right\}
\end{aligned}
$$

Note: this is different to the formulas suggested by Zhuoran on Fractal Forums, but I couldn't get them to work, and this version does seem to work fine.

Merging BLA Steps

If T_x skips l_x iterations from iteration m_x when $|z| < R_x$ and T_y skips l_y iterations from iteration $m_x + l_x$ when $|z| < R_y$ then $T_z = T_y \circ T_x$ skips $l_x + l_y$ iterations from iteration m_x when $|z| < R_z$:

$$
\begin{aligned}
z_{m_x + l_x + l_y} &= A_y (A_x z_{m_x} + B_x c) + B_y c = A_z z_{m_x} + B_z c \\
A_{m_x, l_x + l_y} &= A_z = A_y A_x \\
B_{m_x, l_x + l_y} &= B_z = A_y B_x + B_y \\
R_{m_x, l_x + l_y} &= R_z = \max \left\{ 0, \min \left\{ R_x, \frac{R_y - |B_x||c|}{|A_x|} \right\} \right\}
\end{aligned}
$$

BLA Table Construction

Suppose the reference has M iterations. Create M BLAs each skipping 1 iteration (this can be done in parallel). Then merge neighbours without overlap to create $\lceil \frac{M}{2} \rceil$ each skipping 2 iterations (except for perhaps the last which skips less). Repeat until there is only 1 BLA skipping $M - 1$ iterations: it's best to start the merge from iteration 1 because reference iteration 0 always corresponds to a non-linear perturbation step as $Z = 0$.

The resulting table has $O(M)$ elements.

BLA Table Lookup

Find the BLA starting from iteration m that has the largest skip l satisfying $|z| < R$. If there is none, do a perturbation iteration. Check for rebasing opportunities after each BLA application or perturbation step.

Non-Conformal BLA

The Mandelbrot set is conformal (angles are preserved). This means complex numbers can be used for derivatives. Some other formulas are not conformal, for example the Tricorn aka Mandelbar, defined by:

$$X + iY \to (X - iY)^2 + C$$

For non-conformal formulas, replace complex numbers by 2×2 real matrices for A, B. Dual numbers with two dual parts can be used to calculate the derivatives.

Be careful finding norms. Define $\sup |M|$ and $\inf |M|$ as the largest and smallest singular values of M. Then single step BLA radius becomes

$$R = \epsilon \inf |A| - \frac{\sup |B|}{\inf |A|}|c|$$

and merging BLA steps radius becomes

$$R_z = \max\left\{0, \min\left\{R_x, \frac{R_y - \sup |B_x||c|}{\sup |A_x|}\right\}\right\}$$

ABS Variation BLA

The only problem with the Mandelbrot set is the non-linearity, but some other formulas have other problems, for example the Burning Ship, defined by:

$$X + iY \to (|X| + i|Y|)^2 + C$$

The absolute value folds the plane when X or Y are near 0, so the single step BLA radius becomes the minimum of the non-linearity radius and the folding radii:

$$R = \max\left\{0, \min\left\{\epsilon \inf |A| - \frac{\sup |B|}{\inf |A|}|c|, |X|, |Y|\right\}\right\}$$

Currently Fraktaler 3 uses a fudge factor for paranoia, dividing $|X|$ and $|Y|$ by 2. The merged BLA step radius is unchanged.

Hybrid BLA

For a hybrid loop with multiple phases, you need multiple references, one starting at each phase in the loop. Rebasing switches to the reference for the current phase. You need one BLA table per reference.

Multiple Critical Points

Some formulas (but none among those implemented in Fraktaler 3) have multiple critical points. In this case some modifications need to be made: you need a reference per critical point, and rebasing needs to switch to the nearest orbit among all critical points. There needs to be a separate BLA table for each reference. This also applies to hybrids, you need one reference and BLA table per critical point per phase.

Distance Estimation

Keep track of derivatives of $Z + z$ wrt. pixel coordinates k. As Z is constant for the whole image, you just need $\frac{dz}{dk}$. An easy way to do this is with dual numbers for automatic numeric differentiation. Set up the pixel coordinates as dual numbers with dual part $1 + 0i$, then transform them to the complex C+c plane of the

fractal iterations. At the end you plug the complex derivative into the (directional) distance estimate formula, it is already prescaled by the pixel spacing (this also helps to avoid overflow during iteration).

For non-complex-analytic formulas (like Mandelbar and Burning Ship), you can use dual numbers with two dual parts, for each of the real and imaginary components. At the end they can be combined into a Jacobian matrix and used in the (directional) distance estimate formula for general iterations.

Interior Detection

Keep track of derivatives of $Z + z$ wrt. $Z_1 + z_1$ (where $Z_0 + z_0$ is at the critical point, usually 0). When the absolute value of the derivative drops below a threshold such as 0.001, classify it as interior and stop iterating. For non-complex-analytic formulas, dual numbers with four dual parts can be used (two for distance estimation and two for interior detection), along with matrix operator norm.

Using $\frac{dz}{dz_1}$ works because:

$$\begin{aligned}
&\frac{d(Z+x)}{d(Z_1+z_1)} \\
=&\frac{dZ}{d(Z_1+z_1)} + \frac{dz}{d(Z_1+z_1)} \\
=&\frac{1}{\frac{dZ_1}{dZ} + \frac{dz_1}{dZ}} + \frac{1}{\frac{dZ_1}{dz} + \frac{dz_1}{dz}} \\
=&\frac{1}{\frac{dZ_1}{dZ} + 0} + \frac{1}{0 + \frac{dz_1}{dz}} \\
=&\frac{dZ}{dZ_1} + \frac{dz}{dz_1} \\
=&0 + \frac{dz}{dz_1} \\
=&\frac{dz}{dz_1}
\end{aligned}$$

where the last two lines hold when C is periodic with $Z = 0$ in the orbit which happens precisely when the formula has a critical point at 0 and C is the nucleus of a hyperbolic component.

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).

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.

... 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!

TODO

These missing features could be classified as bugs if you're mean.

- there are no colouring algorithm options
- implement GUI for zoom out sequence rendering
- fix IO
  - should load metadata from images
  - CLI should have an option to save TOML from argument (which could be an image)
  - implement EXR channel output filters (to save disk space and time)
- implement low + high bailout
  - ensure BLA doesn't escape past low bailout
  - don't use BLA between low bailout and high bailout
  - store cooked values at low bailout
  - store cooked values at high bailout
  - option to rename channels to avoid clashes
  - channel filters to save memory and calculation time (no-DE mode?)
- stripe average colouring based on last few iterations
  - checkpoint iterations and roll-back if BLA skipped too far
  - see if low + high bailout is good enough for colouring, hopefully won't need iterations before low bailout?
  - maybe `float` will not have enough range here, switch to `floatexp` for last few iterations (or assume the `+ c` is trivial)
- optimize MPFR memory allocation
  - period detection
  - root finding
  - size calculation
- optimize conformal formulas
  - use complex numbers instead of matrices
  - Mandelbrot set / multibrot only
- number type wisdom
  - detect blank images and omit (platform, device, type) from wisdom
- high resolution rendering dialog
  - dimensions in inches and dots per inch
  - automatically translated to/from pixels
  - exports to toml for command line renderer
  - option to enable reuse reference and zoom out sequence
- extend colouring algorithms
  - port nice algorithm from Rodney, parameterized
  - allow custom OpenCL source for colouring snippet (no parameters)
  - allow custom GLSL source with dynamically generated UI for uniforms
  - use OpenCL/OpenGL (with/without interop) to do colouring with custom GLSL with UI
  - use OSMesa to do colouring without a DISPLAY
- compat with other software
  - KFR location import, including metadata from image files
  - KFR location export, including metadata to image files
  - KFP palette import (with default GLSL implementation copied from KF)
  - KF custom GLSL import mode (see zoomasm)
  - custom GLSL export for zoomasm
- Android
  - log crashes somehow and start with option not to restore persistence
  - support earlier versions
- Web
  - fix copy/paste from host OS into ImGUI dialog boxes
  - export/import parameters to/from host clipboard or via up/download

  - export/import parameters to/from URL hash (base64)
  - export image as download (browser canvas right click is captured)
- Windows
  - GUI on ARM is missing

Bugs

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

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.

Legal

Fraktaler 3 – Fast deep escape time fractals

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License along with this program. If not, see https://www.gnu.org/licenses/.

---

https://mathr.co.uk