

# Kalles Fraktaler 2 + GMP

As the original 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!

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

Original upstream version:

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

This version:

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

Feedback:

- <https://fractalforums.org/kalles-fraktaler/15> new forum (still in beta)
- <http://www.fractalforums.com/kalles-fraktaler/> legacy forum
- [mailto:claude@mathr.co.uk?subject=Kalles%20Fraktaler%20personal mail](mailto:claude@mathr.co.uk?subject=Kalles%20Fraktaler%20personal%20mail)

## Known Bugs

- out of memory conditions cause crashes (for example, if bitmap creation fails - also need to check huge sizes) (reported by gerrit)
- 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 command line rendering corrupts image
- minimizing window during zoom sequence rendering corrupts image (saves blank image or repeated frame) (reported by gerrit and CFJH)
- translating location while reuse reference is active leads to bad images (reported by Dinkydau)
- crash when zooming too quickly near interior black regions (reported by Foxxie) “usually near the elephant valley area or seahorse valley area of minibrots, happens worse the faster you zoom, usually if you try to zoom at the skinniest part very quickly”
- non-responsive GUI when entering “simple” locations (eg -2.0 0.0, doesn’t depend on zoom level) (reported by Foxxie)
- lines in the rendering (reported by CFJH)
- on special locations kf renders endless references and comes to no end (reported by CFJH)
- “resume zoom sequence” assumes “zoom size” is an integer
- “resume zoom sequence” re-uses last set zoom count limit
- “examine zoom sequence” doesn’t save corrected PNG images during glitch solve
- speckles when rendering zoom out sequence
- black regions when rendering zoom out sequence (maximum iterations are reduced too much before spirals appear in next frame) (reported by gerrit)
- 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 zooming to minibrot doesn’t increase maxiters enough sometimes
- status bar reference count doesn’t reset when zooming before it is “Done”
- help button in file browser does nothing
- opengl support is very broken, proof of concept only
- may be difficult to build the source natively at the moment (out of date instructions for Windows)

## Differences From Upstream 2.11.1

### Incompatible Changes

- 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)

### Other Changes

- Makefile build system using MINGW to cross-compile to Windows from Linux
- uses GMP for arbitrary precision floating point instead of custom code
- uses Boost wrapper around GMP 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
- very experimental and broken OpenCL using CLEW (still disabled at build time)
- save images to PNG 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 (both PNG and JPEG)
- 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 quitting
- see change log below for more...

## Change Log

- **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
- upgrade to Boost 1.66.0
- **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)
  - reenabled 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 quitting
    - 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 gerriit)
- **kf-2.12.3** (2017-09-25)
  - multiple finite difference methods for distance colouring (suggested by gerriit)
  - 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 Mandelbrot is much faster due to lower-level calls to gmp)
  - very experimental openc1 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)
- user interface: show box-region for nr-zoom before clicking (suggested by Foxxie)
- user-interface: 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)

- user interface: undo history for location data (suggested by TwinDragon)
- user interface: undo history for calculation data (suggested by TwinDragon)
- user interface: online help within program (suggested by TwinDragon)
- user interface: drag-and-drop parameters or images on to main window to load
- user interface: save image now function (without waiting for calculations)
- calculations: increase ref count limit without restarting from scratch
- calculations: increase maxiters limit without restarting from scratch
- calculations: optimize series approximation and probe point stuff
- calculations: work on OpenCL some more (try to get it working)
- calculations: calculate series approximation in parallel with reference
- calculations: calculate derivatives for “true” distance estimates
- calculations: refine minibrot using interior distance estimates
- calculations: refine minibrot using boundary shrinking (calculate edges only)
- calculations: Horner’s rule for polynomial evaluation (if not used already)
- newton: add zoom depth offset in addition to zoom depth factor to (eg) zoom closer to minibrot or embedded Julia
- newton: zoom to Misiurewicz points (custom zoom factor, manual preperiod selection) (suggested by gerrit)
- newton: properly debug huge zoom values from size estimate
- libraries: upgrade to libmpfr-4.0.0 <http://www.mpfr.org/mpfr-current/mpfr.html>
- libraries: upgrade to libjpeg6b2 <http://jpegclub.org/support/>
- preprocessor: flatten complex numbers to separate real and imaginary parts
- preprocessor: common subexpression elimination (share results, might be especially useful for large powers of complex numbers)
- preprocessor: automatically parallelize reference iterations
- colouring: high bit depth image export (eg 16bit PNG) (suggested by Dinkydau)
- colouring: assume sRGB display and gamma-correct downscaling
- colouring: load/save palette to/from image
- colouring: rework entirely (now: 1024 colours with mandatory interpolation)
- colouring: implement Pauldelbrot’s multiwave colouring
- colouring: colour cycling (suggested by blob)

## 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-2.12     # for current stable development
git checkout kf-2.13     # for current experimental development
git tag -l               # list available release tags
```

## Building On Linux

Build instructions for cross-compiling from GNU/Linux require about 4.2GB of disk space and good internet download speed (or patience). About 600MB of downloads including the chroot debbootstrap step. To build the PDF manual needs some more packages, adding another 600MB of downloads and 1GB of space, so I left that optional. If you have recent Debian you can skip the chroot step and install natively. See also the `prepare.sh` script which does much of the work, saving you from excessive copy/paste from this document.

0. Setup Debian Stretch chroot:

```
mkdir ./vm
```

```

sudo debootstrap stretch ./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
apt-get update
apt-get install \
    build-essential \
    cabal-install \
    ghc \
    git \
    libghc-parsec3-dev \
    libtool \
    lzip \
    m4 \
    mingw-w64 \
    p7zip \
    wine32 \
    wine64 \
    wine-binfmt \
    xsltproc \
    zip
apt-get install \
    pandoc \
    texlive-latex-recommended # optional, for PDF manual

```

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

2. Prepare non-root build user:

```

adduser build
# enter and confirm password
su - build
export CPPFLAGS=-D__USE_MINGW_ANSI_STDIO
export NCPUS=8
mkdir -p ~/win64/src
mkdir -p ~/win32/src

```

3. Download sources:

Download current versions of the needed libraries, and clone kf git sources:

```

cd ~/win64/src
wget -c https://dl.bintray.com/boostorg/release/1.66.0/source/boost_1_66_0.7z
wget -c https://gmplib.org/download/gmp/gmp-6.1.2.tar.lz
wget -c https://ftp.gnu.org/gnu/mpfr/mpfr-3.1.6.tar.xz
wget -c https://zlib.net/zlib-1.2.11.tar.xz
wget -c http://www.ijg.org/files/jpegsrc.v6b.tar.gz
wget -c ftp://ftp-osl.osuosl.org/pub/libpng/src/libpng16/libpng-1.6.34.tar.xz
git clone https://code.mathr.co.uk/kalles-fraktaler-2.git
cd kalles-fraktaler-2
git checkout kf-2.12
cd ..
cp -avt ~/win32/src *z kalles-fraktaler-2/

```

Internet access is no longer required after this step.



#### 4. Build dependencies

##### 1. Build GMP (64bit and 32bit):

```
cd ~/win64/src
tar xf gmp-6.1.2.tar.lz
cd gmp-6.1.2
./configure --host=x86_64-w64-mingw32 --prefix=$HOME/win64
make -j $NCPUS
make install
make check
```

```
cd ~/win32/src
tar xf gmp-6.1.2.tar.lz
cd gmp-6.1.2
./configure --host=i686-w64-mingw32 --prefix=$HOME/win32
make -j $NCPUS
make install
make check
```

##### 2. Build MPFR (64bit and 32bit):

```
cd ~/win64/src
tar xf mpfr-3.1.6.tar.xz
cd mpfr-3.1.6
patch -N -Z -p1 < ../mpfr-3.1.6.patch
./configure --host=x86_64-w64-mingw32 --prefix=$HOME/win64 \
  --with-gmp-build=../gmp-6.1.2 --enable-static --disable-shared
make -j $NCPUS
make install
make check
```

```
cd ~/win32/src
tar xf mpfr-3.1.6.tar.xz
cd mpfr-3.1.6
patch -N -Z -p1 < ../mpfr-3.1.6.patch
./configure --host=i686-w64-mingw32 --prefix=$HOME/win32 \
  --with-gmp-build=../gmp-6.1.2 --enable-static --disable-shared
make -j $NCPUS
make install
make check
```

##### 3. Build ZLIB (64bit and 32bit):

```
cd ~/win64/src
tar xf zlib-1.2.11.tar.xz
cd zlib-1.2.11
CC=x86_64-w64-mingw32-gcc ./configure --static --prefix=$HOME/win64
CC=x86_64-w64-mingw32-gcc make -j $NCPUS
CC=x86_64-w64-mingw32-gcc make install
```

```
cd ~/win32/src
tar xf zlib-1.2.11.tar.xz
cd zlib-1.2.11
CC=i686-w64-mingw32-gcc ./configure --static --prefix=$HOME/win32
CC=i686-w64-mingw32-gcc make -j $NCPUS
CC=i686-w64-mingw32-gcc make install
```

##### 4. Build PNG (64bit and 32bit):

```
cd ~/win64/src
tar xf libpng-1.6.34.tar.xz
```

```

cd libpng-1.6.34
./configure --disable-shared --host=x86_64-w64-mingw32 \
  CPPFLAGS=-I$HOME/win64/include LDFLAGS=-L$HOME/win64/lib \
  --prefix=$HOME/win64
make -j $NCPUS
make install

```

```

cd ~/win32/src
tar xf libpng-1.6.34.tar.xz
cd libpng-1.6.34
./configure --disable-shared --host=i686-w64-mingw32 \
  CPPFLAGS=-I$HOME/win32/include LDFLAGS=-L$HOME/win32/lib \
  --prefix=$HOME/win32
make -j $NCPUS
make install

```

5. Build JPEG (64bit and 32bit):

```

cd ~/win64/src
tar xf jpegsrc.v6b.tar.gz
cd jpeg-6b
./configure --disable-shared CC=x86_64-w64-mingw32-gcc \
  --prefix=$HOME/win64
make -j $NCPUS
cp -av libjpeg.a ~/win64/lib
cp -av jpeglib.h jconfig.h jmorecfg.h jpegint.h jerror.h ~/win64/include

```

```

cd ~/win32/src
tar xf jpegsrc.v6b.tar.gz
cd jpeg-6b
./configure --disable-shared CC=i686-w64-mingw32-gcc \
  --prefix=$HOME/win32
make -j $NCPUS
cp -av libjpeg.a ~/win32/lib
cp -av jpeglib.h jconfig.h jmorecfg.h jpegint.h jerror.h ~/win32/include

```

5. Prepare Boost headers

```

cd ~/win64/src
7zr x boost*.7z
cd ~/win64/include
ln -s ../src/boost*/boost/

cd ~/win32/include
ln -s ../../win64/src/boost*/boost/

```

6. Finally, build Kalles Fraktaler 2 + GMP (64bit and 32bit):

```

cd ~/win64/src
cd kalles-fraktaler-2
make -j $NCPUS SYSTEM=64
make README.pdf # optional, for PDF manual

cd ~/win32/src
cd kalles-fraktaler-2
make -j $NCPUS SYSTEM=32

```

7. To cut a release bundle, use the script

```

export VERSION=2.whatever
git tag -s kf-${VERSION}
./release.sh ${VERSION}

```

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/c
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 Stretch has:

$ x86_64-w64-mingw32-g++ --version
x86_64-w64-mingw32-g++ (GCC) 6.3.0 20170516
Copyright (C) 2016 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

(note: these instructions are out of date)

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

0. Remove any old msys2.
1. Downloaded latest version of msys2 (msys2-x86\_64-20161025.exe). This is the 64 bit version. msys2-i686-20161025.exe is the 32 bit version.
2. After running it, it installs msys2. At the end the msys2 shell is launched.
3. In the msys2 shell, invoke pacman:  

```
pacman -Syuu
```

This have to be done until is says there is nothing to do anymore.
4. Close the msys2 shell:  

```
exit
```
5. Reopen msys2 shell (from startup menu).
6. Install mingw/gcc 64 bit:  

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

one can also install 32 bit version by:

```
pacman -S mingw-w64-i686-toolchain
```
7. Install Boost  

```
pacman -S mingw-w64-x86_64-boost
```

from msys shell

8. Close msys2 shell then open “msys2 mingw 64 bit” shell (in order to have all the environment variables properly set)
9. Change directory to the kalles fraktaler sources (where **Makefile** resides).
10. Compile

```
mingw32-make WINDRES=windres
```

(if this doesn't work edit the Makefile to replace the line

```
WINDRES ?= x86_64-w64-mingw32-windres
```

to

```
WINDRES ?= windres
```

and run `mingw32-make` without arguments)

11. Execute it this way from (msys2 mingw 64 bit) command line:

```
./fraktal_sft64    # for the claude branch  
./kf.exe          # for the claude-gmp branch
```

because it is linked dynamically to some libraries. In order to execute it from the explorer one needs to copy `libgmp-10.dll` and `libwinpthread-1.dll` from `msys64/mingw64/bin` next to the generated executable.

## Legal

- Copyright (c) 2013-2017 Karl Runmo, (c) 2017-2018 Claude Heiland-Allen
- this software is based in part on the work of the Independent JPEG Group <http://jpegclub.org/reference/libjpeg-license/>
- the PNG library is used under the libpng license <http://libpng.org/pub/png/src/libpng-LICENSE.txt>
- the ZLIB library is used under the zlib license [http://zlib.net/zlib\\_license.html](http://zlib.net/zlib_license.html)
- the GMP library is used under the conditions of the GNU Lesser General Public License version 3 and the GNU General Public License version 2 <https://www.gnu.org/licenses/lgpl-3.0.en.html> <https://www.gnu.org/licenses/gpl-2.0.html>
- the Boost library is used under the Boost Software License Version 1.0 [http://www.boost.org/LICENSE\\_1\\_0.txt](http://www.boost.org/LICENSE_1_0.txt)
- the CLEW library is used under the Boost Software License Version 1.0 [http://www.boost.org/LICENSE\\_1\\_0.txt](http://www.boost.org/LICENSE_1_0.txt)

**NOTE:** the binaries are statically linked with GMP, which is under dual LGPLv3 / GPLv2 license. If you redistribute the binaries you must also be prepared to distribute the source corresponding to those binaries to anyone you distribute the binary to. To make this easier for you, the more recent zips include the source too (though you'll also need to get the library sources). And of course insert here the usual legal disclaimers about **NO WARRANTY OF ANY KIND**.

## 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.

## User Manual

Shortcut only:

- **Ctrl+B**

Toggle skew animation. Enter the number of frames in the popup dialog

Menu items:

### File

- **Open**

Opens the current location from a parameter file (\*.kfr)

- **Save**

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

- **Save as**

Saves the current location in a new parameter file (\*.kfr)

- **Save as PNG**

Saves the current location in a PNG file (\*.png)

- **Save as Jpeg**

Saves the current location in a jpeg file (\*.jpg)

- **Store zoom-out images**

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

- **Save map**

Saves the current location in a map file (\*.kfb). This file can be used by the KeyFramMovie program.

- **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 sequence, if it got interrupted.

- **Exit**

Exit this program

## Action

- **Zoom size**

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

- **Location...**

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

- **Iterations...**

Displays the Iterations dialog where the maximum iteration number for this location is displayed and can be edited.

The smooth color transition method is also set here, and the power on the Mandelbrot function.

The fractal types is also set here - Mandelbrot, Burning Ship, Buffalo or Celtic.

This dialog also displays

- **Min:** The minimum iteration count for a pixel in this location
- **Max:** The maximum iteration count for a pixel in this location
- **Appr:** The number of iterations given by Series approximation
- **Calculations:** The number of calculations performed and also the number of calculations per second is shown if this dialog is displayed while the image is rendered

- **Set colors...**

Displays the Number of colors dialog where the colors can be edited.

- **Reset**

Set the location to the start point

- **Center cursor**

Center the cursor to image's pattern center

- **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.

- **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

- **Refresh**

Render the current location

- **Cancel rendering**

Cancel the current rendering

- **Rotate**

Activate rotation, drag to rotate the image

- **Reset rotation**

Clear any rotation

- **Show Inflection**

Activate or deactivate display of Inflection

- **Skew**

Opens the Skew dialog which allows to “un-skew” locations that are skewed

- **Zoom animation**

Turns animation on or off when zooming

## Special

- **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

- **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.

- **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

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

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

- **Auto solve glitches**

Turns the Auto solve glitches function on or off

- **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

- **Find highest iteration**

Centers the mouse pointer over the pixel with the highest iteration

- **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

- **No approximation**

Turns the Series approximation function on or off.

- **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

- **Special**

- **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 . .

- **Show smooth transition colors**

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

- **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 auto iterations**

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

- **Set Ratio**

Enables changing the ratio between height and width of the background image in order to enable stretching locations. Combined with rotation, an almost infinite skewing ability is enabled, useful when exploring the hidden treasures of the new Fractals!

- **Reset Ratio**

Reset ratio to default

- **Skew animation**

Activates or deactivas skew animation. If activated, a popup allows you to specify end skew parameters and number of frames. The fractal will be rendered frame by frame, and can be combined with frame by frame rendering in KeyFrameMovieMaker or MMY3D

- **Show glitches**

When activated, glitches are displayed with a solid color

- **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.

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

Notice that it can take an hour or more to calculate the position of minibrots beyond e1000. 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

## 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.

## Number of colors dialog

- **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



- **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 file
- **Open palette**  
Load palette from file
- **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.

## Command Line Usage

```
kf.exe [options]
  -l, --load-location [FILE.kfr]  load location file
  -s, --load-settings [FILE.kfs]  load settings file
  -p, --save-png         [FILE.png]  save PNG
  -j, --save-jpg         [FILE.jpg]  save JPEG
  -m, --save-map         [FILE.kfb]  save KFB
  -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. The GUI is updated less frequently (only after each reference, instead of twice a second) which should improve rendering times, particularly for large images.

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 -m out.kfb
```