Contents

Kalles Fraktaler 2 +

As the orginal upstream author Karl Runmo says:

> Want to create DEEP Mandelbrot fractals 100 times faster than the commercial programs, for FREE? One hour or one minute? Three months or one day? Try Kalles Fraktaler!

It works by using perturbation techniques and series approximation, to allow faster lower precision number types to be used for pixel iterations, based on a high precision reference.

I (Claude Heiland-Allen) forked the code and swapped out the custom arbitrary precision floating point code for highly optimized libraries, making it even faster. Cross-compiled to Windows from Linux MINGW64. Now with many other enhancements.

Original upstream version:

- http://www.chillheimer.de/kallesfraktaler/

This version:

- https://mathr.co.uk/kf/kf.html

Feedback:

- https://fractalforums.org/kalles-fraktaler/15 current forum
- http://www.fractalforums.com/kalles-fraktaler/ legacy forum (read only)
- mailto:claude@mathr.co.uk?subject=Kalles%20Fraktaler%202 personal mail

Quick Start

- Download the latest archive from the website: https://mathr.co.uk/kf/kf.html#download
- Unzip it wherever you want, no installation required. You need `7-zip` to unzip `.7z` archives, available from https://www.7-zip.org/.
- Launch `kf.x86_64.exe` for 64-bit (recommended), `kf.i686.exe` for 32-bit.
- Start exploring!
  - Use the mouse scroll wheel to zoom.
  - Choose different fractal formulas in the formula dialog (Ctrl-F).
  - Choose different bailout conditions in the bailout dialog (Ctrl-B).
  - Choose different colours in the colouring dialog (Ctrl-C).
  - Zoom deeper more quickly with Newton-Raphson zooming (Ctrl-D).
  - Design your own hybrid fractal formula (Ctrl-H).
  - Design your own colouring algorithm using GLSL (Ctrl-G).

Limits

- Windows limits bitmap images to 2 giga-bytes (~715 M-pixel)
- Windows limits longest bitmap dimension to 64 kilo-pixels
- KF limits iteration count to 9 exa-iterations
- KF limits periods to 2 giga-iterations
- MPFR limits precision to 2 giga-bits (~646 million base-10 digits)
- KF needs up to ~50 bytes per pixel (typically half that for common uses)
- KF needs up to ~40 bytes per reference iteration (depending on number type)
- KF cannot currently make use of more than 64 CPU threads

Known Bugs

- Windows Defender sometimes reports that KF contains malware (false positive, I hope - I check with virustotal.com and all seems fine) (possibly due to screenshots for crosshair window, some code also moves the mouse cursor)
- out of memory conditions cause crashes (for example, if bitmap creation fails - also need to check huge sizes) (reported by gerrit)
- setting window size too big (eg width 12800) makes it disappear while KF is still running (in Wine on Linux/XFCE)
- increasing window size when window is near the top can make the title bar go off screen. workaround for Windows 10: Alt+Space+M, or Shift+RightMouse on program icon in the task bar and select Move; then use cursor keys to move the window; press RightMouse to finish. workaround for Linux/XFCE: hold Alt and drag the window with LeftMouse button (from any point inside it). (reported by saka)
- resizing window during examine zoom sequence auto solve glitches leads to corruption of the zoom sequence data
- "stop autosolve" during examine zoom sequence fails and corrupts zoom sequence
- minimizing window during zoom sequence rendering corrupts image (saves blank image or repeated frame) (reported by gerrit and CFJH)
- with "reuse reference", corrupt image at transition between number types (eg e600) (reported by CFJH) - workaround is to render in segments or force the number type higher ("use long double always", "use floatexp always")
- on special locations kf renders endless references and comes to no end (reported by CFJH) (happens also sometimes in exponential map rendering) (untested workaround may be to adjust glitch center method to random)
- with glitch center found by argmin|z|, endless references with little progress (reported by gerrit, only some locations)
- analytic DE broken with some power 3 Mandelbrot locations (reported by gerrit) (workaround is to disable series approximation)
- fractal type out of range (e.g. parameter from newer KF) is silently reset to Mandelbrot
- "resume zoom sequence" re-uses last set zoom count limit
- "resume zoom sequence" sometimes uses wrong image size (depending on settings and whether there are `last.kfb`, `recovery.kfb`, `*_*.kfb`) (workaround: set image size before resuming sequence, for example by loading the last saved image file with both Open Settings and Open)
- "examine zoom sequence" doesn't save corrected PNG images during glitch solve
- black regions when rendering zoom out sequence (maximum iterations are reduced too much before spirals appear in next frame) (reported by gerrit) workaround is to disable auto-iterations
- there is still a race conditions in guessing (doesn't wait for previous progressive interlacing pass to be 100% done before the next one starts)
- newton-raphson zoom preset depths are bad for formulas with power other than 2

- burning ship series approximation stops at first fold (typically 1 period of a central miniship), could potentially subdivide the region (and shift the series, for the folded part) and carry on?
- burning ship series approximation probe points might miss some folds near the edges of the image (need to increase probe point density?)
- bad combinations of skew, distance estimation, and series approximation
- scaled (long) double yr,yi can underflow to 0, eventually causing derivatives to be too small and de overflows to infinity -> blank screen: workaround is to force long double or floatexp as appropriate
- auto skew (escape) button doesn't work well with some formulas (eg SimonBrot)
- navigation with scroll wheel and -/+ keys is hardcoded to factor of 2 instead of using the zoom size set in the View menu
- NR zoom doesn't work well in skewed locations
- nanomb1/2 OrderM, OrderN can only be changed by hand-editing .kfs Settings files
- nanomb2 RadiusScale can only be changed by hand-editing .kfs Settings files
- nanomb1/2 number type fixed to floatexp (long double or double would be faster)
- nanomb1/2 number type is not rescaled (only matters for long double / double)
- nanomb2 currently disables glitch detection and correction
- nanomb1/2 reference calculations are not multithreaded (single core only)
- nanomb1/2 reference calculations are using slow Boost C++ wrapper for MPFR
- crosshair window contents is misaligned if OS window scaling is changed
  - workaround for Windows 10: "You can quickly open the Magnifier by using the keyboard shortcut Windows key with the plus sign (+) to zoom in and Windows key with minus sign (-) to zoom out. Use Windows key and Esc to exit the magnifier."
  - workaround for Linux/Wine: third-party magnifiers include `kmag`, `magnus`, `vmg`, `xmag`, `xzoom`; I had best results with `kmag`.
- help button in file browser does nothing
- may be difficult to build the source at the moment (dependency on 'et')
- seams/bands with mixed power hybrids and numerical DE/slope (workaround: use analytic DE/slope with derivatives)
- formula `z^2 exp(2 a / z) + c` does not work with OpenCL (causes error dump re missing complex exp and sinh)
- guessing with OpenCL on GPU does not give full speedup (e.g. 1.5x faster instead of 4x faster with lots of very high iteration interior)
- guessing with OpenCL requires both image dimensions to be a multiple of 2
- if hybrid formula is invalid, Ok button does the same as Cancel
- OpenCL runs on the main thread making the GUI less responsive
- sometimes files are overwritten without confirmation, when the filename is entered without extension (eg "test" vs "test.jpg") (reported by CFJH).
- derivative-based glitch test for power 2 Mandelbrot is buggy at some locations (reported by Microfractal); workaround is to disable derivatives if possible, or use OpenCL and/or Hybrid fractal designer (but that may be much slower).
- horizontal seams with OpenGL GLSL entropy colouring when image is larger than the internal tile size.

Differences From Upstream 2.11.1

Incompatible Changes

- **In versions `2.15.2` and above**, OpenGL-based colouring is new.
- **In versions `2.15.1` and above**, slope colouring is different; output is different with smooth method log (iteration count is offset by `-log(log(BailoutRadius))/log(Power)`); Buffalo power 2 is different; rotation and skew are different; hybrid formulas are new; exponential map is new.
- **In versions `2.14.10` and above**, new controls for custom bailout escape radius; bailout norm power can be customized; bailout real/imag factors can be fractional and/or negative;

the smoothing method is decoupled from escape radius; there is a new phase channel (`T` in EXR), with "phase strength" colour control; the texture information is saved and loaded from parameters; new fourth root transfer function (color method 11)

- **In versions `2.14.9` and above**, the new formula `z^2 exp(2 a / z) + c` is not available in earlier versions.
- **In versions `2.14.8` and above**, the iteration count limit is raised above 2 giga-iterations. Parameters with such high iterations counts will load incorrectly in earlier versions.
- **In versions `2.14.8` and above**, directional DE is saved in EXR files using `DEX` and `DEY` channels, instead of distance using `DE` channel. Old EXR files' `DE` can not be re-loaded in newer versions, and new EXR files' `DEX` and `DEY` can not be re-loaded in older versions.
- **In versions `2.14.7` and above**, the new "LogLog" and "ATan" iteration count transfer functions (ColorMethod 9 and 10) are not available in earlier versions.
- **In versions `2.14.5` and above**, the new "Low bailout" (SmoothMethod: 2) is not available in earlier versions.
- **In versions `2.14.5` and above**, the two new General Abs Quadratic formulas are not available in earlier versions.
- **In versions `2.14.4` and above**, the new "Mothbrot" formulas are not available in earlier versions.
- **In versions `2.14.3` and above**, colouring of zoomed out views is fixed, and may be different from earlier versions.
- **In versions `2.14.1` and above**, there are two new formulas which are not available in earlier versions.
- **In versions `2.13.1` through `2.13.9`**, derivatives are calculated by default. In versions `2.14.1` and above, you need to enable the derivatives calculations (in the Iterations dialog, or via a settings file) if you want to use analytic distance estimation colouring. Derivatives are not available in earlier versions.
- **In versions `2.13.1` through `2.13.4` inclusive** the interior is white. In `2.13.5` and above it is user-settable, defaulting to black, Parameter files made with earlier `2.13` versions should be modified to explicitly set a white interior colour. In earlier versions (including upstream `2.11.1`) the interior is black with no way of changing it.
- **In version `kf-2.12.10` only** the jitter is "Gaussian" with no way of changing the shape. In `2.12.11` and above, the shape can be changed, and the default is now "uniform". Earlier versions (including upstream `2.11.1`) do not support jitter at all. To get the same results as `2.12.10` you must enable the Gaussian jitter checkbox and set the jitter scale box to 1.
- **In version `kf-2.12.1` and above**, DE colouring method #5 is once again backwards compatible with upstream `2.11.1`. Parameter files made with `2.11.1+gmp.DATE` versions should be modified to use Distance (Square Root) colouring method #8.
- **In version `kf-2.11.1+gmp.20170822` only**, DE colouring method #5 used log instead of sqrt for a more perceptually linear effect. In later versions, this log scaling is achieved with a new colouring method #7, while the DE colouring method #5 reverts to sqrt as before. The new colouring method ID allows old `2.11.1+gmp.DATE` parameter files to be loaded into current versions and display as intended. Any parameter files saved with the new Distance (Logarithm) colouring method will not display as intended in older versions. Parameter files using Distance colouring method saved with this particular version should be modified to use Distance (Logarithm) in the latest version.
- multiple finite difference methods for distance colouring (only the default `Traditional` is available in `2.11.1`).
- analytic DE colouring with derivatives (not available in `2.12.x` or earlier).

Other Changes

- Makefile build system using MINGW to cross-compile to Windows from Linux
- uses GMP+MPFR for arbitrary precision floating point instead of custom code
- uses Boost wrapper around MPFR floats for higher-level coding
- use installed JPEG library, instead of bundled sources
- long double support built into EXE (no separate DLL needed)
- virtually unlimited precision (memory needed for precise numbers is an issue)
- threaded calculations reimplemented with barriers to avoid WINE slowdown
- workaround for WINE issue artificially limiting image size (up to 2GiB now)
- bugfix: inflection performance issue (was converting number types needlessly)
- bugfix: cross-hair resource issue (reported and fixed by Kalles Fraktaler)
- miscellaneous code cleanups (-fpermissive fixes, const fixes, delete[] fixes, 64bit compatibility paranoia)
- formula inner loops generated at compile time from high level specification XML using XSLT and a preprocessor implemented in Haskell
- optimized some reference calculations by floating temporaries out of loops
- XML preprocessor optimizes more reference calculations in the same way
- optimized Newton-Raphson zooming by using lower-level GMP calls
- save images to PNG, TIFF and EXR format as well as JPEG
- colouring uses floating point internally (fewer quantisation steps)
- dithering at end of colouring to improve perceptual quality (reduced banding)
- "glitch low tolerance" checkbox that can be enabled to detect glitches more aggressively (disabled by default, enable it if you get undetected glitches)
- updated program icon with transparent background and large version
- parameter data is saved as comment in image files (all formats)
- preferences (rendering settings not related to location) save and load (.kfs files and image comments too)
- command line arguments to load settings and/or location
- command line arguments to render and save images / maps before quiting
- see change log below for more...

Change Log

- **kf-2.15.2.4** (2021-05-23)
  - regression fix: restore built-in formula Newton-Raphson zooming
  - fix: use bitmap stride calculations from Microsoft documentation (4 byte row alignment)
- **kf-2.15.2.3** (2021-05-22)
  - fix SIMD implementation of sgn() (test location by FractalAlex)
  - fix opengl pack alignment for non-multiple-of-4 width bitmaps
  - fix palette From Image not updating number of colours (reported by Microfractal)
  - fix palette From Image to use maximum of width and height
  - add PNG to supported images for palette From Image (reported by Microfractal)
  - fix argmin|z| glitch correction for OpenCL
  - report detailed timing information in command line mode
  - update dependencies
    - boost to 1.76.0
    - tiff to 4.3.0
    - openexr to 2.5.6
    - glfw to 3.3.4
- **kf-2.15.2.2** (2021-04-07)
  - new: entropy colouring example palette in OpenGL GLSL
  - fix: tiling seams with certain colouring algorithms in OpenGL GLSL

  - fix: checkbox to disable power 2 Mandelbrot derivative based glitch test (failing location reported by Microfractal)

- **kf-2.15.2.1** (2021-03-31)
  - fix: OpenGL shader errors on NVIDIA drivers

- **kf-2.15.2** (2021-03-31)
  - new: custom OpenGL colouring algorithms (compatible with *zoomasm* 3.0)
    - button bottom left of the colors dialog or Ctrl+G in main window
    - text area for GLSL editing, or import and export to `*.glsl` files
    - GLSL is stored in KFP palettes, KFR files and image metadata
    - user API allows access of KFP parameters within GLSL
  - new: OpenGL implementation of colouring algorithm
    - default shader gives similar results as the regular implementation
    - there are tiny differences (1 or 2 ULP) due to different rounding
    - uses portions of libqd ported to GLSL for higher precision (49 bits, vs 24 bits for float and 53 bits for double on CPU)
    - some features remain unimplemented in this version:
      - least squares 2x2 numerical differencing algorithm
      - least squares 3x3 numerical differencing algorithm
  - new: experimental ARM builds are possible using llvm-mingw
    - 64bit aarch64 can be built but crashes on start
    - 32bit armv7 is blocked on a GMP bug(?) (missing symbols in library)
    - still needs gcc windres because llvm windres is incomplete
  - new: option to save EXR files without their 8bit RGBA preview images (makes files smaller and saves faster)
  - new: sRGB gamma-correct downscaling (using patched pixman)
  - fix: removed special case for zooms less than 1e3, fixes rendering of Redshifter 1 cubic a=2 z0=-4/3 (reported by gerrit)
  - fix: make undo and redo refresh the colors dialog
  - fix: remove warning about slow unused derivatives, because it is hard to tell if GLSL colouring might need them
  - fix: dependency preparation script adapted to new architectures
  - fix: dependency preparation script can build subsets of libraries
  - fix: correctly initialize "no glitch detection" vector in SIMD
  - fix: use `delete[]` instead of `delete` in some places
  - fix: use `static inline` to prevent redefinition errors in llvm
  - upgrade to pixman 0.40+git (claudeha/kf branch)
  - upgrade to tiff 4.2.0
  - upgrade to openexr 2.5.5

- **kf-2.15.1.6** (2020-12-08)
  - merge changes from kf-2.14.10.7

- **kf-2.14.10.7** (2020-12-08)
  - revert 2.14.10.4's threading model change: back to win32 from posix (fixes weirdness like zooming out resetting zoom to infinity or 0) (reported by Dinkydau)

- **kf-2.15.1.5** (2020-11-24)
  - fix long double rendering for hybrid formulas with abs
  - fix typos breaking OpenCL floatexp bailout checks
  - fix guessing for OpenCL floatexp
  - fix endless reference loop in OpenCL
  - merge changes from kf-2.14.10.6

- **kf-2.14.10.6** (2020-11-24)

  - fix long double rendering for formulas with abs (reported by Mr Rebooted)
  - fix floatexp rendering for fractional bailout norm power
  - fix some float conversion warnings and potential problems
  - make prepare script modular (second argument determines libraries to build, default is all libraries)
  - patch OpenEXR to try to fix issues with non-ASCII paths on Windows (reported by Ookamitada Shirohi)

- **kf-2.15.1.4** (2020-11-18)
  - merge changes from kf-2.14.10.5

- **kf-2.14.10.5** (2020-11-18)
  - fix scaled long double rendering (broken since 2.14.8) (reported by CFJH)
  - fix SIMD assertion failure in 32bit build (reported by gerson)
  - upgrade to gmp 6.2.1

- **kf-2.15.1.3** (2020-11-12)
  - fix hybrid perturbation logic (perturbing a constant (power 0) should give 0 not the constant again) (thanks Foxxie)
  - fix more typo breakage in OpenCL hybrids
  - fix OpenCL enablement when loading settings from command line
  - fix OpenCL exponential map DE scaling (fixes visible rings in zoomasm)
  - merge changes from kf-2.14.10.4

- **kf-2.14.10.4** (2020-11-10)
  - fix "solve glitches by random choice" not having enough entropy in command line rendering (picking the same point over and over with no progress)
  - switch MinGW compiler threading model from win32 to posix: no longer need to patch `OpenEXR` to use `mingw-std-threads` (the latter is no longer needed at all)
  - rebuild all dependencies for the win32 to posix switch
  - patch OpenEXR to avoid calling `_wsopen_s` which does not exist in ReactOS' `msvcrt.dll`
  - adjust build system to ensure `stdc++` and `pthread` libraries are statically linked
  - explicitly target Win32 API version 0x501 (Windows XP)
  - return of 32bit builds to the distribution
  - updated Linux build documentation and scripts (native Windows build instructions and scripts are out of date)
  - documentation improvements (thanks to FractalAlex)

- **kf-2.15.1.2** (2020-11-08)
  - refactor OpenCL error handling to display errors in the GUI without exiting
  - OpenCL hybrids: fix breaking typo in neg x (reported by Microfractal)

- **kf-2.15.1.1** (2020-10-28)
  - fix OpenCL support for NVIDIA GPUs (reported by bezo97)
  - fix crash in aligned memory (de)allocation (reported by gerrit)
  - documentation improvements (thanks to FractalAlex)

- **kf-2.15.1** (2020-10-28)
  - OpenCL support for perturbation iterations (requires double precision support on device: CPUs should work, some GPUs might not)
  - hybrid formula editor (design your own fractal formula)
  - exponential map coordinate transformation (useful for export to *zoomasm* for efficient video assembly)
  - rotation and skew transformations are rewritten to be more flexible (but old skewed/rotated KFR locations will not load correctly)

  - `kf-tile.exe` tool supports the new rotation and skew transformations
  - the bitrotten skew animation feature is removed
  - a few speed changes in built in formulas (one example, RedShiftRider 4 with derivatives is almost 2x faster due to using complex analytic derivatives instead of 2x2 Jacobian matrix derivatives)
  - flip imaginary part of Buffalo power 2 (to match other powers; derivative was flipped already)
  - slope implementation rewritten (appearance is different but it is now independent of zoom level and iteration count)
  - smooth (log) iteration count is offset so dwell bands match up with the phase channel

- **kf-2.14.10.3** (2020-10-08)
  - fixes
    * assertion failure loading some formulas (reported by panzerboy)
    * bailout iteration count is reset to 1000 when zooming in quickly after toggling fullscreen (reported by FK68)
    * guessing exterior gives poor quality images (reported by PrinceOfCreation)
    * crash when zooming too quickly near interior black regions (reported by Foxxie) (may have already been fixed in an earlier version)
    * speckles when rendering zoom out sequence (may have already been fixed in an earlier version)
  - library upgrades
    * upgrade to boost 1.74.0
    * upgrade to openexr 2.5.3
    * upgrade compiler to g++-mingw-w64 10.1.0-3+23 (Debian Bullseye/testing) and rebuild everything

- **kf-2.14.10.2** (2020-07-12)
  - fixes
    * store zoom out sequence was saving JPEG contents in PNG filenames sometimes
    * store zoom out sequence was forcing some kind of center reuse even when not requested (reported by saka and Azula)
    * store zoom out sequence was sometimes not saving metadata in image files correctly
    * setting bad SIMD vector size in KFS could crash (reported by FractalAlex)
    * setting window size/image size from KFS should be better behaved now
    * refactor default loading (now it loads default location kf.kfr from next to EXE, as well as default settings kf.kfs; defaults are only loaded when respective command line options are not given)
    * don't recalculate fractal when changing window size (reported by FK68)
    * fix buffer overflow crash in iterdiv -> string for GUI, also fix loss of precision (reported by FK68)
    * fix typos causing last few pixels to be corrupt in SIMD
    * remove annoying dialogs about derivatives (requested by gerrit)
  - library upgrades
    * upgrade to boost 1.73.0
    * upgrade to mpfr 4.1.0
    * upgrade to tiff 4.1.0
    * upgrade to glm 0.9.9.8
    * upgrade to openexr 2.4.2
  - library non-upgrades
    * don't upgrade to pixman 0.40.0 (https://gitlab.freedesktop.org/pixman/pixman/-/issues/43)

- **kf-2.14.10.1** (2020-06-01)
  - fixes
    * location dialog is smaller (by popular request)
    * inverse colour transition (reported by FractalAlex)
    * information dialog shortcut (reported by gerrit)
    * information dialog close (reported by gerrit)
    * information dialog OK button (reported by gerrit)
    * Newton progress backup autosave can be disabled (reported by gerrit)
    * can again set custom max references and series approximation terms (in new Perturbation and Series Approximation Tuning dialog) (reported by gerrit)
    * crash when loading default settings with non-standard image size
    * replaced custom buggy tooltip code with standard Windows controls
    * resume zoom sequence works with no recovery.kfb (reported by Azula)
    * crash when invoking resume zoom sequence while rendering in progress
- **kf-2.14.10** (2020-05-25)
  - speed boosts
    * Mandelbrot power 3+ and TheRedshiftRider formulas Newton zooming: twice as fast
    * Mandelbrot power 3 perturbation: 50% faster
    * Mandelbrot power 3 rescaled perturbation: 7.5% faster
    * "find glitch center using argmin|z|" is now parallelized (suggested by gerrit)
    * "find glitch center" (using original method) memory (de)allocation floated out of inner loop (slowness reported by gerrit)
    * OpenEXR usage is now optionally parallelized (enabled by default) (disabling reduces memory consumption and total work performed, but usually increases wall-clock time significantly)
  - enhancements
    * Show Inflection generalized to arbitrary power Mandelbrot (behaviour for power 2 is slightly different, but more correct)
    * better progress reporting for Newton-Raphson zooming (especially noticable for formulas other than Mandelbrot power 2)
    * use ball method for Newton-Raphson period finding by default (works much better than box method in skewed locations)
    * new setting "Save As -> Set EXR save channels" for choosing which channels to store in EXR files (for conserving disk space when not all of the data is needed in later processing)
    * default settings loaded from next to EXE file (`kf.exe` -> `kf.kfs`)
    * "Save KFR" option for Store Zoom Out Sequence, also `--save-kfr` command line flag (suggested by Fluoroantimonic_Acid) (note that metadata is already saved in all formats apart from KFB)
    * "Save KFR" is fast if no images will be saved (currently only fast in command line mode, and auto-iterations does not work because no images are rendered to be analysed)
    * split up Iterations dialog into Formula, Bailout and Information, with some controls going to the Advanced menu. keyboard shortcuts changed: Ctrl+F formula dialog, Ctrl+B bailout dialog, Ctrl+I information dialog (previously iterations dialog), Ctrl+Shift+F find center of glitch (previously Ctrl+F), Ctrl+Shift+B skew animation (previously Ctrl+B)
    * new p-norm options in the Bailout dialog, radius is tested against

      $$\sqrt[p]{|b_x|z_x|^p + b_y|z_y|^p|}$$

      (suggested by FractalAlex)

    * new Flat colouring mode for when disabling Smooth is not enough (Linear smooth method is recommended; Log can give seams)
    * "Random" method for choosing new references among glitched pixels, is sometimes better/worse than the "Original" and "argmin|z|" methods (suggested by gerrit)
    * Colors dialog has filename of last loaded file in its title (suggested by gerrit)
    * Fourth Root color method (iteration count transfer function) (ported from Mandel Machine, suggested by FractalAlex)
    * new phase channel (`T` in EXR) based on argument angle of final iterate
    * "Phase Strength" colouring control uses new phase channel ("chrome effect" colouring suggested by FractalAlex)
  - fixes
    * stopping rendering with ESC leaves image as-is, instead of recolouring from scratch (badly) (reported by CFJH)
    * status bar correctly resets when zooming before the previous image is completed
    * multiline widgets in location dialog fixes editing problems (reported by CFJH)
    * newton-raphson zooming increases maxiters more suitably
    * clamp crosshair window read region to screen size (prevent strobing display of uninitialized memory)
    * fix crash when zooming out by a huge custom zoom factor (2^25) (reported by CFJH) (divide by zero in `StretchBlt()`)
    * fix crashes when resizing window by dragging frame corner by setting a minimum size
    * glitches are detected with 2-norm always (regardless of p-norm)
    * series approximation uses low bailout radius (fixes far exterior)
    * store zoom out dialog does not proceed to ask for size when cancelled (reported by saka)
    * add prepare-msys.sh to release bundle (reported by PieMan597)
    * fixed -Wnarrowing warnings
    * fixed loading of power setting for fractal types 1-4 (e.g. power 5 burning ship now loads correctly) (bug found thanks to FractalAlex)
    * fixed texture option in colouring settings (broken for years) (reported by FractalAlex)
    * simplified by enforcing "reference pixel is never glitched" in a different way than duplicating pixel iterations in the reference code
    * NanoMB1+2 support directional DE
    * NanoMB1 iteration count was off-by-one compared to the regular way
    * EXR input and output uses much less memory as it no longer needs to copy so many of the data arrays; overhead for output is now 8 bytes per pixel (12 bytes for more than 4G iteration count images) plus 6 bytes per pixel if RGB is used (tip: enabling the half colour buffer preallocates this and saves an extra colouring pass)
    * "Fast" shrinking is used while rendering is still ongoing (prevents denial of service when updates are queued faster than they can be handled)
    * "Best" shrinking is always used when saving images
    * image size adjuster uses aspect ratio from window, instead of the current image size (prevents weird heights after setting width to odd values like 463) (reported by saka)
  - library upgrades
    * upgrade to openexr-2.4.1
    * upgrade to glm-0.9.9.7
    * upgrade to gmp-6.2.0
    * upgrade compiler to g++-mingw-w64 9.3.0-8+22 and rebuild everything

- **kf-2.14.9** (2019-12-31)
  - feature: new formula `z^2 exp(2 a / z) + c` (suggested by gerrit); set seed to `0` (default) for singular orbit; set seed to `a` for critical orbit (large `|a|` bugs at zooms <=

1e3); no Newton zooming yet; thread: https://fractalforums.org/f/28/t/3234
  - feature: use minimum `|z|` pixels for new references (suggested by quaz0r) (option disabled by default as it is not always faster)
  - feature: save file format version number with parameters and settings
  - feature: documentation improvements (references to third-party software and EXR map file semantics)
  - bugfix: don't set zoom size when resuming zoom out sequence (now you need to set it manually before resuming) (fixes issue where it was clamped to integer; reported by david)
  - bugfix: clamp half-precision colour output in EXR (fixes issue with NaNs)
  - bugfix: interpolate neighbourhood when ignoring isolated glitches (appearance should be smoother, especially analytic DE was broken before)
  - bugfix: avoid assertion failure when loading a parameter with fractal type out of range (instead unknown formulas appear incorrectly as Mandelbrot) (reported by gerrit)
  - bugfix: remove -pipe from Makefile to allow builds with smaller RAM
  - bugfix: updated Windows/MSYS2 build instructions (thanks to Patrick Owen)
  - upgrade to boost 1.72.0
  - upgrade to glm 0.9.9.6
  - update openexr repository url
  - update pixman build to skip demos
  - enable fat gmp build (with runtime CPU detection for faster reference calculations)
  - feature: experimental OpenCL support for GPU rendering (not built by default, enable it at build time with `make clean && make OPENCL=1`)

- **kf-2.14.8** (2019-09-23)
  - feature: SIMD support. Adjustable tuning parameters SIMD vector size (default 2) and chunk size (default 64) settings in the Advanced menu. Should speed up perturbation calculations in double precision (up to zoom depth 1e300 or so). SIMD is also supported in scaled double for Mandelbrot power 2 (zoom depth 1e600 or so) and power 3 (zoom depth 1e400 or so). SIMD is not yet implemented for scaled double with derivatives. Compile KF yourself with optimisations for your own specific CPU for a bigger speed boost.
  - feature: Newton-Raphson zooming completion estimate (number of steps)
  - feature: Newton-Raphson zooming progress backups (after each step)
  - feature: fewer required fields when loading location parameters (allows loading the NR-zoom backups)
  - feature: directional DE stored in EXR as `DEX` and `DEY` channels, and used for slopes colouring with `Analytic` differences (requires derivatives, fixes noisy texture when jitter is enabled)
  - feature: 64bit iteration count support (limit is now 9 exa-iterations up from 2 giga-iterations), use EXR for export. This feature is very experimental, I haven't had the patience to test it properly. Please report back success or failure.
  - note: periods are still 32bit and limited to 2 giga-iterations
  - bugfix: command line –save-exr now sets non-interactive mode flag.
  - note: built for 64bit only; 32bit hardware is obsolete by now.
  - note: distribution package compressed with 7-zip for smaller size.
  - upgrade to openexr 2.4.0
  - upgrade to mingw64 g++ 9.2.1-8+22~exp1 (Debian experimental) and rebuild everything

- **kf-2.14.7.1** (2019-08-30)
  - bugfix: iteration data channels are no longer incorrectly vertically flipped when saving and loading EXR files; they now should match the RGB orientation
  - feature: optionally disable threaded reference calculations
  - bugfix: the 20% performance regression vs kf-2.12 has been fixed
  - bugfix: "save overwrites" now saved in .kfs settings files

- **kf-2.14.7** (2019-08-29)
  - feature: save EXR (combined image with iteration map data and metadata) and open EXR (as settings, location or iteration map data). For motivation and specifications see https://fractalforums.org/f/11/t/3017.
  - feature: LogLog and ATan color methods (iteration count transfer functions) (ported from Mandel Machine, suggested by Dinkydau)
  - feature: option to control overwriting existing files with Save/Ctrl-S. With overwrite disabled by default, KF will save files with a timestamp before the .kfr extension, preventing accidental data loss (suggested by fractal). The previous behaviour can be restored in the Advanced menu.
  - feature: option to reduce thread count by 1 (suggested by saka)
  - bugfix: set thread background mode for improved responsiveness (reported by saka)
  - bugfix: remove annoying warning dialogs from store zoom out sequence (reported by saka)
  - bugfix: size status bar sections proportionally to window width (fixes accessibility issues with non-standard system font size) (reported by fractal)
  - bugfix: make Octave/Matlab scripts into function files for broader compatibility (usage example: `stratify("tile",4)`) (bug reported by Chronicler1701, fix suggested by gerrit).
  - new dependency: mingw-std-threads git
  - new dependency: ilmbase 2.3.0
  - new dependency: openexr 2.3.0
  - upgrade to boost 1.71.0
  - upgrade to gsl 2.6
- **kf-2.14.6.1** (2019-07-19)
  - bugfix: load palette after map in command line rendering so that the colours from the palette are actually used (reported by blob)
- **kf-2.14.6** (2019-07-18)
  - new feature: high quality image downscaling with anti-aliasing using the Pixman library
  - new feature: ball method for period detection in Newton-Raphson zoom
  - new feature: command line KFB map colouring
  - bugfix: image is refreshed from first calculated pixels, instead of 1% (reported by Fluoroantimonic_Acid)
  - bugfix: don't read Iteration Divisor from KFB map files (for historical reasons this is an integer in the file format, instead of the floating point value that it can take now in the rest of the program)
  - upgrade to boost 1.70.0
  - upgrade to mpfr 4.0.2p1
  - upgrade to png 1.6.37
  - upgrade to glm 0.9.9.5
  - upgrade to mingw64 g++ 8.3-win32 20190406 and rebuild everything
- **kf-2.14.5** (2019-03-20)
  - new feature: command line zoom sequence rendering with `-z NFRAMES` flag
  - new feature: low bailout (SmoothMethod: 2) (suggested by Dinkydau) https://fractalforums.org/f/28/t/2594/msg12801#msg12801
  - new feature: General Abs Quadratic formulas (inspired by Alef and gerrit) https://fractalforums.org/f/28/t/2653/msg13242#msg13242
  - bugfix: thin black line in exterior between iteration bands (reported by CFJH)
  - bugfix: malformed location no longer crashes (reported by Foxxie)
  - upgrade to mpfr 4.0.2
  - upgrade to glm 0.9.9.4
- **kf-2.14.4** (2019-01-18)

  - new feature: nanomb1 algorithm for power 2 Mandelbrot (experimental) (originally by knighty) (not quite as experimental as nanomb2, suggested by gerrit, requires exact period to be set by Newton or location dialog)
  - new feature: make interior checking for nanomb optional (new menu item)
  - new feature: RadiusScale setting for nanomb2 (only settable in .kfs so far, default value 0.1 may be safe but slow)
  - new feature: Mothbrot family of formulas (suggested by simon.snake) (not optimized yet, preliminary version for testing)
  - bugfix: incremental drawing looks much better with slopes and/or nde (there is a crosshair visible while rendering, which disappears when the image is finished, not sure why this happens)
  - upgrade to libpng 1.6.36
  - upgrade to libtiff 4.0.10
  - upgrade to libmpfr 4.0.1-p14
  - upgrade to boost 1.69.0

- **kf-2.14.3** (2018-10-29)
  - new feature: nanomb2 algorithm for power 2 Mandelbrot (experimental) (originally by knighty)
  - new feature: display Newton period (and set limit for nanomb2 algorithm) in Location dialog
  - new feature: Ctrl-Shift-W shortcut to set image size (suggested by gerrit)
  - bugfix: distinguish dialog titles for set window size and set image size
  - bugfix: fix corrupt images when zooming out from the default view
  - new feature: dialogs to en/disable derivatives when opening file with/out analytic DE; automatically enabling derivatives if needed in CLI mode (reported by gerrit)
  - default zoom size changed from 4 to 2 (press Ctrl-4 to reset to 4, or load a settings file with your preference) (suggested by gerrit, to avoid the dreaded "zoom size is not 2" warning when saving zoom out sequence).
  - merged changes from 2.13.11

- **kf-2.13.11** (2018-10-29)
  - bugfix: make preprocessor compatible with ghc-8.6 (MonadFail)
  - bugfix: fix translation with reuse reference enabled (reported by Dinkydau)
  - bugfix: don't add additional references if autosolve is disabled
  - upgrade to libpng 1.6.35

- **kf-2.14.2** (2018-10-23)
  - bugfix: initial window size was off by a few pixels
  - new feature: undo/redo (menu and Ctrl-Z, Ctrl-Y) (suggested by TwinDragon)
  - new feature: added copy/paste to menu (Ctrl-X, Ctrl-V worked already)
  - new feature: Ctrl-Shift-K shortcut to reset skew
  - new feature: Ctrl-Shift-T shortcut to reset rotation
  - bugfix: minor improvements to documentation
  - bugfix: fix series approximation derivative initialization
  - internal: refactor series approximation for Mandelbrot power > 2
  - merged changes from 2.13.10

- **kf-2.13.10** (2018-10-23)
  - bugfix: corrupt image at transition between number types (eg near e600) (reported by CFJH)
  - bugfix: changing "threads per CPU" during rendering crashes (reported by CFJH) (the menu is now disabled during rendering)
  - bugfix: set approx terms to 3 for other than power 2 Mandelbrot
  - bugfix: fix floatexp toString for negative values and 0
  - internal: update to et-kf API version 5

- **kf-2.14.1** (2018-09-27)
  - new feature: series approximation for Burning Ship power 2, using case analysis to know how it gets reflected at each iteration
  - two new formulas: General Quadratic Minus, General Quadratic Plus; "RedShiftRider factor A" sets coefficients (suggested by gerrit)
  - new feature: "Fast" preset activated by default
  - new feature: GUI menus reorganized to make more sense and updated the manual to reflect this
  - new feature: "check for updates" option in the file menu (requires internet access)
  - new feature: 1/4 and 1/2 threads per CPU settings (minimum 1 thread)
  - new feature: optional use of d/dZ in autoskew escape (suggested by gerrit)
- **kf-2.13.9** (2018-09-06)
  - new feature: auto skew without miniset: new button in Newton-Raphson zooming dialog, "Auto Skew (Escape)", that skews at the center pixels (algorithm suggested by gerrit)
  - bugfix: set window title on parameter drag and drop (reported by gerrit)
  - bugfix: reading PNG metadata works even if it is moved after the image data chunks and has a miscapitalized "Comment" keyword
  - bugfix: fix Mandelbar derivative calculations for ADE (reported by gerrit)
  - bugfix: fix 4th False Quasi formulas Newton-Raphson zooming
- **kf-2.13.8** (2018-08-28)
  - new feature: auto skew (via Newton-Raphson zooming dialog)
  - new feature: show/hide crosshair window (suggested by gerrit and others)
  - new feature: quality presets ("fast" but inaccurate, "best" but slow)
  - new feature: page up / page down keyboard shortcuts to zoom in / out
  - new feature: drag-and-drop parameter files from the file manager to the main window to open them
  - new feature: drag-and-drop palette files from the file manager to the color dialog window to open them
  - new feature: CLI prints total remaining pixels (suggested by gerrit)
  - bugfix: suppress error dialogs when loading metadata from TIFF
  - bugfix: remove obsolete .ini stuff that was overriding default settings
  - bugfix: enabled "no reuse center" by default (without it zoom out sequence sometimes glitches)
  - major documentation improvements
  - new dependency: GLM 0.9.9.0
- **kf-2.13.7** (2018-08-14)
  - export to uncompressed TIFF images (faster than PNG, but larger) (suggested by gerrit)
  - nicer default palette
  - upgrade to Boost 1.68.0
  - upgrade to MPFR 4.0.1-p13
- **kf-2.13.6** (2018-08-02)
  - fix for "a single dot appears in the middle of a mini" (reported by gerrit)
  - fix for "zooming out saves Zoom 0 in KFR parameter file" (reported by CFJH)
  - "open map" function (file menu) workflow: open KFR, ESC to abort, open KFB, adjust colours, save image (ignoring the warning about a KFB map existing with the same name)
- **kf-2.13.5** (2018-06-26)
  - colouring is parallelized for speedup when multiple cores are available

  - only sort pixel indices in interactive mode (speeds up batch mode)
  - palette loading fixed to load all colouring data
  - interior colour can be set now, defaults to black
  - upgrade to GSL 2.5

- **kf-2.13.4** (2018-06-21)
  - derivative computations can be switched off in the iterations dialog (increases rendering speed, breaks analytic DE colouring)

- **kf-2.12.13.1** (2018-06-04)
  - fix severe performance bug introduced in previous version
  - upgrade to Boost 1.67.0
  - upgrade to MPFR 4.0.1-p6

- **kf-2.12.13** (2018-05-15)
  - fix crasher bug relating to series approximation memory allocation (reported by gerrit)

- **kf-2.13.3** (2018-05-08)
  - Newton-Raphson zooming precision loss bug fix (reported by gerrit)
  - Newton-Raphson zooming for Mandelbrot powers 6 7 8 9 10 (reported by gerrit)
  - Newton-Raphson zooming size estimate fixed for more formulas (now correctly uses smallest power instead of largest, for renormalization behaviour near zero)
  - Mandelbrot power 6 7 8 9 10 analytic distance estimation bug fixes (reported by gerrit)

- **kf-2.13.2** (2018-05-01)
  - Newton-Raphson zooming for all formulas (using code generated by 'et') (no progress reporting yet, nor convergence detection for early exit)
  - R^2 matrix derivatives for all formulas (fixes analytic DE with skew)
  - fix image corruption when enabling guessing with analytic DE
  - fix assertion failed popup (race condition between parameter changes and colouring during rendering) (reported by gerrit)

- **kf-2.13.1** (2018-03-30)
  - added derivative calculations
  - known bug: calculated even if not needed for colouring
  - known bug: some derivatives are C approximations instead of R^2 matrix
  - analytic DE colouring mode
  - known bug: DE data not saved to KFB map files
  - interior is white instead of black
  - known bug: interior colour should be customizable
  - series approximation uses Horner's rule to evaluate polynomials (minor optimisation)
  - default settings are best-quality by default

- **kf-2.12.12** (2018-03-30)
  - Free Software under GNU Affero General Public License, version 3 or greater
  - least squares numerical differences (experimental; subject to change)
  - Laplacian numerical differences (suggested by gerrit)
  - (possibly stratified) tiled settings/parameters generator program
  - stratified tiling reassembler GNU Octave script

- **kf-2.12.11** (2018-03-12)
  - uniform jitter by default (suggested by gerrit)
  - Gaussian jitter available as an option
  - jitter scale factor setting (suggested by gerrit)
  - log verbosity flag for command line rendering
  - upgrade to libjpeg6b2 https://jpegclub.org/support/

  - fix bug where Ctrl-S would overwrite the wrong file (reported by gerrit)
  - fix bug where lines would appear aligned with references (reported by gerrit and CFJH)
  - fix "color method and infinite waves are immediately applied" even when colouring is disabled (reported by gerrit)

- **kf-2.12.10** (2018-02-22)
  - upgrade to MPFR 4.0.1
  - removed some build instructions from README (see the prepare.sh script) (it was too annoying to have to keep updating everything in two places)
  - command line non-interactive mode works without opening a window (suggested by gerrit)
  - pixel jitter (dithering of position): set non-zero seed in iterations dialog to enable it, different seeds give slightly different images which can be stacked later for anti-aliasing as an alternative to rendering large and downscaling later; helps reduce Moiré grid artifacts at the cost of noise
  - "Enable" toggle in colouring dialog: colouring takes long for large images, disable the toggle if you want to make many changes at once, then click "Apply" or re-enable to see the results (discussed with gerrit)

- **kf-2.12.9.1** (2018-01-24)
  - fixed broken "approx low tolerance" checkbox (thanks to Dinkydau)
  - fixed NR-zoom dialog to remember custom zoom factor between reopenings (reported by gerrit)
  - fixed initialization of some menu item state on load

- **kf-2.12.9** (2018-01-24)
  - improved Newton-Raphson zooming dialog, now zooms to a factor between current zoom and minibrot size (suggested by Dinkydau)
  - implemented "scaled long double" iterations for Mandelbrot power 2 and 3, which might speed up some zooms between e4900 and e9800
  - fixed "the maxiterations suddenly resets to some very low value" (reported by gerrit)
  - fixed "Newton iterates 100 times before failing", now fails fast if the C goes outside the target (reported by gerrit and Dinkydau, fix suggested by knighty)
  - fixed "Newton zooms to blank image", reported by gerrit, fix was to remove a check on the derivative being huge that was falsely reporting success
  - fixed "manual ApproxTerms gets reset to AutoApproxTerms value" (reported by gerrit)
  - barrier implementation yields if there are fewer CPUs than threads (fixes very slow Newton-Raphson zooming and reference calculations on dual-core laptop) (this is the behaviour of 2.12.5 and earlier, but conditional on number of cores)
  - upgrade to libpng 1.6.34
  - added shell script containing some of the build instructions from the documentation

- **kf-2.12.8** (2017-12-22)
  - fixed the fix for hang in normalisation (was generating bad images) (reported by gerrit with bug7.png)
  - restore ignoring isolated glitches
  - fixed DE spots bug (somehow the reference was being reset to the center of the screen sometimes without its pixel position being updated) (reported by gerrit)
  - fixed bugs with guessing and glitch status (only interpolate when the glitch status of both neighbours is the same) (caused lines when dragging the view, probably other badness too)
  - added logic to prevent too-huge image size being entered via the GUI (settings files are still not checked...)
  - clarified store zoom count user interface (reported by CFJH)
  - fixed store zoom out for deep zooms (bug reported by gerrit) (the issue was exceptions thrown by std::stod(), remember to catch them)

  - refactor auto iterations (issue reported by gerrit remains unresolved)
  - fixed hardcoded count in glitch correction
  - use Taylor intervals in ball-period method (code copied from knighty)

- **kf-2.12.7** (2017-12-07)
  - renabled guessing conditional on menu option (was disabled in 2.12.4 as enabling it made some random speckles, possibly due to a race condition) (requested by Kalles Fraktaler and Fractal universe)
  - added "threads per cpu core" setting
  - compiles clean with -Wwrite-strings
  - even lower resolution preview for more intensive locations (suggested by Foxxie) (implemented with Adam7-style interlacing with circular sorting)
  - copy (Ctrl-X) and paste (Ctrl-V) parameters from the system clipboard
  - fixed hang crash bug when normalizing smooth iteration values
  - use interval arithmetic ball-period method instead of box-period (speeds up Newton-Raphson zooming a bit) (idea from knighty and gerrit)

- **kf-2.12.6** (2017-11-24)
  - fix central differences (reported by gerrit)
  - fix insufficient precision in Zoom: saved in .kfr (reported by CFJH)
  - option to render zoom out sequence without saving KFB maps (suggested by CFJH)
  - option to stop rendering zoom out sequence after a certain number of frames (suggested by CFJH)
  - resume zoom sequence works without KFB maps saved every frame (still needs a "last.kfb", this is saved automatically when needed)
  - barrier no longer yields (fixes priority inversion on heavily loaded systems) (reported by gerrit)
  - switch from GMP mpf_t to MPFR (fixes some blank images on load, also some Newton-Raphson zoom failures - bug involved incorrect normalization) (reported by Kalles Fraktaler and gerrit)

- **kf-2.12.5** (2017-11-02)
  - preferences (rendering settings not related to location) save and load (.kfs files and PNG/JPEG comments too)
  - command line arguments to load settings and/or location
  - command line arguments to render and save PNG/JPEG/KFB Map before quiting
  - auto-added new references recalculate only all glitched pixels (in earlier versions it would recalculate all pixels with same integer iteration count, which may or may not have been glitched, and may have missed some glitches)
  - glitch correction now uses glitch flag instead of just iteration count (this ensures the reference is added in a really glitched pixel, so at least one pixel will be fixed by each reference, ensuring termination with a finite number of references)
  - fix bugs with references when calculating their own pixels
  - fix off-by-one when references fix their own pixels (reported by gerrit)
  - single pixel glitches are no longer fixed by copying neighbour
  - fixed glitch at image boundary correction
  - fixed memory leak in glitch correction
  - fixed horizontal line corruption in examine zoom sequence glitch correction (reported by Fractal universe)
  - fixed "Mandelbar Celtic" formula (reported by Kalles Fraktaler)
  - fixed "Mandelbar" formula (reported by Foxxie)
  - fixed "Burning Ship Power 4" formula (reported by Foxxie)
  - fixed complex formulas reference precision problems (reported by Foxxie)
  - fix for auto-iterations (now respects GUI) (reported by Foxxie)
  - fix for crash selecting invalid power (reported by Foxxie)
  - fix for gigantic zoom value bug in Newton zooming (reported by Foxxie and gerrit)

  - added “no reuse center” option to prevent rectangle pasting (suggested by quaz0r)
  - fix for signed integer overflow reports negative percentage in status bar when the number of iterations is large (reported by Foxxie and another)
  - fix for confusing PNG save options dialog called “JPEG properties”
  - major code refactoring into multiple files for ease of maintenance
  - delete no-longer-used single-threaded Newton-Raphson zooming code

- **kf-2.12.4** (2017-10-06)
  - “glitch low tolerance” checkbox that can be enabled to detect glitches more aggressively (enabling it allows “Olbaid-ST-023.kfr” to render correctly, but taking 16x longer than with it disabled) (incorrect render reported by Kalles Fraktaler)
  - updated program icon with transparent background and large version, and use it for child windows too
  - long double off-by-one bug fixed (incorrect render reported by CFJH)
  - floatexp implementation bug fixes (0.0 was implemented incorrectly) (caused a series approximation underskipping failure that was reported by Kalles Fraktaler)
  - fix distance colouring grid artifacts with small zoom size by disabling “reuse center” when zoom size is not an integer (reported by gerrit)
  - fix some iteration band edge artifacts (with external postprocessing of kfb map files) by increasing the smooth bailout radius from 100 to 10000 (reported by gerrit)
  - ensure added references fix their corresponding pixel (suggested by Pauldelbrot) (prevents loop in auto solve glitches whereby a reference was repeatedly being added at the same location without progress being made)
  - disabled “guessing” (was causing occasional randomly bright single pixels at low zoom levels, possibly a race condition?)
  - build against an installed libjpeg instead of each time after clean
  - fix broken complex formula reference calculations
  - fix compilation warnings (now almost clean with -Wall -Wextra)
  - compile as C++17 (should also still work as C++11 and C++14)
  - delete bitrotten code paths for SetEvent()-based multithreading
  - delete unused thread affinity setting code
  - parameter data is saved as comment in image files (both PNG and JPEG)
  - can load parameters from image file comments (both PNG and JPEG)
  - delete obsolete VS build system
  - fix “infinite waves” colouring (reported by gerrit)

- **kf-2.12.3** (2017-09-25)
  - multiple finite difference methods for distance colouring (suggested by gerrit)
  - fix bug in examine zoom sequence auto solve glitches (first frame only) (reported by Dinkydau and Fractal universe)
  - raise limit for maximum number of references from 199 to 10000 (default still 69, you can change it in iterations dialog)
  - settable number of references per pass for examine zoom sequence auto solve glitches (default still 10)

- **kf-2.12.2** (2017-09-20)
  - PNG image saving support using libpng and zlib;
  - JPEG default quality to 100 (was 99);
  - colouring uses floating point internally to reduce quantisation steps;
  - dithering at end of colouring to improve perceptual quality;
  - `formula.cpp` included in source zip so GHC is not needed unless changing formula code;
  - optimized `diffabs()` code: one test Burning Ship location is 7.5% faster;
  - preprocessor optimizes reference calculations by floating temporary variable (re)allocations out of the inner loops: one test Burning Ship location is 30% faster;

- **kf-2.12.1** (2017-09-19)
  - simplified version numbering;
  - built for 64bit (as before) and 32bit (new);
  - documentation improvements;
  - fix division by zero assertion failure in File -> Examine zoom sequence;
  - fix crash in File -> Examine zoom sequence with only 1 image file;
  - adjust distance colour modes for backwards compatibility;
- **kf-2.11.1+gmp.20170913**
  - revert incompatible de log vs sqrt colouring change, instead add a new Distance (Logarithm) colouring method #7;
  - documentation improvements;
  - limit maximum series approximation terms to 60 to try to fix overskipping with large images
- **kf-2.11.1+gmp.20170822**
  - bugfix preprocessor for abs() formulas
  - de colouring with log instead of sqrt
- **kf-2.11.1+gmp.20170820**
  - bugfix preprocessor for diffabs() formulas
- **kf-2.11.1+gmp.20170714**
  - disabled OpenCL (be more compatible)
- **kf-2.11.1+gmp.20170713**
  - optimized Newton-Raphson zooming (3x faster in one test)
- **kf-2.11.1+gmp.20170711**
  - workaround for WINE issue artificially limiting image size (now bitmaps up to 2GiB can be created on all platforms)
- **kf-2.11.1+gmp.20170710**
  - optimized formulas (reference calculation for quadratic Mandlebrot is much faster due to lower-level calls to gmp)
  - very experimental opencl support (mostly broken)
  - bugfixes (fix hang loading deep zoom locations, fix newton size in new view radius calculation, more complete library credits in documentation)
  - prune dead code (incomplete jpeg library deleted from source, complete version downloaded at build time as needed, delete rudimentary openmp support, delete non-performant barrier variant, delete slower-than-gmp mpfr support, delete custom floating point support)
- **kf-2.11.1+gmp.20170703**
  - formulas now generated at compile time from formula definition XML using XSL stylesheet
  - used fixed format floats instead of scientific
  - try to hide command prompt window on Windows
- **kf-2.11.1+gmp.20170508**
  - restored threaded reference calculations (reimplemented with barrier() semantics to avoid single-threaded WINE SetEvent() rendezvous)
- **kf-2.11.1+gmp.20170504**
  - removed threaded reference calculations (too much overhead)

  - miscellaneous code cleanups (no need for -fpermissive, const fixes, delete[] fixes, 64bit compatibility paranoia)

- **kf-2.11.1+gmp.20170406**
  - fixed precision bugs (easy deep zoom, interactive failure)
  - fixed performance bug with inflections
  - fixed cross-hair resource bug
  - added WINDRES argument to build system
  - added more info to about dialog
  - include source code with release

- **kf-2.11.1+gmp.20170330.1**
  - fixes a crasher bug in the previous version

- **kf-2.11.1+gmp.20170330**
  - unlimited precision
  - separate compilation

- **kf-2.11.1+gmp.20170313**
  - long double compiled into exe (no dll)

- **kf-2.11.1+gmp.20170307**
  - kf-2.11.1 + gmp

- **kf-2.9.3+gmp.20170307**
  - kf-2.9.3 + gmp

TODO

User Interface

- crosshair cursor with more contrast (suggested by CFJH)
- show box-region for nr-zoom before clicking (suggested by Foxxie)
- adjust the size of the box via slider or like shift_scroll wheel or something like that? (suggested by Foxxie for nr-zoom, could also be useful for ctrl-left-click zoom)
- undo history for calculation data (suggested by TwinDragon)
- online help within program (suggested by TwinDragon)
- save image now function (without waiting for calculations)
- command line: print total runtime (suggested by gerrit)
- log window for diagnostics/debugging
- two-phase parameter loading with validation (suggested by Pauldelbrot)
- window and image size presets (suggested by saka and lycium)
- presets menu could be populated from a folder of KFS settings files

Calculations

- increase ref count limit without restarting from scratch
- increase maxiters limit without restarting from scratch
- optimize series approximation and probe point stuff
- calculate series approximation in parallel with reference
- only store reference orbit after series approximation
- refine minibrot using interior distance estimates
- refine minibrot using boundary shrinking (calculate edges only)
- enhanced glitch detection methods for all formulas (knighty + gerrit)
- “find center of glitch” cycle between multiple selected methods
- “Hidden Mandelbrot” formula (suggested by FractalAlex and gerrit)

Newton-Raphson Zooming

- add zoom depth offset in addition to zoom depth factor to (eg) zoom closer to minibrot or embedded Julia
- zoom to Misiurewicz points (custom zoom factor, manual preperiod selection) (suggested by gerrit)
- properly debug huge zoom values from size estimate
- make it work better in hard-skewed locations (need to skew the box period coordinates?)
- improve stopping criterion for resuming (don't over-refine)

NanoMB

- automatic number type selection
- automatic number type promotion on overflow
- re-scaled number types
- SSA escape radius factor control (around 0.1 to 1.0)
- automatic period detection

Preprocessor

- flatten complex numbers to separate real and imaginary parts
- common subexpression elimination (share results, might be especially useful for large powers of complex numbers)
- automatically parallelize reference iterations

Colouring

- load/save palette to/from image
- rework entirely (now: 1024 colours with mandatory interpolation)
- implement Pauldelbrot's multiwave colouring
- colour cycling (suggested by blob)
- more flexible colouring with lighting/layers/etc (suggested by Fraktalist)
- import gradients from various formats (.ugr/.gradient, .map, .xml (flam3)) inspired by Mandel Meute + padleywood's gradient convert tool: https://fractalforums.org/f/11/t/2934
- exterior tile texturing using smooth iteration count and phase
- bailout mode that combines the best of linear and log (for exterior tiling)
- color phase offset control
- orbit traps (eg atom domain, cellular colouring, dancing points, ...)
- stripe average (and triangle inequality average)
- refactor transfer functions so iterdiv is scaled sensibly (suggested by FK68)
- exponential smoothing colouring algorithm (suggested by FractalAlex) (implementation described by Ron Barnett at https://user.xmission.com/pub/lists/fractdev/archive/fractdev.200003)
- entropy colouring (post-processing effect) (suggested by FractalAlex, matlab/octave implementation by gerrit at https://fractalforums.org/f/28/t/368/msg1884#msg1884)

Getting The Code

I distribute EXEs bundled together with the corresponding source code.

The latest source code is available from my git repository:

```
git clone https://code.mathr.co.uk/kalles-fraktaler-2.git
cd kalles-fraktaler-2
git checkout master       # for Karl's original upstream
git checkout claude       # for MINGW build system and bug fixes
git checkout kf-2.14      # old stable (bugfixes only)
git checkout kf-2.15      # for current development
```

```
git tag -l                # list available release tags
```

You also need `et` to generate the formula code for Newton-Raphson zooming:

```
git clone https://code.mathr.co.uk/et.git
```

This is outside the scope of this document at the moment, easiest is to download the release bundle and copy the `formula/generated` folder from the included src zip. Get in touch for help if you want to regenerate these sources yourself.

Building On Linux

Compiling KF for your own CPU is recommended for optimal performance. The performance boost can be significant, as the release EXEs are compiled for generic i686 and x86_64 but newer CPUs have additional instructions available. This is less important now that OpenCL is available, as this optimizes for your hardware at runtime.

Note: there is an upstream bug in the GCC compiler. On Debian the compiler is patched in recent versions. Patching it yourself is not hard, but it does take a long time and need about 30GB of disk space. Without a patched compiler, KF may crash in SIMD code due to 32-byte aligned moves with a 16-byte aligned stack. See https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=939559. Contact me if you need help to patch your compiler.

Build instructions for cross-compiling from GNU/Linux require about 10GB of disk space and good internet download speed (or patience). About 750MB of downloads including the chroot debootstrap step. If you have recent Debian you can skip the chroot step and install natively.

0. Setup Debian Buster chroot:

```
sudo mkdir ./vm
sudo debootstrap buster ./vm/
sudo mount proc ./vm/proc -t proc
sudo mount sysfs ./vm/sys -t sysfs
sudo cp /etc/hosts ./vm/etc/hosts
sudo chroot ./vm /bin/bash
cd
```

1. Install dependencies (inside the chroot if you made one):

```
dpkg --add-architecture i386
echo >> /etc/apt/apt.conf.d/30buster 'APT::Default-Release "buster";'
echo >> /etc/apt/sources.list 'deb http://deb.debian.org/debian bullseye main'
apt update
apt install \
  build-essential \
  cabal-install \
  cmake \
  ghc \
  git \
  libghc-parsec3-dev \
  libtool \
  lzip \
  m4 \
  p7zip \
  pkg-config \
  wget \
  wine32 \
  wine64 \
  wine-binfmt \
  xsltproc \
```

```
  zip
apt install -t bullseye \
  mingw-w64
apt install \
  pandoc \
  texlive-fonts-recommended \
  texlive-latex-recommended   # for PDF manual
mount binfmt_misc /proc/sys/fs/binfmt_misc -t binfmt_misc
update-binfmts --import /usr/share/binfmts/wine
```

For Ubuntu replace "wine32 wine64 wine-binfmt" with "wine" (but see note about build failures with some versions).

2. Compiling is possible with GCC (well-tested) and LLVM (experimental, supports ARM).

   For GCC, configure the system MinGW compilers to use win32 threading model (instead of posix). If you don't do this then you'll get mysterious weird behaviour (like zooming out resetting zoom to infinity or 0).

   Choose the manual win32 alternative for all of these, you can ignore failures for gfortran and gnat if they are not installed:

```
update-alternatives --set x86_64-w64-mingw32-g++ /usr/bin/x86_64-w64-mingw32-g++-win32
update-alternatives --set x86_64-w64-mingw32-gcc /usr/bin/x86_64-w64-mingw32-gcc-win32
update-alternatives --set x86_64-w64-mingw32-gfortran /usr/bin/x86_64-w64-mingw32-gfortran-w
update-alternatives --set x86_64-w64-mingw32-gnat /usr/bin/x86_64-w64-mingw32-gnat-win32
update-alternatives --set i686-w64-mingw32-g++ /usr/bin/i686-w64-mingw32-g++-win32
update-alternatives --set i686-w64-mingw32-gcc /usr/bin/i686-w64-mingw32-gcc-win32
update-alternatives --set i686-w64-mingw32-gfortran /usr/bin/i686-w64-mingw32-gfortran-win32
update-alternatives --set i686-w64-mingw32-gnat /usr/bin/i686-w64-mingw32-gnat-win32
```

   If you have existing builds build with posix threading model, you should delete them or move them out of the way, as mixing win32 with posix leads to a world of pain and misery (as does mixing LLVM with GCC).

3. Prepare non-root build user:

```
adduser build
# enter and confirm password
su - build
mkdir -p ~/win/src
```

   If compiling with LLVM, install the MinGW toolchain (needs ~15GB space):

```
mkdir -p ~/opt/src
cd ~/opt/src/
git clone https://github.com/mstorsjo/llvm-mingw.git
cd llvm-mingw/
./build-all.sh ~/opt
export PATH=${HOME}/opt/bin:$PATH
```

4. Download Kalles Fraktaler 2 + sources:

```
cd ~/win/src
git clone https://code.mathr.co.uk/kalles-fraktaler-2.git
cd kalles-fraktaler-2
git checkout kf-2.15
```

5. Download and build and install 3rd party library sources (inspect the script if you want to be sure it isn't doing anything dodgy, or to copy/paste parts if necessary), the script can build both 64bit and 32bit variants if necessary:

```
cd ~/win/src/kalles-fraktaler-2
bash ./prepare.sh dl
bash ./prepare.sh x86_64
bash ./prepare.sh i686
bash ./prepare.sh aarch64
bash ./prepare.sh armv7
```

6. Download the latest version of Kalles Fraktaler 2 + and copy the `et`-generated formulas from it:

```
cd ~/win/src/kalles-fraktaler-2
wget -c "https://mathr.co.uk/kf/kf-$(wget -q -O- https://mathr.co.uk/kf/VERSION.txt).7z"
7zr x kf-*.7z
cd kf-*/
unzip kf-*src.zip
cd kf-*-src/
cp -avit ../../formula/generated formula/generated/*.c
```

7. To build Kalles Fraktaler 2 + optimized for your own Intel/AMD 64bit CPU:

```
cd ~/win/src/kalles-fraktaler-2
make clean
make SYSTEM=x86_64+native -j $(nproc)
./kf.exe
```

8. To build Kalles Fraktaler 2 + for generic 32bit Intel CPU:

```
cd ~/win/src/kalles-fraktaler-2
make clean
make SYSTEM=i686 -j $(nproc)
./kf.exe
```

9. To build Kalles Fraktaler 2 + for generic 64bit CPU:

```
cd ~/win/src/kalles-fraktaler-2
make clean
make SYSTEM=x86_64 -j $(nproc)
./kf.exe
```

10. To build Kalles Fraktaler 2 + for generic 32bit ARM CPU (might not work due to a bug in GMP):

```
cd ~/win/src/kalles-fraktaler-2
make clean
make SYSTEM=armv7 -j $(nproc)
./kf.exe
```

11. To build Kalles Fraktaler 2 + for generic 64bit ARM CPU:

```
cd ~/win/src/kalles-fraktaler-2
make clean
make SYSTEM=aarch64 -j $(nproc)
./kf.exe
```

12. To build Kalles Fraktaler 2 + release (all architectures + documentation + source zip + everything 7z + signing):

```
cd ~/win/src/kalles-fraktaler-2
./release.sh $(git describe)
```

Note: build fails on Ubuntu 16.04.3 LTS (xenial):

```
$ make
x86_64-w64-mingw32-g++ -mfpmath=sse -xc++ -Wno-write-strings -pipe -MMD -g -O3 -ffast-math -I/home
```

```
In file included from fraktal_sft/CDecNumber.cpp:1:0:
fraktal_sft/CDecNumber.h:5:76: error: 'decNumber' was not declared in this scope
 typedef boost::multiprecision::number<boost::multiprecision::gmp_float<0>> decNumber;
                                                                            ^
fraktal_sft/CDecNumber.h:5:76: error: template argument 1 is invalid
fraktal_sft/CDecNumber.h:5:62: error: template argument 1 is invalid
 typedef boost::multiprecision::number<boost::multiprecision::gmp_float<0>> decNumber;
                                                              ^
fraktal_sft/CDecNumber.h:5:62: error: template argument 2 is invalid
fraktal_sft/CDecNumber.h:5:32: warning: 'typedef' was ignored in this declaration
 typedef boost::multiprecision::number<boost::multiprecision::gmp_float<0>> decNumber;
                               ^
...
$ x86_64-w64-mingw32-g++ --version
x86_64-w64-mingw32-g++ (GCC) 5.3.1 20160211
Copyright (C) 2015 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

The working Debian Buster with compiler from Bullseye (needed for the SIMD alignment fix mentioned above) has:

```
$ x86_64-w64-mingw32-g++ --version
x86_64-w64-mingw32-g++ (GCC) 10-posix 20200525
Copyright (C) 2020 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Building on Windows 64-bit (may be adaptable to 32-bit)

Note these instructions are out of date.

Build instructions for compiling on Windows (thanks to knighty and Patrick Owen!):

0. Remove any old version of MSYS2.

   These instructions assume you are using a fresh version of MSYS2. Depending on how confident you are, you may be fine skipping this step.

1. Download the latest version of MSYS2 from https://www.msys2.org/ (msys2-x86_64-20190524.exe).

2. Run it to install MSYS2. When the installation finishes, the MSYS2 shell should open.

   Note: Many files you interact with in future steps are in the installation directory of MSYS2. It is recommended that you install MSYS2 on a drive with plenty of space, as there will be about 10 gigabytes of files in there at the end of the installation process.

3. In the MSYS2 shell, invoke pacman:

   ```
   pacman -Syuu
   ```

   You will need to keep running this command until is says "there is nothing to do". While it's running, it may ask you to close the terminal. Follow these instructions. When you open the terminal again (see step 5), keep running the command.

4. Close the MSYS2 shell:

   ```
   exit
   ```

5. Reopen the MSYS2 shell (from startup menu, under the name MSYS2 MSYS).

6. Install mingw/gcc 64 bit (this will take a fair bit of time):

```
pacman -S mingw-w64-x86_64-toolchain
```

When it asks you to enter a selection, just hit enter (all).

7. Install required tools (this will likely take less time):

```
pacman -S --needed git patch make diffutils mingw-w64-x86_64-cmake lzip p7zip unzip
```

8. Close the MSYS2 shell then open "MSYS2 MinGW 64-bit" shell (instead of MSYS2 MSYS, in order to have all the environment variables properly set)

9. Install tar. The version of tar that comes installed with MSYS2 has trouble extracting xz files.

```
pacman -S tar
```

10. Restart MSYS2 again (Close and reopen "MSYS2 MinGW 64-bit").

11. Install ghc (Haskell)

```
# See https://gitlab.haskell.org/ghc/ghc/wikis/building/preparation/windows
curl -L https://downloads.haskell.org/~ghc/8.6.5/ghc-8.6.5-x86_64-unknown-mingw32.tar.xz | ta
```

12. Change directory to the kalles fraktaler sources (where `Makefile` resides).

    If this directory is outside of the msys64 directory, include the full Windows path, replacing backslashes with forward slashes and drive letters like `C:\` to paths like `/c/`.

13. Download and build and install 3rd party library sources (inspect the script if you want to be sure it isn't doing anything dodgy, or to copy/paste parts if necessary), the script will need to be modified to build the 32bit version:

```
bash ./prepare-msys.sh dl
bash ./prepare-msys.sh 64
```

    The first script is relatively fast depending on your internet speed, as it downloads dependencies. Depending on the speed of your machines, the second script takes about an hour to run.

    The script has several places where it pauses for a while with no output. You will know that it has finished successfully when it shows a bunch of lines starting with "– Installing:" and then exits, giving you back control of the command line.

14. Download the latest version of Kalles Fraktaler 2 + and copy the `et`-generated formulas from it:

```
cd "/path/to/Kalles Fraktaler 2 +"
wget -c "https://mathr.co.uk/kf/kf-$(wget -q -O- https://mathr.co.uk/kf/VERSION.txt).7z"
7zr x kf-*.7z
cd kf-*/
unzip kf-*src.zip
cd kf-*-src/
cp -avit ../../formula/generated formula/generated/*.c
```

    Feel free to delete `kf-*/` and `kf-*.7z` inside your git repository afterwards.

15. To build Kalles Fraktaler 2 + optimized for your own CPU:

```
cd "/path/to/Kalles Fraktaler 2 +"
make clean
WINDRES=windres make SYSTEM=x86_64+native -j $(nproc)
```

    Note that `mingw32-make` does not properly detect that files are already up-to-date, making building slow each time. Use `make` instead.

16. To build Kalles Fraktaler 2 + release:

    Follow the instructions for Linux, possibly making adaptions to allow them to work with MSYS2

17. Execute it this way from (MSYS2 MinGW 64-bit) command line:

    `./kf.exe`

    because it is linked dynamically to some libraries. In order to execute it from the explorer one needs to copy `libwinpthread-1.dll` from `msys64/mingw64/bin` next to the generated executable.

Legal

Kalles Fraktaler 2 +

- the QD library is used under the QD License https://sources.debian.org/src/qd/2.3.22+dfsg.1-3/COPYING/ https://sources.debian.org/src/qd/2.3.22+dfsg.1-3/BSD-LBNL-License.doc/
- the MINGW-STD-THREADS library is used under the conditions of the Simplified BSD License https://github.com/meganz/mingw-std-threads/blob/master/LICENSE

NOTE: If you redistribute the binaries or provide access to the binaries as a service, you must also be prepared to distribute the source corresponding to those binaries. To make this easier for you, the more recent zips include the source too (though you'll also need to get the third party library sources).

Library sources can be obtained via:

- boost
- clew
- glad
- glfw
- glm
- gmp
- gsl
- jpeg
- mingw-std-threads
- mpfr
- openexr
- pixman
- png
- tiff
- zlib

Acknowledgements

Thanks to:

- K.I.Martin for applying Perturbation and Series Approximation on the Mandelbrot set and generously sharing the theory and Java source code!
- Pauldelbrot for finding the reliable glitch detection method
- Botond Kósa and knighty for the extensions of Series Approximation
- laser blaster for the Burning ship formula
- stardust4ever for other fractal types
- claude for the Newton-Raphson method
- gerrit for the distance colouring differencing variations
- Dinkydau, Fractal universe, CFJH, Foxxie and others for reporting bugs
- Chillheimer for hosting my program

Claude also thanks Karl for releasing the source code and assigning a Free Software license.

User Manual

File

- **Open**

  Opens the current location from a parameter file (`*.kfr`) You can also load metadata from images saved by KF. You can also drag-and-drop files from the file manager on to the main window to open them as parameters.

- **Save**

  Saves the current location in the current parameter file (`*.kfr`)

- **Save as**

- **KFR** Saves the current location in a new parameter file (`*.kfr`)
- **PNG** Saves the current location in a PNG image file (`*.png`). The location and settings are saved in the file metadata.
- **JPEG** Saves the current location in a JPEG image file (`*.jpg`). The location and settings are saved in the file metadata.
- **TIFF** Saves the current location in a TIFF image file (`*.tif`). The location and settings are saved in the file metadata.
- **EXR** Saves the current location in a EXR image file (`*.exr`). The iteration data is also saved in the file by default. The location and settings are saved in the file metadata. This is the recommended format for saving iteration data.
- **Set EXR save channels** Choose which channels to save in EXR image files. This is stored in the settings, so you can use it with command line rendering.
- **KFB** Saves the iteration data in a map file (`*.kfb`). These files can be used by some third party programs. This file format is no longer recommended: use EXR instead. **The location and settings are not saved.**

Saved image dimensions can be smaller than the calculated image size set in the settings. The image will be downscaled according to the shrink quality setting. If the saved image dimensions are larger, the image will be recalculated at the new size.

Note that for EXR, the saved image dimensions are only for the "preview image", the full calculated image size is used for the main image and the iteration data.

- **Open map**

  Load the iteration data from map file (`*.kfb`, `*.exr`).

  Note: you must set the aspect ratio of the window to match the KFB data before opening the map file. If you have also saved images, you can do that by loading the image as a settings file.

  Note: if you want to continue zooming from the location, you must load the KFR file before opening the map file. You can also load a saved image as a location file. Location information is *not* stored in KFB files.

  Note: there is a historical accident whereby the iteration divider is saved in the KFB as an integer, losing any fractional part and sometimes resetting to 1 on load. A workaround is to load the palette from a KFP file (which is just a .kfr renamed to .kfp) after opening the map file.

- **Store zoom-out images**

  Zoom out automatically with the selected Zoom size and store JPEG/PNG/TIFF images and map file (`*.kfb`, `*.exr`) for each zoom out. The zoom out stops when the depth is lower than 1. The resulting files can be used by the KeyFrameMovie program to create a zoom-in animation.

- **Examine Zoom sequence**

  Make sure you store the end location as a kfr file in the same directory as you store the zoom sequence frames. This function allows you to examine the frames one by one and add references to remove eventual visible glitch blobs, or choose another pixel as the main reference.

- **Resume Zoom sequence**

  Make sure you store the end location as a kfr file in the same directory as you store the zoom sequence frames. This function allows you to resume and continue the zoom out sequnce, if

it got interrupted. You must set the zoom size factor to the correct value for the sequence before invoking this command.

- **Open Settings**

  Opens rendering settings from a settings file (`*.kfs`) You can also load metadata from images saved by KF.

- **Save Settings**

  Saves the current rendering settings to a settings file (`*.kfs`)

- **Check for update**

  Retrieves information from KF's homepage on the internet, to let you know if a newer version is available.

- **Exit**

  Exit this program

Fraktal

- **Presets**

  Set groups of settings to suggested preset values.

  - **Fast** accuracy may be compromised but it's fast for browsing. Sets ignore isolated glitch neighbourhood to 4, enables guessing, disables low tolerance for glitches and approximation, and disables derivatives computation unless analytic DE colouring is currently in use.
  - **Best** highest quality settings for important images, but slow. Sets ignore isolated glitch neighbourhood to 0 (disabled), disables guessing, enables low tolerance for glitches and approximation, and enables jitter with a default seed of 1.

- **Formula...**

  Adjust fractal type and power etc. See below.

- **Hybrid Formula...**

  Design your own hybrid fractal formula. See below.

- **Bailout...**

  Adjust iteration limit and escape radius, etc. See below

- **Location...**

  Displays the Location dialog where the coordinates for this location is displayed and can be edited.

- **Colors...**

  Adjust colouring. See below.

- **Information...**

  Display information about the fractal. See below.

- **Refresh**

  Render the current location

- **Cancel rendering**

  Cancel the current rendering

- **Reset**

  Set the location to the start point

- **Undo**

  Go back to previous location(s).

- **Redo**

  Go forward in the undo history.

- **Copy**

  Save the current location to the system clipboard.

- **Paste**

  Set the current location from the system clipboard.

View

- **Zoom size**

  Set the level of zoom, left mouse click to zoom in, right to zoom out

- **Animate soom**

  Turns animation on or off when zooming

- **Set window size**

  Set the size of the display window.

- **Set image size**

  Set the size of the internal image size. If this is larger than the window size, an anti-alias effect is achieved

- **Arbitrary size**

  When disabled, the window aspect ratio is locked to 16:9. When enabled, any aspect ratio is possible.

- **Transformation**

  Activate transformation dialog. Left mouse drag to rotate the image. Right mouse drag to skew the image. Automatic skew is available via the Newton-Raphson zooming dialog.

- **Reset Transformation**

  Clear any rotation and skew.

- **Exponential Map**

  Apply exponential map coordinate transform. For best (more conformal) results use a wide aspect ratio (9:1 works well, window size 1152x128).

  You can store exponential map zoom out sequences (set zoom size to 2) and combine them into a movie with the *zoomasm* zoom video assembler. This is much more efficient than storing flat keyframes.

- **Shrink quality**

  Set quality of image scaling. “Fast” and “Default” are better suited for exploring, but “Best” looks much better at the cost of slower colouring (it does properly filtered anti-aliasing).

  You shouldn’t need to adjust this unless you set the image size larger than the window size, or intend saving image files at smaller resolutions than the image size.

Note: this feature is still a bit experimental and subject to change in future versions.

Navigation

- **Newton-Raphson zooming**

  When activated, a dialog will be displayed, which allows you to select if the zoom should jump directly to the minibrot, or to 3/4 zooms to the minibrot, where the current pattern is doubled, etc.

  The zoom level of the current pattern is set when opening the dialog, and can be changed with the capture button (which gets the current zoom level from the image view).

  Click on the fractal to specify the start point of the search of the minibrot. The current zoom size is used to set the boundaries of search around the selected point. It is recommended to set the zoom size to 128 and use the crosshair window for pixel-accurate selection.

  Notice that it can take a long time to calculate the position of deep minibrots. However, that should be still much faster than zooming to the minibrot manually by selecting the center of the pattern in the view, or with the automatic search of minibrot that is also using the pattern center.

  When "auto skew (newton)" is enabled before activating, the view will be skewed to make features near the minibrot approximately circular.

  When "auto skew (escape)" is activated, the view will be skewed to make features in the current view approximately circular (without zooming).

  "Auto skew escape" has an option to take into account d/dz derivatives as well as d/dc, some fractals work better with it enabled, some without. Experiment and report back!

- **Find Minibrot**

  Starts an automatic zoom-in in the image's pattern center, until a Minibrot is found or if it fails to find the center.

  It's probably better to use Newton-Raphson zooming if possible.

- **Non exact find Minibrot**

  Makes the Find Minibrot function fail every 20 zoom-in, in order to gain depth automatically without ending up in a Minibrot

  Newton-Raphson zooming may be a more useful option.

- **Center cursor**

  Center the cursor to image's pattern center

- **Find center of glitch (Color)**

  Centers the mouse pointer over the glitch blob found, if any

- **Find highest iteration**

  Centers the mouse pointer over the pixel with the highest iteration

- **Show Inflection**

  Activate or deactivate display of Inflection

- **Show crosshair window**

  Display a small window that magnifies the area around the mouse cursor. Perfect for precisely picking particular pixels for zooming etc.

Advanced

- **Auto solve glitches**

  Turns the Auto solve glitches function on or off

- **Reference selection method.**

  Chooose alternative methods for finding center of glitches. May be faster/slower depending on location.

- **Solve glitch with near pixel method**

  Instead of re-render all pixels with the same iteration count value(color) only the connected pixels are re-rendered. On some locations other areas in the same view have the exact same iteration count values. These pixels may be correctly rendered and may be incorrect if re-rendered with another reference

- **Ignore isolated small glitches**

  When enabled, ignores single-pixel glitches by interpolating their value from neighbouring pixels. If the image size is very large, there may be a very large number of these tiny glitches, whose incorrect rendering may be invisible to the eye, and whose correct rendering may take forever.

- **Perturbation and Series Approximation Tuning**
  - **Glitch low tolerance**

    When checked, glitches are more likely to be detected. Disabling it can lead to bad images, but is faster.

  - **Series approximation low tolerance**

    When checked, series approximation is stricter. Disabling it can lead to bad images, but is faster.

  - **Approximation terms**

    Automatic (default, recommended) is based on number of pixels.

  - **Max references**

    Sets limit of secondary reference points for automatic glitch correction. There is a hard limit of 10000, which is also the default.

  - **Derivative glitch test**

    Uses derivatives for a more accurate glitch test which can speed up rendering. Only for power 2 Mandelbrot, not yet implemented in OpenCL. Can be disabled for the rare cases where it fails.

- **Set main reference**

  Let you click the image and select the main reference for the whole image. This can be useful when glitches appears on top of minibrots when the reference is outside this minibrot. The glitch pattern disappears from the minibrot if the main reference is selected inside the minibrot.

- **Add reference (Color)**

  Add a reference and re-calculates the pixels with the same iteration count as the reference. This is useful if the Auto solve glitches function fails to find and solve glitches in the image

- **Reuse reference**

Do not re-calculate the reference for further zooming. This can be useful when during automatic zoom-out and to test different reference points, but must not be used together with the Auto solve glitches function active.

Note: reuse reference cannot be used for zoom sequences in which the number type used for calculations changes (which happens near 1e9864 1e4932 1e616 1e308 for power 2 formulas). To avoid corrupt zoom out images when reuse reference is enabled, also enable "Use floatexp always" (deeper than 1e4932) or "Use long double always" (deeper than 1e308). This may slow down calculations. Alternatively, render the zoom out sequence in several segments, one for each number type (floatexp, scaled long double, long double, scaled double, double).

- **No reuse center**

  Don't paste the previous image in the middle when zooming out. Disabling this (ie, do reuse center) can be faster but can also lead to bad images.

- **Show iterations**

  Displays the image black-and-white with the pixels with the highest iteration as white and the pixels with the lowest iteration as black

- **Show smooth transition colors**

  Displays the image black-and-white representing the smoothing coefficient

- **Show glitches**

  When activated, glitches are displayed with a solid color

- **Mirror**

  mirrors the image around the x-axis. Can be used on the deeper half of a zoom sequence to a minibrot - but not too close to the minibrot and too close to the half...

- **No series approximation**

  Turns the Series approximation function on or off.

- **Use long double always**

  Use always the 80-bit long double hardware data type. This can solve some type of glitches

- **Use floatexp always**

  Use always the double mantissa/integer exponent data type. This probably only make the render slower

- **Use NanoMB1 (experimental)**

  For power 2 Mandelbrot only.

  Use knighty's experimental NanoMB1 algorithm for bivariate super- series-approximation. Calculations are done with floatexp always. A regular series approximation plus perturbation pass follows for glitch correction.

  It is required to set the period in the Location dialog before enabling NanoMB1. Using Newton zoom sets the period automatically.

  It is recommended to enable Reuse Reference after recalculating with NanoMB1. A new reference is calculated automatically after each Newton zooming, whatever the setting of Reuse Reference.

  Whether NanoMB1 is faster or not depends heavily on the location: views close to minis should be significantly faster than the regular 'fast' preset.

- **Use NanoMB2 (experimental)**

  For power 2 Mandelbrot only.

Use knighty's experimental NanoMB2 algorithm for bivariate super- series-approximation. Calculations are done with floatexp always, glitch detection and correction is disabled (and may be unnecessary?).

It is recommended to set the period limit in the Location dialog before enabling NanoMB2, otherwise it uses the maximum iteration count which may take significantly longer. Using Newton zoom sets the period limit automatically.

It is recommended to enable Reuse Reference after recalculating with NanoMB2. A new reference is calculated automatically after each Newton zooming, whatever the setting of Reuse Reference.

Whether NanoMB2 is faster or not depends heavily on the location: views close to minis should be significantly faster than the regular 'fast' preset.

The RadiusScale field in the .kfs settings file controls the scaling of the escape radius calculated for each minibrot in the chain. Increasing it may be faster but lead to visible distortion, decreasing it may help the distortion but slow things down. No GUI for this control yet.

- **Interior checking (NanoMB only)**

  For power 2 Mandelbrot rendered with NanoMB1 or NanoMB2 only.

  Use an interior checking algorithm, which may or may not speed up per-pixel calculations in locations with large interior regions visible.

  Experimental. Correctness is to be evaluated. Subject to change.

- **Use auto iterations**

  Turns automatic iteration control on or off. This is on per default.

- **Use guessing**

  Enable interpolation of neighbouring pixel data when the iteration count is the same. This speeds up rendering of interior regions and glitches.

- **'Save' overwrites existing file**

  When this is checked, Save (Ctrl-S) overwrites the current file without asking. When this is unchecked, Save will add a timestamp to file names to prevent accidental data loss.

- **SIMD vector size**

  Number of pixels to calculate simultaneously per core. Setting it too small or too large will lead to slower performance. The sweet spot will depend on your particular CPU model, cache sizes, how KF is compiled (e.g. 64 vs native), etc.

  Setting it to 1 disables the explicitly vectorized code path; some SIMD instructions may still be used depending on the compiler.

- **SIMD chunk size**

  Number of iterations to calculate between escape/glitch checkes. Setting it too small or too large will lead to slower performance, because checks will be done too often (small chunk size) or too many extra iterations will be done per pixel (large chunk size). The sweet spot will depend on your CPU and the location (if the number of iterations is high, probably a higher chunk size would be beneficial).

  How it works: the last chunk (when any of the SIMD vector escape to infinity or glitch detected) is repeated for each pixel in the vector individually, after rolling back to before the last chunk. This means some iterations are repeated, but the speedup from doing escape checks less frequently may overcome this issue.

- **Half-float image buffer**

Pre-allocate 16bit image buffer for EXR export. Otherwise it is only allocated when needed. May speed up EXR export if this is checked. If not using EXR, leave this unchecked.

- **Multi-threaded EXR input/output**

  Use multiple threads in EXR input/output. Speed and CPU efficiency (as well as memory usage) may differ when toggling this flag; evaluate the optimal choice for your image size and system.

- **Multi-threaded reference calcs**

  Use multiple threads when calculating reference orbits (available for long double and floatexp only). Speed and CPU efficiency may differ when toggling this flag; evaluate the optimal choice for your location and system.

- **Threads per CPU**

  The default of 1 is sensible if KF is the only thing you're running. Increase the count if you want KF to peg your system, decrease it if you want to do other things at the same time.

  The last menu item reduces the calculated number of threads by 1, which can help improve system responsiveness in some configurations.

  There will always be at least one thread.

  These options only affect perturbation rendering and image colouring, not Newton-Raphson zooming or reference calculations.

  Note: changing this during rendering could lead to crashes, so the menu is disabled during rendering to prevent that.

- **OpenCL**

  Select which device to use for OpenCL, or "(none)" to use the regular old implementation on CPU. Currently OpenCL is used for perturbation iterations only, and needs a device with double precision (fp64) support. On some operating systems you might need to adjust GPU timeout behaviour if the calculations take too long and are interrupted.

  OpenCL may or may not be faster, depending on device, location, etc. For zooms between ~1e300 and ~1e4900, the regular old CPU implementation is likely to be significantly faster because OpenCL does not support x87 long double (which has more range compared to double), and thus OpenCL has to use the very much slower floatexp number type.

About

At the very top right:

- **?**

  Open about dialog, with version information and credits.

  This also functions as a lock mechanism, preventing accidental zooming while a long render is taking place.

Status bar

Left

- D% pixels done with current reference.
- G% pixels guessed by neighboring interior.
- R% reference iterations vs maximum iteration count.
- A% series approximation skipped vs maximum iteration count.

Middle

- Zoom magnification (e.g. 1.23E45).
- Elapsed wall clock time.
- Number of references.
- Done marker.

Right

- Pixel coordinates of mouse cursor.
- Iteration count at mouse cursor.

Formula dialog

- Fractal type: Mandelbrot, Burning Ship, etc
- Power
- Seed: where to start iterating from (default `0 + 0 i`, for best semantics it should be a critical point of the iteration formula, where its `d/dz` derivative is zero).
- Factor A: set the complex number `a` (denoted `f = d + e i` in the formula list below) for TheRedshiftRider formulas.
- Derivatives: compute derivatives (required for Analytic distance estimation, and Analytic is recommended for slopes when jitter is enabled). Also enables a more accurate glitch detection criterion (for power 2 Mandelbrot only).
- Jitter seed: non-zero enables jitter with a pseudo-random-number generator seed value.
- Jitter scale in units of a pixel (1 pixel is sensible in most cases).
- Gaussian jitter is probably best left disabled (uniform jitter looks better).

Hybrid Formula dialog

- There are up to 4 formula groups you can enable. Enable them with a positive number of **Repeats** at the top right.

  Groups are executed in a loop, apart from a prefix which is executed only once. The **Start Loop** radio buttons on the right let you choose where the loop starts.

- Each group has up to 2 lines. Don't forget to enable with the checkbox at the right. Enabled groups *must* have their first line enabled.

  Lines are executed in sequence, with the '+ C' part of the formula happening at the end of all the lines.

- Each line has two operators which can be combined with `+`, `-`, or `*`.
- Each operator has controls (from left to right):
  - Abs X
  - Abs Y
  - Neg X
  - Neg Y
  - Power
  - A Real
  - A Imag

  Setting all of them gives `(-|x| + -|y| i)^p * (a_r + a_i i)`.

- The **OK** button first validates the dialog, if it doesn't pass validation it has the same effect as the **Cancel** button. This is not ideal...

- Hybrid formulas are significantly slower (up to 5x) than the built in formulas for the same formula, so use the built in ones if you can. Sometimes using OpenCL can reduce the slowdown.

Bailout dialog

- Number of iterations. Increase this if the interior is "blobby".
- Smooth method (new in 2.14.10, was previously linked to escape radius)
  - Log: use the renormalized smooth iteration count for Mandelbrot
  - Linear: interpolate between last two iterations (better for small escape radius and Flat colouring)
- Escape radius
  - High: large escape radius gives a smoother appearance.
  - Bailout=2: classic escape radius for power 2 Mandelbrot.
  - Low bailout: smallest valid escape radius for higher power Mandelbrot.
  - Custom: choose your own escape radius value (new in 2.14.10)
- Bailout Re+Im: modify these for special effects (default 1, 1)
- Norm power (new in 2.14.10)
  - 1: use abs norm (taxicab, Manhatten)
  - 2: regular Euclidean norm
  - Infinity: use maximum norm (special case)
  - Custom: choose your own power

  The value compared to the escape radius when testing for escape is `pow(abs(b_x * pow(abs(z_x), p) + b_y * pow(abs(z_y), p)), 1/p)`.

Colors dialog

The Colors dialog is accessed via the Fraktal menu, or via the keyboard shortcut `Ctrl+C` in the main window.

- **Number of key colors**

  Set the number of key colors between 1 and 1024.

- **Divide iteration**

  Divide each iteration number with this value, for dense images this value can be greater than 1. For DE, values less than 1 can be useful.

- **Color offset**

  Offset the colors in the palette

- **Random**

  Fill the palette with random colors made from the Seed value. The Seed button select a seed value randomly.

- **More contrast**

  Move RGB values closer to max or min

- **Less contrast**

  Move RGB values closer to the middle

- **Show slopes**

  Enable slope encoding for 3D effect.

  First value is the magnification of the slopes. The start value of 100 is suitable for the unzoomed view. Deep views requires a couple of magnitudes higher value.

  The second value is the percentage with which the slope encoding is applied on the coloring. 100 is max, however flat areas will still have the palette color visible.

- **Save palette**

  Save the current palette in KFP (`*.kfp`) file

- **Open palette**

  Load palette from a KFP (`*.kfp`) file

  Palettes can also be loaded via drag and drop.

- **Expand double**

  Double the number of key colors without changing the palette. This allows finer control of individual colors without changing the palette for other colors

- **Expand all**

  Increase the number of key color to maximum 1024 without changing the palette

- **Double**

  Double the key colors by repeating them

- **Merge Colors**

  Allows a selected color to be merged to every specied key color

- **Show index**

  Capture the mouse, hover the mouse over the fractal image and the corresponding color in the list will be highlighted. Click and the color selection dialog will be displayed for the active color

- **Smooth color transition**

  Makes the transitions of colors smooth

- **Inverse smooth color transition**

  Inverse the smooth color transition which makes edges more visible

- **Unnamed dropdown box**

  Specifies handling of the iteration count values prior to coloring

- **Palette waves**

  The palette can be filled from sine waves applied on Red, Green, Blue and Black-and-white. Each input box specifies the number of periods applied on the number of key colors in the palette. If the input box is left empty, no wave of this color is applied. At right of each input box the “P”-button makes the number you entered prime, since different prime numbers probably give more variation. The last input box specifies the waves offset.

  The button “Generate” applies the waves on the palette, the “Seed” button fills the fields with random values

- **Infinite waves**

  Waves can be applied on Hue, Saturation and Brightness rather than RGB values. The Period value specifies the length of the period (not the number of periods as for the Palette waves). Periods with prime numbers should be able to produce an infinite number unique colors

  A negative value on Hue, Saturation or Brightness makes a flat percentage value to be applied on all iterations.

- **Texture**

  A background image can be loaded, which is distorted by a slope effect. Only BMP, JPEG, GIF files can be loaded so far.

  Privacy note: the full file system path to the texture file is saved in the parameter files (including saved image metadata).

- **OpenGL**

  Use an OpenGL Shader Language (GLSL) snippet for colouring the image. See below. The shader source code is saved inside the palette and parameter files (including saved image metadata).

OpenGL dialog

The OpenGL dialog is accessed via the OpenGL button in the bottom left of the Colors dialog, or via the keyboard shortcut `Ctrl+G` in the main window.

- **GLSL**

  Enter OpenGL shader language source code for colouring snippet. See below for API.

- **Default**

  Reset colouring GLSL to default (emulating the regular CPU-based colouring algorithm).

- **Import**

  Load GLSL fragment from a text file (`*.glsl`).

  GLSL fragments can also be imported via drag and drop.

- **Export**

  Export GLSL fragment to a text file (`*.glsl`).

- **Use sRGB**

  Convert input and output from/to the sRGB colour space (which is non-linear, to get better perceptual accuracy). In this mode the shader operates in linear light.

- **Enable**

  Use OpenGL shader colouring instead of CPU colouring.

- **Apply**

  Apply changes, recompiling the shader.

- **Close**

  Cancel any unapplied changes and close the window.

- **OpenGL Version**

  The version of OpenGL in use is shown here, typically the highest supported version on your hardware and operating system + drivers. Shaders should use `__VERSION__` to adapt to the environment.

OpenGL Shader Language API

OpenGL Shader Language basic syntax is similar to C or C++ (braces and semicolons). KF provides a number of preset uniform variables and a small library of functions which you can use to access the raw iteration data in the neighbourhood of a pixel (random access is not possible due to the tiled rendering used for larger images). KF's default colouring algorithm (and modularized subparts) are also available for use in custom code.

The GLSL fragment should implement a function with this name and type signature:

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

The default colouring can be emulated by using `KF_Colour`:

```
vec3 colour() { return KF_Colour(); }
```

There should be no `#version` line in the GLSL fragment, instead `__VERSION__` should be used to adapt to the environment.

`float49` numbers

To avoid precision loss when representing iteration counts, a `float49` type is available which has 49 bits, compared to only 24 bits for regular float. GLSL does not have operator overloading (only function overloading) so you can manipulate them with these functions:

```
add neg sub mul inv div sqr ldexp
is_zero is_one is_positive is_negative
gt lt ge le eq ne abs max
fmod divrem drem nint aint floor ceil
exp log log10 pow sqrt nroot
sin cos tan sincos asin acos atan
sinh cosh tanh sincosh asinh acosh atanh
```

Some constants are also available:

```
f49_nan f49_inf f49_0 f49_1 f49_e f49_log2 f49_log10
f49_2pi f49_pi f49_3pi4 f49_pi2 f49_pi4 f49_pi16
f49_eps f49_min_normalized f49_max
```

You can convert a `float49` to `float` by `f.x[0]`, and to `double` by `double(f.x[0]) + double(f.x[1])`.

Accessing pixel data

The following functions get data for the current pixel, or for a nearby one if an `ivec2` offset is passed. The maximum absolute useable pixel offset is in `uniform ivec2 KFP_TilePadding`. See the documentation of the exported EXR channels for semantics.

`getGlitch` Returns `true` when the current (or offset) pixel is glitched.

`getInterior` Returns `true` when the current (or offset) pixel is interior.

`getN1` Returns the `uint` most significant 32bits of the iteration count of the current (or offset) pixel.

`getN0` Returns the `uint` least significant 32bits of the iteration count of the current (or offset) pixel.

`getNF` Returns the `float` smooth part of the iteration count of the current (or offset) pixel.

getN Returns a `float49` of the current (or offset) pixel's smooth iteration count, calculated by `(N1 << 32) + N0 + (1.0 - NF)` in 49 bits of precision.

getT Returns a `float` of the current (or offset) pixel's phase of final iterate (normalized to [0..1)).

getDEX Returns a `float` of the current (or offset) pixel's horizontal analytic distance estimate (normalized to 1.0 being a neighbouring pixel's boundary).

Requires derivatives to be enabled in the Formula dialog.

getDEY Returns a `float` of the current (or offset) pixel's vertical analytic distance estimate (normalized to 1.0 being a neighbouring pixel's boundary).

Requires derivatives to be enabled in the Formula dialog.

getDE Returns a `vec2` combining the current (or offset) pixel's analytic `DEX` and `DEY`.

Requires derivatives to be enabled in the Formula dialog.

getZoomLog2() The base-2 logarithm of the current (or offset) pixel's zoom level. Unzoomed is 0.0, deeper zooms are larger.

See `palettes/claude/text-overlay.glsl` for an example using this to display text.

inImage Returns `true` when the provided offset relative to the current pixel (not absolute coordinates!) lies inside the image bounds. The bounds can be found in `uniform ivec2 ImageSize`.

getCoord Get output coordinates, relative to the uniform variable `ImageSize`. Origin is bottom left per OpenGL conventions.

getJitter Return the `vec2` jitter delta of the current (or offset) pixel. Uses the uniform variables `uint KFP_JitterSeed`, `int KFP_JitterShape` and `float KFP_JitterScale`.

Not available in *zoomasm*. Numerical (non-analytic) distance estimation also cannot use jitter. Analytic DE with derivatives is recommended.

getN3x3 Fills three `mat3` with the relative iteration counts and pixel positions (offset plus jitter) of a 3x3 stencil centered on the current pixel. Used for numerical distance estimation. Returns the center pixel's `float49 N`.

Distance estimation

A 3x3 stencil of relative iteration counts can be used for numerical distance estimation.

KF_DE(int method) Returns numerical distance via a differencing method, one of the following `#define`d constants:

```
Differences_Traditional
Differences_Forward3x3
Differences_Central3x3
Differences_Diagonal2x2
Differences_LeastSquares2x2
Differences_LeastSquares3x3
Differences_Laplacian3x3
Differences_Analytic
```

Note: `Differences_LeastSquares2x2` and `Differences_LeastSquares3x3` are not implemented in GLSL, and `Differences_Analytic` uses the `DEX` and `DEY` channels which require derivatives to be enabled in the Formula dialog.

`KF_Traditional()` Return DE via KF's original implementation.

`KF_Forward3x3()` Return DE via 3x3 forward differences.

`KF_Central3x3()` Return DE via 3x3 central differences.

`KF_Diagonal2x2()` Return DE via 2x2 diagonals, aka Robert's Cross.

`KF_Laplacian3x3()` Return DE via 3x3 Laplacian.

`KF_Analytic()` Return DE via derivatives.

Accessing colouring parameters

The following `uniform` variables are set from the Colors dialog:

`ivec2 ImageSize` The size of the final image.

`sampler1D KFP_Palette` The key colours of the palette. The function `KF_Palette` interpolates the palette emulating KF's regular implementation, input is in [0..1).

`vec3 KFP_InteriorColor` The interior colour.

`bool KFP_ShowGlitches` Whether the menu option Advanced -> Show Glitches is selected.

`uvec2 KFP_Iterations` The iteration count. `[0]` contains the least significant word, `[1]` the most significant word.

`uvec2 KFP_IterationsMin` The minimum iteration count achieved in the image.

`uvec2 KFP_IterationsMax` The maximum escaped iteration count achieved in the image.

`float KFP_IterDiv` Divide iterations.

`float KFP_ColorOffset` Color offset.

`bool KFP_Smooth` Smooth colouring.

`bool KFP_Flat` Flat colouring

`bool KFP_InverseTransition` Invert colour transition. Can be applied to `float49` iteration counts with the function `KF_InverseTransition()`.

`int KFP_Differences` Numerical differencing method. See above.

`int KFP_ColorMethod` Color method, one of the following `#defined` constants:

```
ColorMethod_Standard
ColorMethod_SquareRoot
ColorMethod_CubicRoot
ColorMethod_Logarithm
ColorMethod_Stretched
ColorMethod_DistanceLinear
ColorMethod_DEPlusStandard
ColorMethod_DistanceLog
ColorMethod_DistanceSqrt
ColorMethod_LogLog
ColorMethod_ATan
ColorMethod_FourthRoot
```

The `float49` iteration counts can be transformed by the ColorMethod (including distance estimation, IterDiv, ColorOffset, InverseTransition) can be applied with the function `KF_IterTransform()`

`float KFP_PhaseColorStrength` Phase color strength.

Slopes

`bool KFP_Slopes` Whether slopes are enabled.

`float KFP_SlopePower` Slope power.

`float KFP_SlopeRatio` Slope ratio.

`vec2 KFP_SlopeDir` The cosine and sine of the slope angle, suitable for dot products.

`vec4 KF_Slopes(bool Analytic, vec2 SlopeDir, float Power, float Ratio)` This API function implements slope colouring emulating KF's default colouring. The return value `s` can be combined with the background colour `bg` with `mix(bg, s.rgb, s.a)`;

Infinite waves

There are up to `KFP_MultiWavesCountMax` waves (a constant, currently 32):

`bool KFP_MultiWavesEnabled` Whether waves are enabled.

`bool KFP_MultiWavesBlend` Whether waves should be blended.

`int KFP_MultiWavesCount` How many waves are added.

`ivec3 KFP_MultiWaves[KFP_MultiWavesCountMax]` The wave data. The `ivec3` contains period `.x` and type `.z`, the `.y` coordinate is currently not used. Only the first `KFP_MultiWavesCount` slots are filled.

`vec3 KF_InfiniteWaves(bool Smooth, float49 N)` This API function implements infinite waves emulating KF's default colouring.

Image texture

`bool KFP_TextureEnabled` Whether image texture is enabled.

`sampler2D KFP_Texture` The texture.

`float KFP_TextureMerge` Image texture merge.

`float KFP_TexturePower` Image texture distortion power.

`float KFP_TextureRatio` Image texture distortion ratio.

`vec2 KF_TextureWarp(float TexturePower, float TextureRatio, vec2 SlopeDir)` This API function implements image texture distortion emulating KF's default colouring. Example usage:

```
if (KFP_TextureEnabled)
{
  vec2 tc = getCoord() + KF_TextureWarp(KFP_TexturePower, KFP_TextureRatio, KFP_SlopeDir);
  tc /= vec2(ImageSize.xy);
  s = mix(s, texture(KFP_Texture, tc).rgb, KFP_TextureMerge);
}
```

Information dialog

- Minimum iteration count achieved in the image (display only).
- Maximum iteration count achieved in the image (display only).
- Series approximation iteration count (display only).
- Series approximation terms in use (display only).
- Calculations per second (display only).

Formulas

Notation:

```
i            -- imaginary unit (square root of -1)
p            -- integer power between 2 and 5 (10 for Mandelbrot)
c = a + i b  -- pixel coordinates (parameter plane)
z = x + i y  -- iteration variable
w = u + i v  -- temporary variable for two-stage formulas
f = d + i e  -- constant 'a' for TheRedshiftRider formulas
l m ...      -- juxtaposition is multiplication
^            -- raise an expression to a positive integer power
|.|          -- surrounding a real-valued expression: absolute value
```

Formulas:

- Mandelbrot

  ```
  z := z^p + c
  ```

- Burning Ship

  ```
  z := (|x| + i |y|)^p + c
  ```

- Buffalo

  ```
  w := z^p
  z := (|u| + i |v|) + c
  ```

- Celtic

  ```
  w := z^p
  z := (|u| + i v) + c
  ```

- Mandelbar

```
z := (x - i y)^p + c
```

- Mandelbar Celtic

```
w := (x - i y)^2
z := (|u| + i v) + c
```

- Perpendicular Mandelbrot

```
z := (|x| - i y)^2 + c
```

- Perpendicular Burning Ship

```
z := (x - i |y|)^2 + c
```

- Perpendicular Celtic

```
w := (|x| - i y)^2
z := (|u| + i v) + c
```

- Perpendicular Buffalo

```
w := (x - i |y|)^2
z := (|u| + i v) + c
```

- Cubic Quasi Burning Ship

```
z := (|x| (x^2 - 3 y^2) - i |y (3 x^2 - y^2)|) + c
```

- Cubic Partial BS Real

```
z := (|x| (x^2 - 3 y^2) + i y (3 x^2 - y^2)) + c
```

- Cubic Partial BS Imag

```
z := (x (x^2 - 3 y^2) + i |y| (3 x^2 - y^2)) + c
```

- Cubic Flying Squirrel (Buffalo Imag)

```
z := (x (x^2 - 3 y^2) + i |y (3 x^2 - y^2)|) + c
```

- Cubic Quasi Perpendicular

```
z := (|x| (x^2 - 3 y^2) - i y |3 x^2 - y^2|) + c
```

- 4th Burning Ship Partial Imag

```
z := (x + i |y|)^4 + c
```

- 4th Burning Ship Partial Real

```
z := (|x| + i y)^4 + c
```

- 4th Burning Ship Partial Real Mbar

```
z := (|x| - i y)^4 + c
```

- 4th Celtic Burning Ship Partial Imag

```
w := (x + i |y|)^4
z := (|u| + i v) + c
```

- 4th Celtic Burning Ship Partial Real

```
w := (|x| + i y)^4
z := (|u| + i v) + c
```

- 4th Celtic Burning Ship Partial Real Mbar

```
w := (|x| - i |y|)^4
z := (|u| + i v) + c
```

- 4th Buffalo Partial Imag

```
w := z^4
z := (u + i |v|) + c
```

- 4th Celtic Mbar

```
w := (x - i y)^4
z := (|u| + i v) + c
```

- 4th False Quasi Perpendicular

```
z := ((x^4 + y^4 - 6 x^2 y^2) - i 4 x y |x^2 - y^2|) + c
```

- 4th False Quasi Heart

```
z := ((x^4 + y^4 - 6 x^2 y^2) + i 4 x y |x^2 - y^2|) + c
```

- 4th Celtic False Quasi Perpendicular

```
z := (|x^4 + y^4 - 6 x^2 y^2| - i 4 x y |x^2 - y^2|) + c
```

- 4th Celtic False Quasi Heart

```
z := (|x^4 + y^4 - 6 x^2 y^2| + i 4 x y |x^2 - y^2|) + c
```

- 5th Burning Ship Partial

```
z := (|x| + i y)^5 + c
```

- 5th Burning Ship Partial Mbar

```
z := (|x| - i y)^5 + c
```

- 5th Celtic Mbar

```
w := (x - i y)^5
z := (|u| + i v) + c
```

- 5th Quasi Burning Ship (BS/Buffalo Hybrid)

```
w := (|x| + i y)^5
z := (u - i |v|) + c
```

- 5th Quasi Perpendicular

```
z := (|x| (x^4 + 5 y^4 - 10 x^2 y^2) - i y (|5 x^4 + y^4 - 10 x^2 y^2|)) + c
```

- 5th Quasi Heart

```
z := (|x| (x^4 + 5 y^4 - 10 x^2 y^2) + i y (|5 x^4 + y^4 - 10 x^2 y^2|)) + c
```

- SimonBrot 4th

```
z := z^2 (|x| + i |y|)^2 + c
```

- 4th Imag Quasi Perpendicular / Heart

```
z := ((x^4 + y^4 - 6 x^2 y^2) + i 4 x |y (x^2 - y^2)|) + c
```

- 4th Real Quasi Perpendicular

```
z := ((x^4 + y^4 - 6 x^2 y^2) - i 4 y |x (x^2 - y^2)|) + c
```

- 4th Real Quasi Heart

```
z := ((x^4 + y^4 - 6 x^2 y^2) + i 4 y |x (x^2 - y^2)|) + c
```

- 4th Celtic Imag Quasi Perpendicular / Heart

```
z := (|x^4 + y^4 - 6 x^2 y^2| + i 4 x |y (x^2 - y^2)|) + c
```

- 4th Celtic Real Quasi Perpendicular

```
z := (|x^4 + y^4 - 6 x^2 y^2| - i 4 y |x (x^2 - y^2)|) + c
```

- 4th Celtic Real Quasi Heart

```
z := (|x^4 + y^4 - 6 x^2 y^2| + i 4 y |x (x^2 - y^2)|) + c
```

- SimonBrot 6th

```
z := z^3 (|x| + i |y|)^3 + c
```

- HPDZ Buffalo

```
z := (((x^2 - y^2) - |x|) + i (|2xy| - |y|)) + c
```

- TheRedshiftRider 1: `a*z^2+z^3+c`

```
z := (f z^2 + z^3) + c
```

- TheRedshiftRider 2: `a*z^2-z^3+c`

```
z := (f z^2 - z^3) + c
```

- TheRedshiftRider 3: `2*z^2-z^3+c`

```
z := (2 z^2 - z^3) + c
```

- TheRedshiftRider 4: `a*z^2+z^4+c`

```
z := (f z^2 + z^4) + c
```

- TheRedshiftRider 5: `a*z^2-z^4+c`

```
z := (f z^2 - z^4) + c
```

- TheRedshiftRider 6: `a*z^2+z^5+c`

```
z := (f z^2 + z^5) + c
```

- TheRedshiftRider 7: `a*z^2-z^5+c`

```
z := (f z^2 - z^5) + c
```

- TheRedshiftRider 8: `a*z^2+z^6+c`

```
z := (f z^2 + z^6) + c
```

- TheRedshiftRider 9: `a*z^2-z^6+c`

```
z := (f z^2 - z^6) + c
```

- SimonBrot2 4th

```
w := z^2
z := w (|u| + i |v|) + c
```

- General Quadratic Minus

```
z := ((x^2 - y^2) + i (2 d x y + e x^2)) + c
```

- General Quadratic Plus

```
z := ((x^2 + y^2) + i (2 d x y + e x^2)) + c
```

- Mothbrot 2nd 1x1

```
z := z^1 (|x| + i |y|)^1 + c
```

- Mothbrot 3rd 1x2

```
z := z^1 (|x| + i |y|)^2 + c
```

- Mothbrot 3rd 2x1

```
z := z^2 (|x| + i |y|)^1 + c
```

- Mothbrot 4th 1x3

  ```
  z := z^1 (|x| + i |y|)^3 + c
  ```

- Mothbrot 4th 2x2 (aka SimonBrot 4th)

  ```
  z := z^2 (|x| + i |y|)^2 + c
  ```

- Mothbrot 4th 3x1

  ```
  z := z^3 (|x| + i |y|)^1 + c
  ```

- Mothbrot 5th 1x4

  ```
  z := z^1 (|x| + i |y|)^4 + c
  ```

- Mothbrot 5th 2x3

  ```
  z := z^2 (|x| + i |y|)^3 + c
  ```

- Mothbrot 5th 3x2

  ```
  z := z^3 (|x| + i |y|)^2 + c
  ```

- Mothbrot 5th 4x1

  ```
  z := z^4 (|x| + i |y|)^1 + c
  ```

- Mothbrot 6th 1x5

  ```
  z := z^1 (|x| + i |y|)^5 + c
  ```

- Mothbrot 6th 2x4 (Simon's Mothbrot)

  ```
  z := z^2 (|x| + i |y|)^4 + c
  ```

- Mothbrot 6th 3x3 (aka SimonBrot 6th)

  ```
  z := z^3 (|x| + i |y|)^3 + c
  ```

- Mothbrot 6th 4x2

  ```
  z := z^4 (|x| + i |y|)^2 + c
  ```

- Mothbrot 6th 5x1

  ```
  z := z^5 (|x| + i |y|)^1 + c
  ```

- General Abs Quadratic Minus

  ```
  z := ((x^2 - y^2) + i (2 d |x y| + e x^2)) + c
  ```

- General Abs Quadratic Plus

  ```
  z := ((x^2 + y^2) + i (2 d |x y| + e x^2)) + c
  ```

- `z^2 exp(2 a / z) + c`

  ```
  z := z^2 exp(2 f / z) + c
  ```

A machine-readable version of this formula list is found in the 'et' repository: https://code.mathr.co.uk/et/blob/kf:/kf/formulas.et This is used by 'et' when generating formula code (for Newton-Raphson zooming, etc).

Command Line Usage

```
kf.exe [options]
    -o, --load-map      [FILE.kfb]  load map file
    -c, --load-palette  [FILE.kfp]  load palette file
    -l, --load-location [FILE.kfr]  load location file
    -s, --load-settings [FILE.kfs]  load settings file
```

```
-x, --save-exr      [FILE.exr]  save EXR
-t, --save-tif      [FILE.tif]  save TIFF
-p, --save-png      [FILE.png]  save PNG
-j, --save-jpg      [FILE.jpg]  save JPEG
-m, --save-map      [FILE.kfb]  save KFB
    --save-kfr      [FILE.kfr]  save KFR
-z, --zoom-out      [NFRAMES]   zoom sequence
--log (debug|status|info|warn|error)
                                logging verbosity
-v, -V, --version               show version
-h, -H, -?, --help              show this help
```

Locations and settings can also be image files with embedded comments.

If any of the save options are give, KF switches to a non-interactive mode - it will render the image and save to all specified types before quitting. No GUI.

A typical workflow would be to start KF without arguments, set up the window size (eg 640x360), image size (eg 3840x2160), glitch low tolerance flag, etc, then save the settings to a .kfs file, before quitting.

Then launch KF from the command line telling it to load the settings file you just saved, plus the location file you want to render, and where to save the output images to. Then wait patiently. You can write a script that renders multiple locations in succession, either in batch .BAT on Windows, or in Shell .sh on *nix with WINE.

Note that you might have to double up backslashes within quoted paths (if they contain spaces). Maybe forward slashes work instead, but you do need quotes (either single `''` or double `""`, in a matching pair around the whole argument) if there are spaces. Your shell might also do funky stuff with slashes and quotes, so you might need to double up or quadruple up the backslashes and quotes. Easiest to avoid spaces and keep your files in the current working directory... Example:

```
kf.exe -s settings.kfs -l location.kfr -p out.png -j out.jpg -x out.exr
```

Use `--log info` to disable the status updates, use `--log warn` to output only important messages. The default is `--log status`.

With `-z` you can specify how many frames to render, or `-1` to zoom all the way out. In zoom out mode the output save filenames should contain a printf flag for an integer, for example `image-%08d.png` will have 8 decimal digits padded with leading 0. This is filled by the frame number, which always starts from 0. Zooming is by the zoom size in the settings file.

New in 2.14.6 is standalone KFB map colouring support with the `-o`/`--load-map` flag:

```
kf.exe -o map.kfb -c palette.kfp -p out.png
```

The `-o`/`--load-map` flag can also load raw iteration data from EXR.

New in 2.14.10 is KFR writing, if no image files need to be rendered it is very fast to output a zoom sequence (note: no auto-iterations support in this mode).

Tiled Rendering

Tiled rendering is useful for large images, including images larger than the 2GB pixel data limit of Windows bitmaps. The currently supported way of rendering tiled images is via EXR files, though there are some Octave/Matlab scripts still available that can work with more common image files.

Example: target final size 64000x36000, tile factor 10x10. Configure image size to 6400x3600, adjust other settings, save `input.kfs` and `input.kfr`. Then run (syntax for Bash shell):

```
kf-tile.exe input.kfs input.kfr 10
for tile in input-*.kfs
do
```

```
  kf.exe -s ${tile} -l ${tile%.kfs}.kfr --save-exr ${tile%.kfs}.exr
done
exrtactile.exe input 10 0 output.exr  # for side-by-side tiles
exrtactile.exe input 10 1 output.exr  # for stratified tiles
```

You can append one argument at the end of the `kf-tile.exe` command:

- `--auto`: use stratified tiling if sensible, otherwise adjacent
- `--adjacent`: force using side-by-side tiles (may have seams between tiles)
- `--stratify`: force using stratified tiles (may have artifacts if numerical differencing is used for DE and/or slopes)

The EXR tile assembler is available at https://mathr.co.uk/exrtact including Windows program binary.

Third-party Software

exrtact

https://mathr.co.uk/exrtact

exrtact is a suite of small command-line programs for manipulating EXR files. Of particular note is `exrtactile` for tile assembly (see above).

zoomasm

https://mathr.co.uk/zoomasm

zoomasm is a zoom video assembler for turning exponential map EXR keyframe sequences into movies. It has a timeline system in which you can set zoom depth at specific times, for example to synchronize with a soundtrack.

kf-extras

https://code.mathr.co.uk/kf-extras/blob/HEAD:/README

```
git clone https://code.mathr.co.uk/kf-extras.git
```

kf-extras is a suite of small command-line programs for manipulating KFB files. Programs include colouring algorithms (curvature, histogram, pseudo-de, rainbow, stretch) and other tools (de-histogram, expmap, resize, statistics, to-exr, to-mmit) as well as a bash script to generate a zoom video (which needs the mightymandel zoom assembler below). The programs use a small library to abstract the KFB and PPM reading and writing, which can be used as a basis for your own programs.

After Effects zoom tools

https://fractalforums.org/f/15/t/2965

saka on fractalforums.org is working on a plugin for Adobe After Effects:

> I'm working on a plug-in for Adobe After Effects. It renders from the .kfb file data generated by Kalles Fraktaler. I will release the plug-in as open-source, but of course Adobe products are not free.
>
> It's still very much a work in progress, but I thought I'd share in case anyone is interested in trying it out at this early stage. It might have some bugs.
>
> Why After Effects? It provides a pretty solid animation, layering and rendering engine. All the settings can be animated, and it is easy to build layered compositions using KF data. Plus having 4k and 8k export in various codecs is appealing.
>
> Requirements: After Effects 2019, Windows only.

See fractalforums.org for downloads and tutorials.

Key Frames Movie Maker

http://www.chillheimer.de/kallesfraktaler

The original Movie Maker by Karl Runmo. Has controls for changing speed, rotation, colour cycling, and more, at specific frames. Also has an option for zoom depth text overlay. Works from KFB files and generates multiple video file segments.

Yann Le Bihan's Movie Maker 3D

http://www.chillheimer.de/kallesfraktaler/mmy/

Works from KFB files and generates video segments with a 3D effect:

> This Manual is formed as a tutorial describing the steps to make a movie with 3D effect, which includes all settings.

Dinkydau's Mandel Machine Convertor

Mandel Machine is a fast Mandelbrot renderer for Windows written in Java and assembly.

Dinkydau on fractalforums.org has written:

> something to convert the MMI file format to KFB

so that Mandel Machine glitches can be corrected in KF or KF renders can be coloured in Mandel Machine. But Dinkydau has not released it. Maybe if you ask nicely you will be granted access. More information at:

https://fractalforums.org/f/66/t/769/msg3960#msg3960

SeryZone's Map Visualizer and Fractal Video Maker

Dinkydau on fractalforums.org has written:

> My favorite zoom video maker (it only creates PNG sequences) is SeryZone's kfb map visualiser. It's hard to use because of the very not-intuitive interface though.

http://www.fractalforums.com/announcements-and-news/sft-map-visualizer

> With this tool you can create PNG images from KFB maps and use them as keyframes with the other tool fractal video maker. Use fractal video maker to interpolate extra png files, for example 60 between each keyframe for a 60 fps zoom video at a speed of 1 zoom per second. (where by "zoom" I mean an increase in magnification by a factor 2, which is the recommended zoom size in kalles fraktaler)

https://fractalforums.org/f/15/t/1364/msg7023#msg7023

Later in that thread are download links for the tools.

mightymandel zoom interpolator

mightymandel is a Mandelbrot set renderer using OpenGL.

https://mightymandel.mathr.co.uk/

It has a zoom assembler in its `extra` subfolder.

https://mightymandel.mathr.co.uk/current/usage.html#usagezoom

```
usage: zoom iwidth iheight iframes olength [ofps [oshutter]] \
  < stream.ppm > stream.y4m
```

It is a bit picky about the format of the input PPM stream, and input frames must be 2x zoomed at each frame from zoomed out to zoomed in. Arguments:

iwidth: input frame width

iheight: input frame height

iframes: input frame count

olength: output video length in seconds

ofps: output frame rate (25fps default)

oshutter: motion blur (0 no blur, 1 full blur, 0.5 default)

You might need to downscale the input PPM stream to something your GPU can manage, noting that the internal code needs a texture 2x bigger than the input/output frames. Up to 4096x4096 input should be ok on most modern cards. The output dimensions are the same as the input. For best quality downscale after zoom interpolation, not before, if possible.

et zoom interpolator

et is an escape time fractal project implemented in Haskell.

https://mathr.co.uk/et

It has a forked version of the mightymandel zoom interpolator optimized for distance estimation rendering. Usage is the same as the mightymandel zoom interpolator except that the input is raw headerless floating point data (single channel containing distance estimate, normalized to 1 for the neighbouring pixel).

The difference to the mightymandel zoom interpolator is that it interpolates the DE data before colouring instead of after, which gives a much smoother image (no strobing filaments). Output size is hardcoded to 1920x1080, you can edit this near the top of the `main()` function in `zoom.c` before compiling. Input frame size should be as large as possible for best quality.

The program has two more arguments compared to the mightymandel zoom interpolator: whether to output a header (set to 1 unless you know what you are doing) and an "inverse video" flag that makes it white on black instead of black on white. The output Y4M stream is mono, you might have to tell ffmpeg to encode `-pix_fmt yuv420p` for some codecs.

```
usage: zoom iwidth iheight iframes olength \
  [ofps [oshutter [header [inverse]]]] < stream.raw > stream.y4m
```

zoom-tools zoom-interpolator

https://mathr.co.uk/zoom

zoom-tools is a suite of small command-line programs for turning EXR zoom sequences into zoom videos. It has filters for input keyframes and output video frames, allowing video files to be encoded without needing so much temporary disk space.

zoom-tools has been obsoleted by *zoomasm*.

"book" zoom interpolator

The mightymandel zoom interpolator is better in every way (it is based on the same code, just enhanced). And the book repository is 60MB because it has images. But if you want to study it for historical reference:

```
wget -O Makefile https://code.mathr.co.uk/book/blob_plain/HEAD:/code/Makefile
wget -O zoom.c https://code.mathr.co.uk/book/blob_plain/HEAD:/code/zoom.c
make zoom
```

Or using git (note: the repository is large):

```
git clone https://code.mathr.co.uk/book.git
cd book/code
make zoom
```

Usage:

```
./zoom iwidth iheight iframes olength < stream.ppm > stream.y4m
```

File Formats

EXR Channels and Metadata

Discussion about a common format for escape time fractal data is taking place here:

https://fractalforums.org/f/11/t/3017

Here are the channels and metadata that KF currently supports:

- RGB half float colours
- normalized iteration count

  `uint32 N` integer iteration count

  - `0xFFFFFFFF` is non-escaped before header metadata field `int Iterations` (or `string Iterations`, as it can exceed the range of `int`)
  - 0x00000000 is uncalculated/glitch/no-data-available.
  - If actual iteration values can be zero or negative, add a bias constant to each count and store it in the header metadata field `int IterationsBias` (or `string IterationsBias`, it can exceed the range of `int`). The bias could be negative, this might allow you to store high iteration counts without necessarily needing two channels if the actual min/max range is small enough)

  For images with biased iteration counts above `0xFFFFFFFE`, split into two channels:

  `uint32 N0` least significant 32 bits

  `uint32 N1` most significant 32 bits

  `(0xFFFFFFFF, 0xFFFFFFFF)` is interpreted as non-escaped

  For future supercomputers, this can be extended with `N2` etc...

  `float NF` fractional iteration count, expected to be in `[0.0 .. 1.0)`

  The continuous iteration count (when escaped) is `N+NF-IterationsBias`. This is stored separately to avoid losing precision at high iteration counts

- phase of first escaped Z value, measured in turns

  `float T` in `[0.0 .. 1.0)`

  It is desirable that this aligns with `NF` to give 2D exterior grid cell coordinates, KF versions before 2.15 align only with `Linear` smoothing.

- directional DE (when derivatives have been calculated)

  `float DEX`, `float DEY` directional distance estimate in cartesian form, normalized such that distance to a neighbouring boundary pixel `sqrt(DEX^2 + DEY^2)` is approximately `1.0`.

  If some pixels have no directional DE the missing data can be written as `(0.0, 0.0)`, but readers should also handle `NaN`s in this case. The vector points away from the fractal boundary.