Kalles Fraktaler 2 + GMP

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!

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%202 personal mail

Known Bugs

- "no newton.kfr" blank image on load and newton-raphson zoom fails with bad period detected (reported by Kalles Fraktaler)
- 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
- examine zoom sequence auto solve glitches is much less effective at solving glitches than when rendering with auto solve glitches: possible precision loss / misalignment between pixels and true parameter plane locations? for now it is recommended to avoid the "reuse reference" setting, and keep "auto solve glitches" enabled, when rendering zoom out sequences (reported by Fractal universe)
- newton-raphson zooming to minibrot doesn't increase maxiters enough sometimes
- PNG save option opens a dialog called "JPEG properties" with an unused "quality" option (you can still set image size, but potentially confusing)
- occasional randomly bright single pixels: possible race condition in rendering completion testing meaning some pixels are not rendered?
- opencl 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- may be difficult to build the source natively at the moment (out of date instructions for Windows)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
- used JPEG library downloaded as necessary at build time, instead of bundled
- 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)

Change Log

- **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

- building: build against an installed libjpeg instead of each time after clean
- user interface: batch mode
- user interface: scripting interface
- calculations: implement scaled long double for e4900 to e9800
- calculations: optimize series approximation and probe point stuff
- calculations: work on OpenCL some more (try to get it working)
- preprocessor: flatten complex numbers to separate real and imaginary parts
- 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

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 claude-gmp   # for the GMP fork
git checkout formulas     # for current 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 debootstrap 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.

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
```

2. Prepare non-root build user:

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

3. Download sources:

   Download the latest Boost (which is at time of writing is 1.65.1) and latest GMP (currently version 6.1.2) and clone kf git sources:

```
cd ~/win64/src
wget https://dl.bintray.com/boostorg/release/1.65.1/source/boost_1_65_1.7z
wget https://gmplib.org/download/gmp/gmp-6.1.2.tar.lz
wget https://zlib.net/zlib-1.2.11.tar.xz
git clone https://code.mathr.co.uk/kalles-fraktaler-2.git
cd kalles-fraktaler-2
git checkout formulas
make jpegsrc.v6b.tar.gz
cd ..
cp -avit ~/win32/src boost*.7z gmp*.lz zlib*.xz kalles-fraktaler-2/
```

   You also need to get libpng (version 1.6.32) from a non-automatable link at http://www.libpng.org/pub/png/libpng.html (save it to `.../vm/home/build/win64/src` and copy it to `.../vm/home/build/win32/src` too).

   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 8
```

```
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 8
make install
make check
```

2. 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 8
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 8
CC=i686-w64-mingw32-gcc make install
```

3. Build PNG (64bit and 32bit):

```
cd ~/win64/src
tar xf libpng-1.6.32.tar.xz
cd libpng-1.6.32
./configure --disable-shared --host=x86_64-w64-mingw32 \
  CPPFLAGS=-I$HOME/win64/include LDFLAGS=-L$HOME/win64/lib \
  --prefix=$HOME/win64
make -j 8
make install

cd ~/win32/src
tar xf libpng-1.6.32.tar.xz
cd libpng-1.6.32
./configure --disable-shared --host=i686-w64-mingw32 \
  CPPFLAGS=-I$HOME/win32/include LDFLAGS=-L$HOME/win32/lib \
  --prefix=$HOME/win32
make -j 8
make install
```

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 8 SYSTEM=64
```

```
make README.pdf   # optional, for PDF manual

cd ~/win32/src
cd kalles-fraktaler-2
make -j 8 SYSTEM=32
```

7. To cut a release bundle, use the script

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

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

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 and others for reporting bugs
- Chillheimer for hosting my program

Claude also thanks Karl for releasing the source to this program so that we all could learn from it and make modifications.

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 sequnce, 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. Combinated 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

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.