XaoS 3.2

A fast real-time interactive fractal zoomer — User's manual

Jan Hubička
Dukelských bojovníku 1944
390 03 Tábor
Czech Republic
Email: jh@ucw.cz

Jan 23, 2006

1 Overview

1.1 Why yet another fractal generator?

We decided to make XaoS, because all fractal browsers we know of take a long time to calculate each image. You may browse nice images generated by them but real impressions of fractals — the self similarity and infinite zooming into the nice parts of fractals — can be seen only in animations. There are many programs available that make nice animations, but they take a long time to calculate and lots of space on disk. Most such animations are quite ugly because their authors can't see them without many hours of calculations.

A natural question is: is it possible to generate such animations in real-time? The answer was negative for many years, since the Mandelbrot set is very computationally expensive. Things are changing. Today's computers are fast enough to calculate approx. 10.000 of pixels per frame, which is enough for a very low resolution animation (100x100). Several programs doing that are available. But 100x100 animation still looks quite ugly. To make animation nice you need at least 320x200 pixels. And that is 6 times more! One possibility is to wait until computers will be fast enough, but it will take many years, and then 320x200 animations will be obsolete and everyone will want 1024x768 resolution instead or more.

We found a special algorithm that saves up to 99.98% of calculations during zooming by reusing pixels already calculated in previous frames. There were some programs doing similiar tricks before but we don't know about any able to do zooming interactively with a speed similar to XaoS. Many other tricks were later implemented XaoS to achieve yet higher framerates. Now XaoS does up to 120 frames per second on a 120Mhz pentium in a fullscreen 320x200 animation, and calculates an average of 160 (0.24%) pixels per frame. This makes XaoS fast enough to achieve its primary goal, realtime animation, but there are still many areas that could improve, since more complex fractals, higher resolutions, or slower computers still bring many problems.

1.2 What does this software do then?

XaoS is a realtime interactive fractal zoomer. This means that it lets you zoom smoothly into any place in the fractal you choose without the many hours of calculation required by most other fractal generators. It now has many other features too, like 13 different fractal types, autopilot, special coloring modes, support for various bit depths (8bpp, truecolor, hicolor and realcolor), random palette generation, color cycling etc...

2 XaoS tutorial

This is a brief introduction to the basic XaoS features.

2.1 How to zoom

The main advantage of XaoS is that after a few seconds' delay to calculate the first image, you may choose any point with the mouse and press the *left* button. The image will start to zoom smoothly into the point you choose. You may move the mouse and zoom smoothly into interesting areas. By pressing the *middle button* (or *left*+*right* buttons) you may also *move the image* using "drag & drop" if you missed an interesting place. *Unzooming* is also possible by using the *right button*, but it is much slower because optimizations are not so effective as for zooming.

In case you think that the default *speed* is boring (it is quite slow, to make XaoS smooth on a slow computer) you may change it by pressing *arrow up/down*. But faster zooming is more expensive, so if the speed is too high you will see little but funny colorful blinking rectangles.

2.2 Autopilot

To make XaoS yet more impressive we made a special autopilot that automatically drives into interesting boundaries of the set. So you can press `A`, play your favorite music, drink coffee and relax. I never tried this but it should be really relaxing! Many pictures in the XaoS gallery were discovered using the autopilot.

The autopilot also has some additional features. It turns back when the zoomed picture stops being interesting, and is able to spot when it's zoomed into a really boring part (or has reached the limit of floating point numbers) and restart zooming from the top.

2.3 Various fractal formulae

XaoS also supports formulae other than the Mandelbrot set. You may change *formula* using the *number keys* or *SHIFT*+*letters*.

On keys `1` to `5` are *Mandelbrot sets of various power*. The "normal" Mandelbrot set is on key `1`.

On key `6` is a fractal called *Newton*. It is Newton's famous formula for finding roots.

On key `7` is the *fourth ordered Newton* fractal.

On key `8` is a fractal called *Barnsley*.

On key `9` is *Barnsley's second* fractal.

On key `0` is *Barnsley's third* fractal.

With keys `SHIFT-A` you can display a fractal called *octo*. It is a fractal that Thomas discovered in fractint.

With keys `SHIFT-B` you can display a fractal called *Phoenix*. It is a very nice and quite famous fractal.

With keys `SHIFT-C` you can display a fractal called *Magnet*. This fractal has quite a complex formula so it is a bit slow.

With keys `SHIFT-D` you can display the *Magnet2* fractal.

The rest of the built-in fractals are accessible through an other menu, but you can still use the hotkeys.

On `SHIFT-E` is a fractal called *Triceratops* found by Arpad.

On `SHIFT-F` is a fractal called *Catseye* found by Arpad. This is more interesting if you change the bailout value.

On `SHIFT-G` is a fractal called *Mandelbar*. It was in Gnofract4d, and they found it at: http://mathworld.wolfram.com/MandelbarSet.html

On `SHIFT-H` is the *Lambda* fractal.

On `SHIFT-I` and `SHIFT-J` are the *Manowar* and *Spider* fractals, they were found by users of fractint. (Scott Taylor or Lee Skinner) It was on http://spanky.triumf.ca/www/fractint/taylor_skinner_type.html

The next 3 fractals are famous classic fractals.

On `SHIFT-K` is the *Sierpinski* Gasket. You can change its shape by selecting another Julia seed. (This is for technical reasons.)

On `SHIFT-L` is the *Sierpinski Carpet.* It's shape can also be changed by selecting another Julia seed.

On `SHIFT-M` is the *Koch Snowflake.*

2.4 Out-coloring modes

To make fractals yet more interesting, more coloring modes for points outside the set are provided. "Classical coloring mode" uses the number of iterations that the orbit required to escape to (nearly) infinity. You can change this mode from the *Fractal menu* or by pressing key `C` To see more about coloring modes, try the tutorial on Incoloring modes from the XaoS features overview.

Those cryptic names for coloring modes are mathematical formulae, where *iter* means number of iterations, *real* means real coordinate of last orbit, and *imag* means imaginary coordinate of last orbit.

2.5 In-coloring mode

In-coloring mode is similar to out-coloring, except that it changes how things inside the set are displayed. This can also be changed from the *fractal menu* or by pressing `F`.

You might also want to see the tutorial on Out-coloring modes from the XaoS features overview.

2.6 Planes

All fractals displayed by XaoS are functions with a complex parameter. It can be displayed in the normal complex plane, where x is the real part of the number, and y is the imaginary part; but it can also be displayed in a number of other planes. You can select the plane to use from the *Fractal menu*, or by pressing `I`.

Like the coloring modes, planes have cryptic names. You guessed it, they're mathematical formulae. Here `mu` means coordinates in the normal complex plane. If you have coordinates in `1/mu` plane, and you need coordinates in the a complex plane (to calculate the Mandelbrot set) you simply use the coordinates as mu. Lambda is another plane that can be converted to mu using a similar formula.

mu normal mode.

1/mu Inversion: infinity goes to 0 and 0 goes to infinity.

1/(mu+0.25)
Similar to inversion, but moves the center outside of the Mandelbrot set so that it looks parabolic.

lambda Lambda plane.

1/lambda Inversion of lambda plane.

1/lambda-1
: Inversion with moved center.

1/(mu-1.40115)
: A very interesting mode for the Mandelbrot set. It makes small things big, so you can browse the set's details easily.

2.7 Mandelbrot/Julia switching

Most of the fractals displayed by XaoS (currently all of them) have two forms: Mandelbrot and Julia. Every point in a Mandelbrot set has its own Julia set. To see more about this correspondence, try the tutorial on Julia set from the Introduction to fractals.

In the Mandelbrot mode, you can get a corresponding Julia by moving the mouse to an interesting point and pressing `M`. To get back press `M` again. Some fractals (Barnsley and phoenix) are already in their Julia versions, because the Mandelbrot ones are boring. But by pressing `M` in such fractal you should get the Mandelbrot version, and by choosing another point as the base point and pressing `M` again you should get a completely different fractal. The most interesting points for Julia sets are at the boundaries of the Mandelbrot set. Most of the Julias inside or outside the set are boring.

2.8 Fast Julia preview mode

Fast Julia mode is a quick way to find a point to use as a base for the Julia set.. Just press `J` and a small Julia set will be displayed in the top left corner. Then move the mouse around with button 1 depressed, and the Julia for the point the mouse is over will be automatically generated.

2.9 Palette

If you think that the default XaoS colors are ugly or you are just bored by them you can change it by pressing `P`. XaoS will automatically generate random palettes. Many of them look ugly, so press `P` again to get another one until you find one you like.

2.10 Filters

Many interesting effects are done by post-calculation filters. See Section A.12.1 [filter], page 31. XaoS has filters that do everything from embossing, through motion-blurring, right through to turning the fractal into a stereogram. To enable them use the `filter menu` or press `E`.

2.11 Palette cycling

This is a very old trick that makes the Mandelbrot set a little flashier. You may enable or disable it using `Y`. In the truecolor modes you need to enable the palette emulator filter first. This is done via the `E` key, or from the filter menu.

2.12 Changing number of iterations

To calculate fractals perfectly, you need an infinite number of iterations. XaoS does just the first few of them, so after lots of zooming you may get into a place that looks quite boring, and the boundaries of the set are rounded, without any interesting details. This can be changed by changing the number of iterations:

Press and hold `arrow right` and wait until iterations are high enough. This may slow down calculation much. To reduce number of iterations press `arrow left`.

2.13 Changing resolution

XaoS usually starts in a low resolution (320x200 or thereabouts) to make calculations faster. If you have a fast computer or you need to save bigger `.gif` images, you may change the resolution. This can be done by pressing `=` in the full screen drivers, or simply by resizing the XaoS window.

2.14 Changing driver

XaoS usually has more than one driver available. You may change it on the fly in case you want a different one. For example, XaoS started in X11 can be switched at runtime to use the AA driver. This can be done from the UI menu.

This action is bit dangerous, because XaoS can crash during initialization if there is some problem with initialization; XaoS tries to initialize a new driver, and if it fails it attempts to return back to the original. Sometimes this is impossible, and all XaoS can do is terminate..

2.15 Other features

XaoS has many other features, but they don't fit into this tutorial. Most of them are available from the menu, so you can experiment with them. You might also want to see the *animated tutorials* from the *help menu*, to have an idea what XaoS can do.

3 Basic controls

By default the mouse buttons work in the following way:

left zoom in

right zoom out

middle move fractal in a drag-and-drop fashion

This behavior can change. If you enable rotation, the first button is used for rotating fractals. Also, in fast Julia mode, the first button is used to change the seed.

If you don't have a middle button, press both buttons to enable emulation.

After few minutes of zooming you will probably exceed the precision and the fractals will get boring. If you are getting strange big rectangles on the screen, you probably reached the numeric limit: there is no way to avoid this except un-zoom back and choose a different area. It doesn't hurt so much, since you have zoomed approximately 64 051 194 700 380 384 times, so there are quite a lot of areas to explore. Algorithms with unlimited precision exist, but they are still too slow for real-time zooming.

The other possibility is that you have reached the iteration limit. The fractal is calculated approximately, and in this case you need to increase number of iterations used for approximation (and decrease the speed in the process). This is done from the menu or using the arrow keys *Left* and *Right*.

An *Up* and *Down* keys should be used to change zooming speed. Note that higher speed costs more and image will be blocky.

This behavior can also change. With palette cycling enabled, *Left* and *Right* keys change cycling speed; in continuous rotation they change rotation speed.

All other functions are available from the menu, which (in the default configuration) are displayed when you move the mouse to the top of the screen/window. It is useful to learn the shortcut keys, which are shown in gray next to the menu items they trigger.

4 How to encode MPEG files from XaoS

To save a sequence, make a `xaf` file first (the easiest way to do this is to use the *record* function in the file menu). Then you need to render the sequence. XaoS can output sequences of ordinary `PNG` images, that can later be used by an MPEG encoder.

4.1 Generating sequences for MPEG

To encode a sequence, use the following command:

```
xaos -render [filename] -size 352x240 -antialiasing
-renderframerate 24 -basename [basename]
```

`[filename]` is the name of the `xaf` file, `[basename]` is the name used as the base for rendered images. XaoS adds a four digit number and extension automatically.

You might also want to change the resolution. 352x240 is the default size for MPEG files, but other sizes work as well. Each dimension must be a multiple of 16.

The framerate can also be altered. MPEG supports only a few selected framerates (namely 23.976, 24, 25, 29.97, 30) and you can pick any of them.

`-antialiasing` is used to produce anti-aliased images. It takes a much longer time and much more memory to calculate them, but resulting images are better for MPEG DCT compression and they are *compressed about 3 times more*. (the same is true of *JPEG images*)

At the other hand, the other XaoS rendering option `-alwaysrecalc` (which disables XaoS's zooming optimizations) is *not recommended*. If that's used, the sequence of animation then contains quite a lot of extra information, which increases size of MPEG file, but because of MPEG's lossy compression it is hard to see any difference. So it don't worth it.

4.2 Rendered files

Once you start it, XaoS will generate thousands of frames. They take quite a long time to calculate and save, and consume plenty of disk space. (e.g. to render part 1 of the tutorial you need about 60MB and half an hour of time).

All images are named `[basename]framenum.png`. For example `intro0001.png` is the first frame of the animation intro. If consecutive frames are the same, XaoS doesn't save them, so some frames may be missing. If your encoder can't handle that, you will need to write a simple script which will fill in the gaps by means of `mv` or symbolic linking.

A list of all filenames is saved into the file `[basename].par`, where each line is the name of one frame. The names repeat here if necessary, so you can use this file to supply filenames to the encoder.

4.3 Pattern file

Some other files are generated as well. A *pattern file* is generated, which contains a *recommended order of P, B and I frames*.

MPEG sequence consist of these three frames. The *I frames* are just images saved in a format similar to JPEG files.

The *P frames* are images which are constructed as a delta from the previous one (the latest I or P frame). In case consecutive frames are similar (and in animations they often are), a P frame takes much less disk space than an I frame.

The *B frames* are constructed from the nearest previous P or I frame and the next P or I frame. They take even less disk space, but they are quite hard to encode. Also they are *not used as previous frames*, so their information is lost once they are displayed. They are usually

rendered at lower quality than a P or I frame and used just to interpolate nearest frame and make animation smoother. It is generally not a good idea to make a whole sequence just from B frames.

Using just P frames is generally not a good idea. It makes the file shorter, but to jump into Nth frame of animation you need to *recalculate all P and B frames since latest I frame.* Decoders often want to jump to some frame (when the user asks, or when they can't decode a sequence in time and must skip some), so you need to have some I frames in the animation to make this possible. The latter reason means that you need to place them quite often. Usually they are used for every 15th frame or thereabouts. Because they cost quite a lot, in my animations I usually use every 27th frame. To set this distance use `-iframedist` option. It should be a multiple of 3.

XaoS generates a recommended order of frames based on its knowledge of fractal motion. Situations where the screen doesn't move at all are rendered just from P frames (since jumping is usually not required here); in situations where the screen changes completely (at least in XaoS's opinion) I frames are used and in other cases, a standard sequence `IBBPBBPBBPBBP...` is used.

If your encoder supports this, you should supply this pattern for encoding to squeeze out some bytes.

4.4 Motion vector files

XaoS also generates a motion vector recommendation for the encoder. This is useful for encoding of B and P frames.

If some objects on the screen are moving at a constant speed, motion vectors can store that speed, so no image needs to be saved to represent that change.

Calculating this motion vector is a very significant task. If you guess them well, you increase quality and reduce file size, which is always great. Calculation also takes lots of CPU and it is hard to get optimal vectors (it just takes too long).

XaoS knows how the fractals move, so it can calculate this vectors quite easily. XaoS saves this information into `*.p` and `*.b` files. (`*.p` are for P frames, `*.b` are for B frames). If your encoder supports this, you should provide this vector to increase quality. They are `not exact` (XaoS can make mistakes); the encoder should try to find its own vectors, then try XaoS's ones, and pick whichever is better.

This technique saves quite a lot of bytes in fast zooming/un-zooming animations (where images move more than 3 or 5 pixels per frame—since most programs look 10-20 pixels around each point for motion vectors).

To enable saving of motion vector files, add the option `-rendervectors`.

4.5 Berkeley parallel MPEG encoder

This is the encoder I use. It seems to be the best freely available software encoder I've tested. It can generate quite small files, but it is rather slow. It is available at Berkeley's FTP site `mm-ftp.CS.Berkeley.EDU` and called `mpeg_encode1.5b`

It has lots of options to tune, so you should spend quite a lot of time playing with this. The configuration I use is in file `doc/mpeg.param`.

I've also made some patches that makes possible to use the pattern and motion files generated from XaoS. The patch is in `doc/mpeg_encode.patch`. So if you want to use these features (they are *EXPERIMENTAL*) you might apply this patch and recompile the encoder.

Once you filled the `mpeg.param` file (see comments inside), you should render sequence using `mpeg_encode [filename]` and with luck you are done.

5 XaoS' file format

This section describes the format used by XaoS for animations, configuration files and saved positions. All these files have a common format, designed to be easily readable, to allow manual editing of files and easy conversion by other programs.

I also taken care to make it easily extensible for future versions of XaoS so I hope there will not be many incompatibilities between various XaoS versions.

The format is a simple set of commands executed sequentially. XaoS does not provide any variables/cycles as usual scripting languages do, but future extension to full-blown Scheme should be easy since the format uses Scheme-like syntax. The syntax of every command is:

`(command_name` *[param1] [param2]*`)`

where parameters are optional and separated by whitespace (an arbitrary number of spaces, tabs and newlines). The parameters can have the following types:

integer number w/o decimal point (`123`)

float floating point number in decimal notation with optional exponent (`1.23E2`)

keyword text started by quote `'`. It is used to pass various string constants like formula name (`'mandel`) Quote is required for scheme compatibility

string Text inside double quotes. The only parameter that should contain whitespace

boolean `#t` for true or `#f` for false

There is a complete description of all XaoS functions (with some examples) and an index of functions in the XaoS registry. See Appendix A [menus], page 18. You may particularly want to read about the animation functions. See Section A.4 [animf], page 19. Also, the following functions are significant:

load

This function loads and interprets a file. It works similarly to `#include` in C.

initstate

Available in version 3.0 and above, this function resets XaoS' state to default values. This command should be at the beginning of each animation file, since otherwise some stuff previously enabled by user could cause unexpected effects. State is not reset by default before playing animations since it would make it impossible to write macros. Current versions don't really need macros, but in future versions, when the Scheme programming language will be available, this should be a much more interesting subject.

usleep

This function waits for a selected amount of time(in usec) before processing the next command. The screen is recalculated and displayed at the beginning of the sleep if necessary. The remaining time is spent by waiting, calculating if necessary, or performing any animation you entered via animation commands.

wait

Waits until the animation or image rendering is complete. Do not call this function when zoom, or continuous rotation is active otherwise deadlock happens. It is a good idea to call it immediately before text subtitles are displayed, since it looks ugly when they are displayed over a blocky unfinished fractal. Because the degree of blockiness at a given instant is a function of your machine speed, it may look nice for you but ugly for others with slower machines. Also you should call this

after an animation is performed, before the switch to another fractal happens; since the switch involves calculation, the screen is stopped for a while and an unfinished fractal there looks ugly. You should also call it, when you want to do something as soon as possible.

Example:

```
;configure everything for the first frame
(inistate)
(palette 1 1163254293 0) ;custom palette
(cycling #t) ;enable cycling
(cyclingspeed 7)
(maxiter 276) ;higher number of iterations
(range 3) ;default range for solid guessing
(usleep 1000000) ;second frame starts here
(moveview -1.8101154154614007889 -8.2687205907162041209E-05)
;just move the image
(usleep 1000000) ;third frame
(morphview -1.8101154154614007889 -8.2687205907162041209E-05
6.27721097106945236 1E-10 6.27721097853346698 75E-10)
;10 seconds of zooming into selected
rectangle
(usleep 100000000)
```

The best way to learn XaoS command language is probably to read position files and modify them. For example, to create zooming animation from the original file:

```
(initstate)
(defaultpalette 0)
(formula 'mandel)
(view -1.64128273713 -5.50393226816E-05 9.69332308848E-08
9.69332308834E-08)
```

Just change the `view` command to `morphview`, and add `usleep`:

```
(initstate)
(defaultpalette 0)
(formula 'mandel)
(morphview -1.64128273713 -5.50393226816E-05 9.69332308848E-08
9.69332308834E-08)
(usleep 10000000)
```

The following code produces Julia morphing in the real axis:

```
(initstate)
(fastjulia #t)
(juliaseed -2 0)
(morphjulia 2 0)
(usleep 2000000)
```

And following is the "rotozooming" animation:

```
(initstate)
(fastrotate #t)
(morphview -1.64128273713 -5.50393226816E-05 9.69332308848E-08
9.69332308834E-08)
(morphangle 300)
(usleep 10000000)
(wait)
(fastrotate #f)
```

6 XaoS gallery

I plan to make a gallery of animations and position files on the XaoS home-page, so please send any nice animations and images you created using XaoS to `jh@ucw.cz`.

7 How to write XaoS help files

XaoS help is stored in the file `help/xaos.hlp`. It is divided into parts, each part being started by a *keyword*. In the help file keywords are written as `%keyword`

If you are writing documentation about some command in the XaoS function registry, use the same keyword as the name of the command in order to make context sensitive help work.

7.1 xshl

Xshl stands for *XaoS simple hypertext language*. It uses similar tags to HTML. It is simpler and more restrictive in order to make it easy to parse using various scripts. In C code you can use the library present in `src/util/xshl.c` to parse it.

The following tags are supported:

head make headings (should be at the beginning of the page, at least)

emph emphasize

tt Use non proportional font

br Break line

p Next paragraph

dl Definition list

dt Definition tag (should be used only inside a definition list)

dd Definition description (should be used only inside a definition list)

center align to center

right align to right

red change color to red (should not be used in help files)

black change color to black (should not be used in help files)

white change color to white (should not be used in help files)

a name link to other help page

tutor name
activate tutorial

notex Ignore this in texinfo manuals

8 Driver specific documentation

XaoS is portable and works on many different platforms. Since not all platforms are exactly the same, there are some differences between the behaviour of XaoS on different platforms. Here is documentation about each specific port.

8.1 AA-lib driver

The AA driver is currently the most advanced and portable driver for XaoS. It is based on AAlib—a high quality ASCII-art library developed by the AA-project. (see `http://aa-project.sf.net`)

It is a fully featured XaoS driver for text mode displays. It supports 256 colors and the mouse where possible.

It also has some extended features available from the UI menu:

Attributes AA-lib may use character attributes to improve image quality. By default it uses normal, dim and bold characters where possible, but you can also enable different attributes like reversed or bold font characters. You may also enable usage of non ansii/reversed characters if your device supports it.

Font AA-lib uses a bitmap image of the font to prepare the approximation table used for ASCII art rendering. This bitmap is expected to be same as the one used by your device. AAlib performs detection where possible however some devices (like UNIX text terminals or MDA) do not support this. AAlib has few font images compiled in, so in this case you should try to use one of them to achieve best results.

Inversion Some devices use inverse video: use this to get correct results on such devices.

Dithering mode
Dithering is an way to get more exact color in approximations, by combining more characters; but this method can produce ugly looking noise on certain images. Use this menu to disable or tune it.

Palette options
By default AA driver uses the XaoS palette to render images, but it quite often looks ugly on text displays. Here you can choose a special text palette instead. Note that with filters enabled, the results may be rather ugly. This function is available from the *palette menu*.

Save text screen
The normal save function will generate a PNG image instead of nice ASCII-art. To save ASCII art use this function instead. It supports many text file formats like HTML, ANSI, more, etc... It will also ask you for font and attributes(see above). It is available from the *file menu*.

The AA-lib driver also provides the full set of standard AA-lib's command line options. You may use them to tune parameters like gamma correction, and so on. See `xaos -help` or the AA-lib documentation for details.

The AA driver was written by Jan Hubicka, 1997.

8.2 BeOS support

XaoS has pretty advanced support for BeOS R4. It works on both PowerPC and Intel platforms, supports multithreading, the clipboard, file dragging, has native look and feel and can work as an image translator from XaoS files to images.

The first version of the BeOS driver was written by Jens Kilian and later extended by Jan Hubicka.

8.2.1 Installation

You can start the installation script to do everything for you. If you want something special, read this section.

In order for XaoS to work you need to keep the executable together with its data files (`help`, `examples`, `catalogs` and the `tutorials` directory)

When first started, XaoS registers two new mime types called `image/x-xaos-position` for XaoS Position Files and `video/x-xaos-animation` for XaoS Animation Files, registers icons for them and sets itself as default application.

8.2.2 Available display drivers

XaoS supports following drivers:

BeOS Standard windowed driver using application server

DirectWindow
Driver done using Game Kit's direct window class

WindowScreen
Fullscreen driver.

By default, XaoS starts in windowed mode and uses the application server for output. You could change the driver to DirectWindow to use direct access to video RAM. Note that this mode is slower in most cases, and not supported by some videocards.

The BeOS driver by default chooses the most similar bitmap supported by XaoS to achieve best and fastest results. In the UI menu you can change this default choice to another one if you wish. Also you can ask the BeOS and DirectWindow to resize to fullscreen mode.

XaoS also supports real fullscreen mode using the BWindowScreen API. To switch XaoS to this driver, use the UI menu. If you want to use this mode by default, use the `-driver WindowScreen` command line option.

This driver differs a lot from windowed ones. It use direct access to the video card, allowing you to change video mode. Also, the 256 color mode can access the palette, so it is not dithered like the windowed mode. Because BeOS can't do GUI in fullscreen mode, XaoS uses its own toolkit. I hope you will feel confortable in it.

8.2.3 XaoS as translator

You should be able to open XaoS files in graphics applications such as ShowImage or ArtPaing. In Preferences you can find the DataTranslations program, that can be used to set the size, type and DPI of the resulting image. Also antialiasing can be enabled.

Note that *translation can take a while*. So be patient and wait for the result.

If the translator doesn't work, ensure that you have a link to the XaoS executable in `/boot/beos/system/add-ons/Translators/`.

8.3 DGA driver

This is the driver for DGA (Direct Graphics Architecture) extension used by XFree86 X servers. It is pretty new so it could be buggy.

Bugs/limitations:

In 8bpp mode, XaoS has problems with the palette with certain window managers
I don't know why this happens. Just let me know what's wrong, or use another window manager.

Banked modes are not supported.
I don't have any card to test this with, so it doesn't work in the current version.

DGA driver was written by Jan Hubicka, 1999.

8.4 DOS driver

This is a fully featured driver for DJGPP and allegro. It supports many VGA modes, VESA 1.0—3.0, VBE/AF, S3 and some other cards.

The following problems may occur:

Some DPMI servers may cause problems
: Some DPMI servers like the one from Novell/Dr/Open DOS are buggy. Use clean DOS instead and XaoS will automatically start `cwsdpmi`. Under Open Dr DOS use `dpmi off` at command line to disable it.

Higher resolutions don't work
: If your videocard has enough memory for the selected resolution, you most probably have an unsupported videocard. Please use a VESA BIOS extension on this videocard. (See the note about VESA at the end of this section.)

XaoS needs a coprocessor
: I don't distribute a coprocessor library linked into XaoS because it is too slow for a real-time zoomer. Coprocessor emulation will not help, because xaos works in protected mode.

XaoS needs mouse driver to be usable
XaoS works slowly in higher resolution
: This could also be caused by Allegro's slow driver or your videocard's VESA BIOS. You could try some other VESA BIOS extension instead. Look at the `http://www.talula.demon.co.uk` for the FreeBE project or Scitech Display Doctor package. (See the note about VESA at the end of this section.)

8.4.1 VESA

VESA is a standard for using higher resolutions in DOS. Many videocards have VESA support in the BIOS so you don't need any additional software, while others need support from a special program. Also some VESA BIOS implementations are buggy or suboptimal; there are 3 different versions, version 1.0 is many times slower than 2.0, which has support for protected mode and linear framebuffers. So if you have problems with higher resolutions, or some graphics modes are not available (like 320x200 truecolor), you might try some software package which emulates VESA.

The most famous VESA emulating program is Scitech Display Doctor. It has support for many videocards and is quite reliable. It's disadvantage is that it is shareware and works for only 30 days. You might also look on `ftp.simtel.net`, where there are many VESA emulation packages such as `s3vbe` or the new FreeBe project at `http://www.talula.demon.co.uk`

DOS driver was written by Jan Hubicka, 1997.

8.5 DirectX fullscreen driver

This is da river for Windows 9x and NT. It is new since version 3.1 and because of some oddities of Windows API and kludges in DirectX it may be rather unstable. Please report all bugs. In case of problems you could use the DOS version of XaoS instead.

This driver allows the Windows port of XaoS to run in full screen mode. The driver supports 256, 65536 and 16777216 color modes (24bpp and 32bpp) in all resolutions supported by DirectX. You can change graphics mode by pressing the `=` key (or by using the UI/Resize menu). If the selected mode is not supported, the driver will restore the previous setting.

Use the `-mode WIDTHxHEIGHTxDEPTH` (like `-mode 640x480x16`) command line option to change graphics mode.

If you want to start XaoS in DirectX, use the `-driver dX-fullscreen` option.

See the Win32 driver documentation for some more Windows releated information.

DirectX driver was written by Jan Hubicka, Jan Olderdissen and Pavel Tzekov, 1999.

8.6 GGI driver

GGI stands for General Graphics Interface. Part of this project is to develop libggi, a portable graphics library, and XaoS's GGI driver uses that. It is experimental, since the API of libggi is not stabilized yet. There are some problems with keyboard handling—the shift key doesn't work yet.

Everything else might work well, but there are no guarantees. It is alpha quality software.

GGI driver was written by Jan Hubicka, 1998.

8.7 Plan9 driver

Plan9 is a very nice small operating system by the authors of Unix at Bell Labs. It is very incompatible with other operating systems; even the C compiler and header files are different, but XaoS should work well there (even on the limited free demo installation without any POSIX compatibility stuff)

There are a few limitations: the file selector and image saving don't work. You can save position files and then later render them on the other OS, or save screenshots.

Plan9 terminals also don't provide any way to catch the arrow keys, so you can't use them. Use the mouse to navigate in the menus. Also, getting the screen resolution is impossible, so use `-pixelwidth` and `-pixelheight` instead of `-screenwidth` and `-screenheight`.

By default XaoS changes the colormap. This will collide with other colorful programs like Mortha. You can disable this behavior using `-nopalette` switch, but this will slow down XaoS.

Plan9 driver was written by Jan Hubicka, 1997.

8.8 SVGAlib driver

This is a driver for Linux SVGAlib. I really like this driver, because I much prefer full screen zooming instead of a small 320x320 window in X11. It was one of the first drivers for XaoS and is fully featured. The following problems can occur:

XaoS doesn't initialize graphics mode
: when started under users other than root SVGAlib requires root privileges to directly access the hardware. When you really want to start XaoS as a normal user, enable the suid bit (`chmod +s`) at XaoS executable. note that I take care to disable all security holes caused by this bit so I believe it is safe.

Mouse doesn't work

Screen is blank at higher resolutions
: Both this problems are probably caused by misconfiguration of SVGAlib. Please configure it in `etc/vga/libvga.cong` or `/usr/local/lib/libvga.conf` GPM can also cause problems. Try to kill it before starting XaoS.

When I switch console I can't switch back
: This is another typical SVGAlib bug. Try to hold `F` key longer than `alt`. It helps on my computer. On older SVGAlib there was a famous "enter bug" that caused a crash after pressing enter. Try to update to a newer release.

SVGAlib driver was written by Jan Hubicka, 1997.

8.9 Win32 driver

This is a driver for Windows 9x and NT. It is new since version 3.1 and because of some oddities of Windows API it may be rather unstable. Please report all bugs. In case of problems you could use the DOS version of XaoS instead.

The driver should work in all bit depths, but 16 color mode is not natively supported by the XaoS engine. XaoS internally works in 32k colors and the result is converted to 16 colors by Windows. Because Windows conversion routines are slow and ugly, the result is slow and ugly. Please configure your display to another bit depth to "solve" this problem.

Use `-size WIDTHxHEIGHT` command line option to change the default window size.

This driver also maps to native Windows look and feel. There is a small problem with combo boxes in dialogs. They are expected to give you a choice between a few strings. The keyboard controls (changing choice by arrow keys) work, but mouse selection is broken. If you know how to solve this bug, please let me know.

XaoS is a UNIX application and has many command line options. Some features are not available from the GUI. Because Windows applications can't have normal output, most of the critical messages are displayed in message boxes, but some longer messages are omitted. The most significant omission is the help about command line options that you can find in `doc/cmdopts.txt`.

One thing that might be confusing is that animation rendering mode doesn't display anything, but only renders images. Start the rendering, and a message box will inform you that XaoS is entering the calculation loop. Relax and wait for the message box signaling the end of the loop.

Note that XaoS also supports the DirectX API.

Win32 driver was written by Jan Hubicka, Jan Olderdissen and Pavel Tzekov, 1999.

8.10 X11 driver

This was the first driver done for XaoS. It supports many visuals, shared colormaps and MitSHM extension.

Bugs/limitations:

XaoS makes some X servers too busy

Sometimes XaoS generates images faster than X can display them. In this case XaoS responds poorly to the mouse, and other applications slow down too. This happens especially often on old R4 servers. Use `-sync` to avoid this problem. Note that `-sync` does `not` make all communication with X asynchronous; it just adds one additional XSync call. So the slowdown is not as large as you might expect.

Does not work on all visuals

This driver supports only 8bpp pseudocolor/grayscales, 15,16,24 and 32bpp truecolor, 1bpp and 8bpp staticolor visuals.

Palette rotating does not work for 8bpp pseudocolor w/o private palette

X11 driver was written by Jan Hubicka and Thomas Marsh, 1997.

Appendix A Menus, functions and command line parameters

All XaoS functions are referenced by a central function registry. The scripting language, menus, dialogs and command line options are built from this database. This section contains information about all functions available in this registry.

A.1 Animation root menu

This menu is displayed at the top of the screen when animation replay is active.

A.1.1 Stop replay

Terminate animation replay.

Available as: menu item

A.2 Replay only commands

Some commands, such as timing primitives or animation functions, are available only in animation files.

A.2.1 Include file

`load` *file* [Function]

This function lets you include another file in your script. It works similarly to `#include` in C or `load` in Scheme. The file is searched for in the same directory as the current source file.

Available as: command

A.3 Line drawing functions

XaoS has support for drawing lines. These functions are available only in animations and could be used to show some parts of fractals or draw simple diagrams. See the tutorial "Introduction to fractals" for examples of this feature.

Lines can be drawn in *screen* coordinates, where 0,0 is the top left corner and 1,1 is bottom right, *scaled* coordinates, which are similar, but scaled to keep 0,0—1,1 rectangular, or *Fractal* coordinates, to draw a line at an exact position on the screen.

The color of the line should be specified by the `color` command. You might draw an arbitrary number of lines and, later, morph them. Each line is identified by a unique numeric key; the current key can be set using `linekey`. Commands for changing lines operate on the line with the current key. (Lines drawn in sequence have consecutive numbers.)

For example:

```
(color 'red)
(line 'scaled 0.3 0.5 0.7 0.5)
(line 'scaled 0.3 0.5 0.7 0.5)
(line 'scaled 0.3 0.5 0.3 0.5)
(line 'scaled 0.7 0.5 0.7 0.5)
(linekey 0)
(morphline 'scaled 0.3 0.3 0.7 0.3)
(morphline 'scaled 0.3 0.7 0.7 0.7)
(morphline 'scaled 0.3 0.3 0.3 0.7)
(morphline 'scaled 0.7 0.3 0.7 0.7)
(usleep 1000000)
```

Creates line morphing to rectangle.

`line` *keyword complex complex* [Function]
Draw line between two points. `keyword` specifies type of coordinates and should be one of the following: `'fractal`, `'screen` or `'scaled`. This function also increases the line key.

Available as: command

`morphline` *keyword complex complex* [Function]
Morph current line to given coordinates. `keyword` specifies type of coordinates and should be one of the following: `'fractal`, `'screen` or `'scaled`. The line will start moving at the next timing command, and reach the final position before the end of it. This function also increases the line key.

Available as: command

`morphlastline` *keyword complex complex* [Function]
This function has the same functionality as morphline, but it doesn't increase the line key, and touches the line with the previous key. This is useful when you want to move a just-drawn line—you don't need to set linekey back.

Available as: command

`linekey` *integer* [Function]
Set current line key.

Available as: command

`clearline` [Function]
Clear current line. This function also increases the line key.

Available as: command

`clearlines` [Function]
Clear all displayed lines. Lines can also be cleared using the `clearscreen` or `display` commands available from the Misc menu. See Section A.14 [misc], page 36.

Available as: command

A.4 Animation functions

These functions are used to animate fractal state—to zoom, unzoom and morph various parameters. They should be used only in animation files. Animations are usually performed for a time selected by an immediately following timing function. See Section A.5 [time], page 21. For example:

```
(view 0 0 1 1)
(morphview 0 0 2 2)
(usleep 5000000)
```

Will do a 5 second long unzooming animation.

A.4.1 Animate view

`animateview` *float float float float* [Function]
This function is almost identical to function `view`. See Section A.8.5 [uiview], page 25. It expects that the view will be changed only slightly, so recalculation is done with `ANIMATE` priority. This means that dynamic resolution is used by default.

Viewport is selected by the center and two radiuses (real and imaginary). See `view` for more information.

Available as: command

A.4.2 Smooth Morphing

`morphview` *keystring starttime endtime* [Function]

This function lets you smoothly start and stop morphing. Specify starttime and stoptime as nonzero, and morphing will speed up and slow down for that number of usecs.

The keystring is used to select what morphing you want to control. It is one of the following:

'view control morphview

'angle control morphangle

'julia control morphjulia

'line control morphline

A.4.3 Morph view

`morphview` *float float float float* [Function]

For the time selected by the next `usleep` or other timing function, the viewpoint is smoothly morphed from the current one to that selected by `morphview`.

Viewport is selected by the center and two radiuses (real and imaginary). See `view` for more information.

This function can easily be used for creating zooming/unzooming animations using position files. This is an example position file generated by XaoS:

```
(initstate)
(defaultpalette 0)
(formula 'mandel)
(view -1.64128273713 -5.50393226816E-05 9.69332308848E-08
9.69332308834E-08)
```

By replacing the `view` command with `morphview` and adding `usleep` you can create a zooming animation:

```
(initstate)
(defaultpalette 0)
(formula 'mandel)
(morphview -1.64128273713 -5.50393226816E-05 9.69332308848E-08
9.69332308834E-08)
(usleep 10000000)
```

Available as: command

A.4.4 Morph Julia

`morphjulia` *complex* [Function]

For the time selected by the next `usleep` or other timing function, the Julia seed is smoothly interpolated from the current one to that selected by `morphjulia`. By default this will cause recalculation of the whole screen. To avoid this, use fast Julia mode. See Section A.9.4 [fastjulia], page 30.

A simple animation morphing Julia seed in the X axis:

```
(initstate)
(fastjulia #t)
(juliaseed -2 0)
(morphjulia 2 0)
(usleep 2000000)
```

Available as: command

A.4.5 Move view

`moveview` *complex* [Function]

Smoothly move the image center to another position.

Available as: command

A.4.6 Morph angle

`morphangle` *float* [Function]

Smoothly rotate the image to another angle. By default rotation causes recalculation of the whole screen. To avoid this you need to enable fast rotate mode. See Section A.11 [rotate], page 30. Don't forget to disable it later, since it slows down other animations.

A simple "rotozooming" animation:

```
(initstate)
(fastrotate #t)
(morphview -1.64128273713 -5.50393226816E-05 9.69332308848E-08
9.69332308834E-08)
(morphangle 300)
(usleep 10000000)
(wait)
(fastrotate #f)
```

Available as: command

A.4.7 Zooming functions

The functions for zooming/unzooming were created mainly for recording animations. In manually created animation files, it is easier to use `morphview`. See Section A.4.3 [morphview], page 20.

`zoomcenter` *complex* [Function]

This function sets the center to zoom in on. The center is given as a position in fractal coordinates.

Available as: command

`zoom` [Function]

Start zooming to the area specified by `zoomcenter`.

The speed of zooming should be controlled by the function `speed` or in a more exact way by `maxstep` and `speedup`.

`unzoom` [Function]

Start unzooming from the area specified by `zoomcenter`.

Available as: command

`stop` [Function]

Stop zooming or unzooming.

Available as: command

A.5 Timing functions

Timing functions are used to control the animation replay daemon. It can wait for a given time, or wait until calculation is complete. The animation functions are controlled by such waiting; animations that are running while delays start keep running through the delay.

A.5.1 Usleep

`usleep` *integer* [Function]

This function waits for a given amount of time (in usec) before processing the next command. The screen is recalculated and displayed at the beginning of usleep if necessary necessary. The remaining time is spent in waiting or performing animation.

Available as: command

A.5.2 Wait for text

`textsleep` [Function]

This function's behavior is very similar to `usleep`, but the time is calculated from the number of letters currently displayed onscreen. If you want to wait just long enough for the user to read the subtitle, use this function. The user can alter the replay speed as desired using `letterspersec`. See Section A.13.2 [letterspersec], page 34. This value can be changed during replay with the arrow keys.

Available as: command

A.5.3 Wait for complete image

`wait` [Function]

Wait until the image is complete. You should always use this function after zooming or unzooming when dynamic resolution is in use. This ensures that the image calculation will be complete so the user can see the result before the animation continues. It may also be useful in combination with filters like motion blur. See Section A.12.7 [blur], page 33.

This function deadlocks if used with animation functions; don't do that.

Available as: command

A.6 File

A.6.1 Load XaoS position file

Load a XaoS position file (`*.xpf`). See the format description for more information.

Available as: menu item, command line option

A.6.2 Save XaoS position file

`savepos` *file* [Function]

Save current state to a XaoS position file (`*.xpf`). This file is human-readable, and can easily be improved by hand after saving, or used as a base for animations. See the format description for more information.

Available as: menu item, command line option, command

A.6.3 Record animation

`record` *bool* [*file*] [Function]

e Toggle recording to a XaoS animation file (`*.xaf`). This file is human-readable, and can easily be improved by hand after recording. See the format description for more information.

From the scripting language, `(record #t)` enables recording, and `(record #f)` disables it.

Available as: menu item, command line option, command

A.6.4 Replay animation

Replay a XaoS animation file (`.xaf`).

Available as: menu item, command line option

A.6.5 Save image

`saveimg` *file* [Function]

Save current state to an image file. This file is in `.png` (portable network graphics) format, which can be read by many applications varying from graphics programs all the way to Web browsers.

This function needs an external library called `libpng`. If the library wasn't available during compilation, this function is unavailable too. Please see `INSTALL` for more information about obtaining libpng and recompiling XaoS.

Available as: menu item, command line option, command

A.6.6 Load random example

`loadexample` [Function]

Choose random `.xpf` file from the `examples` directory and load it. You might use it as the starting point for next exploration.

Available as: menu item, command line option, command

A.6.7 Save configuration

`savecfg` [Function]

Save current configuration to `~/.xaosrc` (under Unix) or `xaos.cfg` (under DOS and Windows). XaoS automatically reloads the configuration from this file when it starts.

Available as: menu item, command line option, command

A.6.8 Quit

`quit` [Function]

Quit XaoS.

Available as: menu item, command line option, command

A.7 Edit

A fairly ordinary Edit menu.

A.7.1 Undo

Undo last operation. 'Last operation' is quite hard to define in XaoS (where changes are continuous), so it might be surprising. I hope it will do what you want.

Available as: menu item

A.7.2 Redo

Redo last undone operation. See undo. See Section A.7.1 [undo], page 23.

Available as: menu item

A.7.3 Copy

Copy fractal to clipboard. This is a platform-dependent operation that may not have an analogue on your platform (e.g. there is no concept of a clipboard under aalib).

Available as: menu item

A.7.4 Paste

Paste fractal from clipboard. This is a platform-dependent operation that may not have an analogue on your platform (e.g. there is no concept of a clipboard under aalib).

Available as: menu item

A.8 Fractal

This menu contains all functions related to fractal parameters and display; you can change things like the formula used, coloring modes, seeds and much else.

A.8.1 Formula

`formula` *keyword* [Function]

Set the current fractal formula. `keyword` should be one of the following:

'mandel Standard Mandelbrot set. See Section A.18.1 [mandel], page 37.

'mandel3 Mandelbrot set, power 3. See Section A.18.2 [mandel3], page 37.

'mandel4 Mandelbrot set, power 4.

'mandel5 Mandelbrot set, power 5.

'mandel6 Mandelbrot set, power 6.

'newton Newton's approximation method. See Section A.18.4 [newton], page 38.

'barnsley First Barnsley's formula. See Section A.18.5 [barnsley], page 38.

'octo Fractint's octo. See Section A.18.3 [octal], page 37.

'phoenix Phoenix. See Section A.18.6 [phoenix], page 38.

'magnet Magnet. See Section A.18.7 [magnet], page 38.

Available as: command

A.8.2 Mandelbrot/Julia mode

Most fractals rendered by XaoS can be represented as Mandelbrot sets or Julias. Each point in the Mandelbrot set has its own Julia set. To learn more about this correspondence, see the tutorial on the Julia set.

This function switches between Mandelbrot and Julia representations. When switching to Julia, you need to set the seed—a point selected from the Mandelbrot set.

If you run this function from the menu, you are prompted for the Julia seed as a number. Often, this can be clumsy, and it would be easier to specify a point with the mouse pointer. If you hit the `M` key instead of using the menu, the current mouse position is used.

Good seedpoints lie at the boundaries of the Mandelbrot set; other seeds usually generate quite a boring fractal. You can also explore various seeds at high speed using the Fast Julia mode. See Section A.9.4 [fastjulia], page 30.

Not all fractals have Julias, but XaoS can generate fake Julia sets for those that do not, which use some Julia-like modification of the formula; so this function is currently usable for all fractal types.

Available as: menu item

`julia` *bool* [Function]

This function is used to enable/disable julia mode in animation files.

Available as: command line option, command

`juliaseed` *complex* [Function]

Select the current julia seed.

Available as: command line option, command

A.8.3 Perturbation

Perturbation is a simple trick which changes the point at which orbits start. Traditionally zero is used, but other values can generate interesting results too.

On enabling this function from the menu, you will be asked for a complex number specifying the perturbation. It is a toggle; selecting it again resets the perturbation to zero without prompting.

It can be used to specify a complex number representing a point on the screen. If you hit the `B` key instead of using the menu, the current mouse position is used. This too is a toggle, so `B` again will disable perturbation by setting it to zero.

This function only has an effect for certain formulae (like the Mandelbrot set) and only then in <a uimandelbrot>Mandelbrot mode. See Section A.18.1 [mandel], page 37.

Available as: menu item

`perturbation` *complex* [Function]

This is the scripting-language variation of the perturbation function. Instead of toggling, you always specify the perturbation to use. Use 0 0 to disable perturbation.

Available as: command line option, command

A.8.4 Bailout

Bailout is the value which is checked for each point of the orbit if the point is far enough from the complex zero point in the current iteration. If the point is far enough, then the iteration immediately stops and the starting point on the screen will be painted with a given colour, depending on the fractal type and many other settings.

For the Mandelbrot set this value is 4. Other fractal types usually have the same bailout value. For most fractals many bailout values give more or less similar output. E.g., for the second order Mandelbrot set one can prove that the sequence `|z|` (z:=z^2+c) tends to infinity if and only if `|z|>2` for some element z of this sequence. In XaoS program, Bailout value is the square of this 2, i.e. you can change this to any value greater than 2 for similar results.

Other fractal types may use other bailout values. The default is 4 for each types.

Available as: menu item, command line option, command

`bailout` *float* [Function]

A.8.5 View

Set your current viewpoint in the fractal. This function is useful when you have found some interesting coordinates somewhere (on a web page, perhaps) and you want to see that position in XaoS.

In the dialog you will be asked for the *center*, *radius* and *angle* of the image.

The center specifies the point which is displayed at the center of the screen. The radius is the radius of a circle around this point; XaoS will size the image so that this circle only just fits on the screen. The angle gives the rotation of the image in degrees.

People specify fractal coordinates in many ways. Some people use the coordinates of the upper-left and lower-right visible points, specifying the coordinates as four numbers $x1$, $y1$, $x2$, $y2$. To set the same viewpoint in XaoS, set the real portion of the center to $(x1 + x2)/2$, the imaginary part of center to $(y1 + y2)/2$, and the radius to the greater of $x2 - x1$ and $y2 - y1$.

Other programs use a zoom factor instead of a radius. For these, you can set the radius to $2/zoom$.

Available as: menu item

`view` *float float float float* [Function]

This function is used to set the visible area of fractal in animation files. It doesn't let let you specify the angle, (for that, see the separate function `angle`), but lets you specify an ellipse instead of a circle. You can specify both a real and an imaginary radius, so you have better control over the area that will be visible. XaoS will size the image so that the ellipse only just fits on the screen.

Available as: command line option, command

`angle` *float* [Function]

Set the rotation angle in degrees. By default this causes recalculation of the screen. You can enable the fast rotation mode, which lets you rotate the screen without recalculation; but it slows down other things, so don't forget to disable it later.

Available as: command line option, command

A.8.6 Reset to defaults

`initstate` [Function]

This function resets most of XaoS' values to their defaults. It is useful when you get lost and want to start from the beginning. It should also be used as the first command of every animation file, to ensure that the file is always played with the same settings in effect.

Available as: menu item, command line option, command

A.8.7 Plane

`plane` *integer* [Function]

All fractals displayed by XaoS are functions with a complex parameter. They can be be displayed in the normal complex plane where the `x` coordinate is the real part of the number and the `y` is imaginary; but they can also be displayed differently:

mu Normal complex plane (default)

$1/mu$ Inversion—infinity is at 0 and 0 is at infinity.

$1/(mu + 0.25)$
Similar to inversion, but moves the center outside the Mandelbrot set, so it looks parabolic.

lambdaplane, $1/lambda$, $1/lambda - 1$
Lambda plane and its inversion, and with a different center.

$1/(mu - 1.40115)$
A very interesting mode for the Mandelbrot set, this makes small things large, for easier browsing of the set's details.

The tutorial about planes has some examples.

In the scripting language, the planes are numbered as follows:

0 mu

1 $1/mu$

2 $1/(mu + 0.25)$

3 $lambda$

4 $1/lambda$

5 $1/(lambda - 1)$

6 $1/(mu - 1.40115)$

Available as: command line option, command

A.8.8 Inside coloring mode

`incoloring` *integer* [Function]

Areas inside the set are usually filled in black, but this is only a convention; you could color them in differently to make the fractal look more interesting. The only method available to make areas inside the set visible is to display the value of the latest orbit as the value of each pixel.

The tutorial on incoloring has more information and examples.

XaoS has many different ways to show that value. The cryptic names of the modes are mathematical formulae, where *real* means the real part of the latest orbit, and *imag* means the imaginary part. *zmag* uses the magnitude of the value. The *Decomposition-like* method uses the angle of the orbit. Also, truecolor incoloring modes are available, that display one value in each of the red, blue and green color planes (or, for some modes, in each of the hue, saturation and value planes).

In the scripting language, the incoloring mode is specified by one of the following integers:

0 0 (default)

1 $zmag$

2 Decomposition-like

3 $real/imag$

4 $abs(abs(c) - abs(r))$

5 $cos(mag)$

6 $mag * cos(real^2)$

7 $sin(real^2 - imag^2)$

8 $atan(real * imag * creal * cimag)$

9 squares

10 Truecolor. To set exact parameters for truecolor coloring use the `tcolor` command.

Available as: command line option, command

A.8.9 Outside coloring mode

`outcoloring` *integer* [Function]

Outcoloring modes are similar to incoloring modes, but indicate how to display the areas outside the set instead. As with incoloring modes, the value of the latest orbit can be used to determine the color of each pixel, but the default is to use the number of iterations needed for the value at that point to become recognisably divergent as the color.

The tutorial on outcoloring has more information and examples.

The cryptic names of the modes are mathematical formulae, where *iter* means the number of iterations required for the value to become recognisably divergent, *real* means the real part of the latest orbit, and *imag* means the imaginary part. *binary decomposition* uses a different color when the imaginary part of the orbit is lower than zero, and *smooth* attempts to remove stripes and discontinuities. Also, truecolor outcoloring modes are available, that display one value in each of the red, blue and green color planes (or, for some modes, in each of the hue, saturation and value planes).

In the scripting language, the outcoloring mode is specified by one of the following integers:

0 *iter* (default)

1 $iter + real$

2 $iter + imag$

3 $iter + real/imag$

4 $iter + real + imag + real/imag$

5 binary decomposition

6 biomorphs

7 potential

8 color decomposition

9 smooth

10 True-color outcoloring mode. To set exact parameters for truecolor coloring use `outtcoloring`. See Section A.8.10 [tcolor], page 28.

Available as: command line option, command

A.8.10 Truecolor coloring mode

`intcoloring` *integer* [Function]

`outtcoloring` *integer* [Function]

Truecolor coloring modes are similar to incolor and <a outcoloring>outcolor coloring modes; but instead of using a palette, they directly calculate the red, green and blue components of the color. This lets you display more parameters at once, and produces interesting and often attractive results. On 8bpp displays you need to enable the palette emulator filter first to see anything, amd the quality won't be so good, as far fewer colors are available per parameter.

The tutorial on truecolor coloring modes has more information and examples.

The cryptic names of the modes are always three mathematical formulae (one for each color component), where *real* means the real part of the latest orbit, and *imag* means the imaginary part.

To enable inside/outside truecolor coloring mode in the scripting language, set `incoloring`/`outcoloring` value to 10 (truecolor coloring mode) before (or after) calling `intcoloring` or `outtcoloring`.

In the scripting language, the coloring mode is specified by one of the following integers:

0 black

1 $re * im\ sin(re^2)$ angle

2 $sin(re)\ sin(im)\ sin(square)$

3 hsv

4 hsv2

5 $cos(re^c)\ cos(im^2)\ cos(square)$

6 $abs(re^2)\ abs(im^2)\ abs(square)$

7 $re * im\ re * re\ im * im$

8 $abs(im * cim)\ abs(re * cre)\ abs(re * cim)$

9 $abs(re * im - csqr)\ abs(re^2 - csqr)\ abs(im^2 - csqr)$

Available as: command line option, command

A.9 Calculation

This menu contains functions that control calculation parameters such as the maximum iteration count and periodicity checking.

A.9.1 Solid guessing range

`range` *integer* [Function]

XaoS has a solid guessing optimization: if all corners of a rectangle have the same color, it assumes that the whole rectangle is a solid colored block, and doesn't calculate points inside the rectangle. This optimization saves lots of calculation, but sometimes introduces errors. This value alters the maximum size of the rectangle that can be guessed at one time. The default value is 3; use 0 to disable the optimization.

Available as: command line option, command

A.9.2 Periodicity checking

`periodicity` *bool* [Function]

Periodicity checking is one way to speed up the calculation. Areas inside the set always need `maxiter` iterations to determine that the point is probably inside the set (while it is rare for areas outside to need anywhere near that much). Often the orbital trajectory falls into a periodic, repeating cycle; if that can be detected, the calculation can be stopped early, as there's no way that the orbit can ever leave the cycle again (hence it cannot diverge, hence the point must be inside the set).

Implementating this method efficiently is quite problematic. It slows down the cases where cycles are not found, because cycle-checking is quite hard work and has to take place for all points, even those that don't become cyclic. Because of the inexactness of floating-point calculations, the cycles are never exact, so you need to use an error value. Higher error values mean that cycles will be detected sooner, while lower error values increase the exactness of the calculation. Higher values can introduce serious errors, especially at the front of the Mandelbrot set. XaoS detects this automatically and corrects for it in most cases, but sometimes it might be wrong. Also, other optimizations in XaoS (such as boundary tracing) don't give this method much of a chance to run, since areas inside the set are usually not calculated at all.

That's why the advantages of this optimization are questionable. You should probably experiment with enabling and disabling it. Sometimes XaoS is faster with this enabled, sometimes when disabled. Also, this method works only when incoloring methods are disabled, and only for some fractal types (some fractal types, e.g. newton, don't have any concept of an area 'inside the set' at all.)

The tutorial chapter "Escape time fractals" has more information on fractal calculation in XaoS, and there is a lengthy section in the hacker's manual (`xaosdev.texinfo`) devoted to the subject.

Available as: menu item, command line option, command

A.9.3 Iterations

`maxiter` *integer* [Function]

When the fractal set is calculated, a orbital trajectory is examined for each point. If the orbit diverges to infinity, the point is outside the set. Otherwise, the point is inside the set. For exact calculations, you need to know the entire orbital trajectory, which is infinitely long for areas inside the set, so fractals cannot be calculated exactly. By default, XaoS calculates at most 170 positions (iterations) and then gives up; if the point is still inside the bail-out value, it guesses that the point is inside the set.

When zoomed into a detailed area, especially one close to the set boundary, this value could become too low, and the fractal will become boring. You might try increasing this value if you want to get the image interesting again; but this necessarily slows down the calculation at the same time.

The tutorial chapter "Escape time fractals" has more information on fractal calculation in XaoS, and there is a lengthy section in the hacker's manual (`xaosdev.texinfo`) devoted to the subject.

Available as: menu item, command line option, command

A.9.4 Fast Julia mode

`fastjulia` *bool* [Function]

By default, changing the seed for the Julia set requires recalculation of the image (which is quite slow). It's a nice effect to change the seed smoothly and show the Julia set morphing as the seed changes. XaoS has a special algorithm which can calculate such morphings in realtime. It is very inexact, but it is good enough for a fast preview.

If you want to select a good seedpoint, enable fast Julia mode and find a nice place by dragging with the first mouse button depressed; then change to the Julia mode to see the exact image.

Available as: menu item, command line option, command

A.10 Dynamic resolution

`fastmode` *keyword* [Function]

XaoS performs many optimizations, but fairly often this is not enough. In order to keep a high framerate, XaoS automatically lowers the resolution of the image, increasing it when there is time for more calculation. This feature is enabled by default when animating, but you might also like to enable it for new images (which makes the image 'come into focus' when it is recalculated from scratch for whatever reason), or disable it completely if you don't like it.

In the scripting languge, the keyword should be one of the following:

`'never` Disable dynamic resolution

`'animate` Use only for animations (default)

`'new` Use also for new images

A.11 Image rotation

XaoS has support for rotation of the image to any angle. By default, changing the angle requires recalculation of the whole screen, but when *fast rotation mode* is enabled, the angle can be changed smoothly. In this mode XaoS calculates a larger non-rotated image and rotates it when needed, so it increases memory requirements and slows XaoS down; hence, it should be disabled when rotation is not being used.

The user interface provides two rotation modes—*rotate by mouse* which allows the angle to be changed by dragging with the first mouse button depressed, and *continuous rotation mode*, where the image is rotated clockwise continuously, and the arrow keys can be used to change the rotiation speed.

`fastrotate` *bool* [Function]

This function is used to enable and disable fast rotation mode. *Available as*: command line option, command

A.11.1 Automatic rotation

`autorotate` *bool* [Function]

Use this function to enable continuous rotation. In the scripting language you can also use `morphangle` to get an outwardly similar but more controllable effect.

`rotationspeed` *float* [Function]

Specify the speed of continuous rotation, in degrees per second. Negative values are allowed and rotate anticlockwise.

Available as: menu item, command line option, command

A.12 Filters

Filters are a post-calculation effect applied to the resulting image. They can do things like motion blurring, edge detection, emulation of palettes or truecolor on displays that can't handle them, and such things. There is a tutorial chapter about them.

A.12.1 Filter command

`filter` *keyword bool* [Function]

This command is used to enable or disable filters. See Section A.12 [mfilter], page 31. The *keyword* specifies the filter to change, and should be one of the following:

`'edge` Edge detection

`'edge2` Edge detection2

`'starfield`
: Starfield

`'stereogram`
: Random dot stereogram

`'interlace`
: Interlace filter

`'blur` Motion blur

`'emboss` Emboss

`'palette` Palette emulator

`'anti` Antialiasing

`'truecolor`
: Truecolor

Available as: command

A.12.2 Edge detection

This filter is a standard edge detection algorithm; solid areas are filled in black. Some fractals look very interesting with this filter (and some areas of some fractals just look like noise). This version of the filter produces relatively wide lines, so is useful at higher resolutions. The filter edge detection2 makes thinner lines, for the low resolution modes.

Available as: menu item, command line option

A.12.3 Edge detection2

This filter is a standard edge detection algorithm; solid areas are filled in black. Some fractals look very interesting with this filter (and some areas of some fractals just look like noise). This version of the filter produces relatively tight lines, so is useful at lower resolutions. The filter edge detection makes thinner lines, for the high resolution modes.

Available as: menu item, command line option

A.12.4 Starfield

The starfield filter generates random stars whose density depends on the iteration count. Choose your favorite spiral fractal and enable this filter to get a Grand Design spiral galaxy :)

Available as: menu item, command line option

A.12.5 Random dot stereogram

Fractal images are good as a base for random dot stereograms. In case you don't know what these are, please point your browser to Google or another search engine and find some articles about such images, because learning to read such images takes some effort. They make it possible to generate three dimensional images on a normal monitor without any additional hardware, by exploiting bugs in the human brain (although you need two working eyes, and some people never learn to see them; they can simply ignore this feature).

XaoS is able to generate these images in animations, so you may use all normal XaoS functions (except palette changing and palette rotation, which makes no sense applied to a stereogram). To make the animation yet more exciting, XaoS emulates "falling" into the set; while you zoom in, your distance from the set drops and drops—but you never hit it; when the set reaches the level of your monitor, the distance is changed again so you are far away.

To make this work right, XaoS needs to know the *exact size of your monitor*. Because most platforms have no way to determine this, you need to use *command line options* to tune it. If it's not set or is wrong, the stereograms will probably be impossible to see (if your monitor is too big or resolution too low), or the images will seem to be shallow (if your monitor is too small or resolution too high).

By default XaoS expects my 15" monitor (29.0cm x 21.5 cm). Another cause of problems is the virtual screen supported by some windowed environments (like some X servers) that makes a program think that the resolution is higher than it actually is, and you see only part of this extra-large screen.

The worst thing you could possibly do is to run full-screen XaoS in some graphical windowing system (OS/2 on top of Windows or Wine on top of Linux, perhaps) where XaoS can't tell the real size of its window at all. In such cases, it's normally better (not to mention faster) to run XaoS natively, rather than under such an emulation layer.

The following command line options are provided to specify sizes:

`-screenwidth`, `-screenheight`
: Lets you specify the size of your screen in centimeters. Note that you need to specify the size of the visible image on the monitor, not the size with edge borders, or the size of the tube. The simplistic 'my monitor is 17", just turn 17" into centimeters' doesn't work; that 17" is a marketing figure and has only a vague connection to reality. Get out a ruler and measure it.

`-pixelwidth`, `-pixelheight`
: Lets you specify the exact size of a single pixel, if XaoS cannot determine this for itself from your screen size.

These options are used by some other parts of XaoS as well, so you should use them even when you don't want to see stereograms. You should probably write a small starting script (or alias, or shortcut; whatever your environment uses) that passes the correct parameters to XaoS.

If the window is *smaller than 8cm in any direction*, you will probably be unable to see anything; make the window bigger.

The correct way to see XaoS stereograms is:

1 Start XaoS with options specifying the exact size of your screen or one pixel on it

2 Sit 60cm away from monitor

3 If you use a windowed environment, resize XaoS' window to make it wider than, say, 15 cm.

4 Enable the filter (by pressing `E`)

5 focus on a point far away from the monitor (try to use your own reflection, if your monitor's not antireflective); the random blurring should eventually fall into the pattern of a Mandelbrot set.

6 Carefully use your mouse to zoom into interesting areas (it is easy to lose concentration when you are not trained; but you can use the autopilot...)

7 Enjoy animation :)

If you still can't see the stereograms, it could be that the fractal, or your eye, is deformed. A deformed fractal can be caused by your specifying your monitor size wrongly. Visual problems that damage depth perception, as well as problems like astigmatism, can make it impossible to see stereograms at all.

Available as: menu item, command line option

A.12.6 Interlace filter

The interlace filter halves the horizontal resolution, and in each frame alternates between drawing only the even and only the odd lines. This speeds up the calculation, and in higher resolutions produces a motion-blur-like effect.

Available as: menu item, command line option

A.12.7 Motion blur

Motion blur mixes the current frame with previous ones to produce a motion-blur effect. It might be rather slow in 16bpp truecolor modes. The best results can probably be seen in 8bpp modes, so you might want to enable the palette filter first.

Available as: menu item, command line option

A.12.8 Emboss

This is a standard emboss filter, as seen in programs such as the GIMP or Photoshop. It produces especially nice results with the smooth outcoloring mode. See Section A.8.9 [outcoloring], page 27.

Available as: menu item, command line option

A.12.9 Palette emulator

XaoS can work in either palette or truecolor mode. Both modes have advantages and disadvantages. Palette mode allows effects such as palette rotation, while truecolor mode allows smoother incoloring and outcoloring modes and the truecolor coloring modes. If your display is truecolor, you can enable this filter to get palette emulation (albeit not as cheaply as in a real paletted mode).

Available as: menu item, command line option

A.12.10 Antialiasing

Antialiasing is a technique to increase image quality by eliminating jagged edges. XaoS calculates four values for each pixel (on the subpixel boundaries) and uses the average of them for the pixel value.

This filter slows XaoS down a *lot* and greatly increases memory requirements. It is useful mainly when you want to save images and want to make them look as nice as possible. Antialiasing also helps a lot when you want to encode JPEG or MPEG files; they are much shorter if antialiased (MPEG and JPEG hate jagged edges).

Available as: menu item, command line option

A.12.11 Truecolor emulator

XaoS can work in either palette or truecolor mode. Both modes have advantages and disadvantages. Palette mode allows effects such as palette rotation, while truecolor mode allows smoother incoloring and outcoloring modes and the truecolor coloring modes. If your display is 8bpp, you can enable this filter to get truecolor emulation (but, obviously, not with as many colors as a real truecolor display).

More information about filters

Available as: menu item, command line option

A.13 UI

This menu contains functions to control the user interface layer of XaoS: zooming speed, the autopilot, realtime status information, and so on.

A.13.1 Zooming speed

`speed` *float* [Function]

Change zooming speed, where 1 is the default, 2 means twice as fast, and so on.

Available as: menu item, command line option, command

In the scripting language you can use the following functions for better control:

`maxstep` *float* [Function]

Selects the zooming/unzooming speed. The parameter specifies how much of the range will be removed each twentieth of a second; 0 means nothing, 1 means everything (the parameter obviously has to be less than 1). Higher values mean faster zooming.

Available as: command

`speedup` *float* [Function]

When zooming/unzooming, every twentieth of a second the `speedup` value is added to the current step until `maxstep` is reached. So this value selects the rate at which zooming stops and starts. Both these functions are more for internal use of XaoS then for manually written scripts, but they could come in useful nonetheless.

Available as: command

A.13.2 Letters per second

`letterspersec` *integer* [Function]

Speed of subtitles for the `textsleep` function. The user can set this value to suit; it can also be changed with the left and right arrow keys during animation replay.

Available as: command line option, command

A.13.3 Autopilot

`autopilot` *bool* [Function]

To make XaoS yet more impressive, we made a special autopilot mode that automatically drives into interesting boundaries of the set; you should press `A`, play your favorite music, drink coffee and relax. I never tried this but it should be really relaxing! Many pictures in the XaoS gallery were discovered using the autopilot.

The autopilot also has some additional features. It backtracks if the zoomed picture is not interesting anymore, and can detect when it's zoomed into really a boring part of the fractal or reached the limit of floating point arithmetic on the platform, and restart zooming from the top.

Available as: menu item, command line option, command

A.13.4 Recalculate

`recalculate` [Function]

Recalculate current fractal. This should be used when the fractal on the screen is strange because of error propagation caused by solid guessing. See Section A.9.1 [range], page 29.

Available as: menu item, command line option, command

A.13.5 Interrupt

`interrupt` [Function]

Interrupt current calculation.

Available as: menu item, command line option, command

A.13.6 Disable XaoS's builtin GUI

`nogui` *bool* [Function]

Disable XaoS menus and dialogs. This function should be used by external GUI programs; these manipulate XaoS via a pipe, so the internal GUI should be disabled at the same time. See the hacker's manual (`xaosdev.texinfo`) for more details.

Available as: menu item, command line option, command

A.13.7 Status

`status` *bool* [Function]

Enable/disable status information. This displays some useful information about the current fractal, such as viewpoint etc. (In low-resolution modes it also almost completely obscures the current fractal...)

Available as: menu item, command line option, command

A.13.8 Ministatus

`ministatus` *bool* [Function]

Enable/disable status line. This contains basic information such as how much you are zoomed and the framerate.

Available as: menu item, command line option, command

A.14 Misc

Miscellaneous functions.

A.14.1 Command

You can invoke all XaoS functions using a simple command language reminiscent of Scheme. This option lets you run a single command. If you want to run more than one, you might want to use an XaoS animation file instead; they are written in the same language.

Available as: menu item

A.14.2 Render animation

Render an animation to image files. See How to encode MPEG files for more information.

Available as: menu item,

A.14.3 Clear screen

`clearscreen` [Function]

Clear the screen. To display the fractal again, use `display`. See Section A.14.4 [display], page 36. This function is mainly useful in tutorials and similar animations.

Available as: menu item, command

A.14.4 Display fractal

`display` [Function]

Display fractal. This functions reverses the effect of the `clearscreen`, line drawing and text output functions.

Available as: menu item, command

A.14.5 Display text

`text` *string* [Function]

Display the given text on the screen. This function is mainly useful in tutorials. Text should be cleared by printing lots of spaces, or using the `clearscreen</a>` or `<a display>display` functions. You might also want to use the `textposition` function to select the part of the screen to display the text on.

To wait for the user to read the text, you can use the `textsleep` function.

Example:

```
(clearscreen)
(textposition 'center 'middle)
(text "Welcome into my animation")
(textsleep)
(display)
```

Available as: menu item, command line option, command

A.14.6 Color

`color` *keyword* [Function]

Change text and line color. *keyword* should be one of `'white`, `'black` and `'red`.

Available as: menu item, command line option, command

A.14.7 Text position

`textposition` *keyword keyword* [Function]

Select text position. The first keyword specifies the horizontal position, the second the vertical position. The horizontal position should be one of `'left`, `'center`, and `'right`. The vertical should be one of `'top`, `'middle`, and `'bottom`.

Available as: command line option, command

A.14.8 Message

`message` *string* [Function]

This function is almost identical to the `text` function, except that it uses message catalogs in the `catalog` directory to translate messages into other languages. It should be used only in the multi-lingual XaoS tutorials.

Available as: command line option, command

A.15 Help

This menu contains help and tutorials.

A.16 Horizontal text position

Select the horizontal position used to display text. See Section A.14.5 [text], page 36. It can be placed at the left, in the center or at the right.

A.17 Vertical text position

Select the vertical position used to display text. See Section A.14.5 [text], page 36. It can be placed at the top, in the middle or at the bottom of the screen.

A.18 formulae

Each escape time fractal has its own formula. XaoS supports the following formulae:

A.18.1 Mandelbrot

The Mandelbrot set is the most famous escape time fractal ever. It has the simple formula $z = z^2 + c$. See the tutorial chapter.

Available as: menu item, command line option

A.18.2 Mandelbrot^3—Mandelbrot^6

This fractal is a simple modification of the standard Mandelbrot set formula, using $z = z^3 + c$ instead of $z = z^2 + c$.

Other derivations of the Mandelbrot set (Mandelbrot^4 and so on) use even higher powers. See the tutorial chapter.

Available as: menu item, command line option

A.18.3 Octal

This is a less well-known fractal that Thomas discovered in Fractint. It has an interesting shape when displayed in the alternative <a mplane>planes. See the tutorial chapter.

Available as: menu item, command line option

A.18.4 Newton

This is Newton's approximation method for finding the roots of a polynomial. It uses the polynomial $x^3 = 1$ and counts the number of iterations needed to reach the approximate value of the root. See the tutorial chapter.

This fractal doesn't have Julia sets, but XaoS is able to generate Julia-like sets which are also very interesting (they are sometimes called "Nova formulae").

Available as: menu item, command line option

A.18.5 Barnsley1

This is a formula by Michael Barnsley. It produces very nice crystalline Julia sets. See the tutorial chapter.

Available as: menu item, command line option

A.18.6 Phoenix

This formula produces very nice Julia sets. See the tutorial chapter.

Available as: menu item, command line option

A.18.7 Magnet

This is a formula that comes from theoretical physics. It is derived from the study of theoretical lattices in the context of magnetic renormalization transformations. See the tutorial chapter.

Available as: menu item, command line option

A.19 Palette

This menu contains functions to change the palette the fractal is displayed with.

A.19.1 Default palette

`defaultpalette` *number* [Function]

Create a default palette. In the scripting language, `number` specifies how much the palette is shifted by.

Note that changing the palette in truecolor modes forces recalculation of the whole screen. To avoid this, you can enable the palette emulation filter first.

Available as: menu item, command line option, command

A.19.2 Random palette

`randompalette` [Function]

Create a random palette. XaoS will automatically pick one of its palette-generation algorithms and create one.

Note that changing the palette in truecolor modes forces recalculation of the whole screen. To avoid this, you can enable the palette emulation filter first.

Available as: menu item, command line option, command

A.19.3 Custom palette

`palette` *integer integer integer* [Function]

A custom palette lets you re-create some of the random palettes. The first value specifies the algorithm, which should currently be one of the following:

0 Default palette

1 Black to color gradient

2 Black to color to white gradient

3 Cubistic-like algorithm.

The seed specifies a random seed for the palette; different seeds generate different palettes. The last value is the amount by which the palette is shifted.

Note that changing the palette in the truecolor modes forces recalculation of the whole screen. To avoid this, you can enable the palette emulation filter first.

Available as: menu item, command line option, command

A.19.4 Color cycling

`cycling` *bool* [Function]

Color cycling is an old and simple effect to animate fractals. The Mandelbrot set looks particularly nice when color-cycled. On truecolor displays, color cycling fails to initialize (since those displays don't have a palette). You can enable palette emulation filter to make it possible.

Available as: menu item, command line option, command

In the user interface, colors can also be cycled in the opposite direction with the "*Reversed color cycling*" function.

To control the cycling speed, you coan use arrow keys or the "*Color cycling speed*" function.

Available as: menu item

`cyclingspeed` *integer* [Function]

The parameter specifies the number of skips per second. It can be negative to cycle in the opposite direction.

Available as: menu item, command line option, command

A.19.5 Shift palette

`shiftpalette` *integer* [Function]

Shift palette by the specified number of cells. This can be used to tune the palette's position on the fractal. You can also use the *Shift one forward* and *Shift one backward* functions for fine-tuning. Note that shifted and rotated palettes could look different on different displays (because they may have different palette sizes).

Shifting the palette on truecolor displays causes a recalculation of the screen. To avoid this, you could use palette emulation filter. See Section A.12.9 [palettef], page 33.

Available as: menu item, command line option, command

Appendix B Credits

(alphabetically)

Eric Courteau (ecourteau@cplus.fr)
francais.cat (translation of tutorials)

Jean-Pierre Demailly (Jean-Pierre.Demailly@ujf-grenoble.fr)
Updates for French translation

Radek Doulik (rodo@atrey.karlin.mff.cuni.cz)
GTK interface, windowid patches

Martin Dozsa (madsoft@centrum.cz)
cs.po (Czech translation of menus)

Arpad Fekete (Fekete.Arpad.2@stud.u-szeged.hu)
some new fractals, and the 'More formulae' menu

Tim Goodwin (tgoodwin@cygnus.co.uk)
english.cat corrections

Ben Hines autoconf suggestions, Mac OS X port

Jan Hubicka (jh@ucw.cz)
Zooming routines, ugly interface, palettes, drivers, autopilot, filters, documentation, tutorials etc.

Jens Kilian (jjk@acm.org)
BeOS driver, deutsch.cat

Thomas A. K. Kjaer (takjaer@imv.aau.dk)
OS/2 ports (320x200 graphics and AA-lib)

Zoltan Kovacs (kovzol@math.u-szeged.hu)
Internationalization, Hungarian translations, finalizing version 3.1, bug fixes, web design, current maintainer

Zsigmond Kovacs
Fractal examples

J. B. Langston III (jb-langston@austin.rr.com)
Native Mac OS X port (from version 3.2.2)

Andreas Madritsch (amadritsch@datacomm.ch)
New fractal types, bailout, many fixes

Giorgio Marazzi (gmarazzi@vtr.net)
Improvements and fixes for espanhol.cat

Thomas Marsh
First zoomer, formulae, planes, X11 driver, inversions, many ideas

Dominic Mazzoni (dmazzoni@cs.cmu.edu)
Macintosh port (version 2.0)

David Meleedy
Grammatical and spelling fixed version of `xaos.6`

Paul Nasca (zynaddsubfx@yahoo.com)
Ministatus improvement

Nix (nix@esperi.demon.co.uk)
Grammatical and spelling fixed version of `xaos.hlp` and other files

Terje Pedersen (terjepe@login.eunet.no)
Amiga port

Cesar Perez (oroz@users.sourceforge.net)
Spanish translations

Fabrice Premel (premelfa@etu.utc.fr)
Periodicity checking

Jan Olderdissen (jan@olderdissen.com)
Win32 port

Ilinca Sitaru (ilinca.sitaru@gmail.com)
Romanian translation

Daniel Skarda
Fractal examples

Andrew Stone (Stone Design - www.stone.com)
Mac OS X improvements, Videator support

Marton Torok (marton.torok@gmail.com)
Small fixes for pipes

Pavel Tzekov (paveltz@csoft.bg)
Win32 support

Charles Vidal
Tcl/Tk interface

Tapio K. Vocaldo (taps@rmx.com)
Macintosh port

Philippe Wautelet (p.wautelet@fractalzone.be)
Bug fixes for version 3.1.1, French translation, gcc 4.0 fixes

To contact the authors you can also use the mailing lists. See Appendix C [lists], page 42.

Appendix C Mailing-lists:

Until 1999 (or so) there was a high traffic mailing list on a mail server located in Czech Republic. Unfortunately, the server with its mailing lists no longer works.

In 2001 XaoS moved to the SourceForge developer site and now there are 3 new mailing lists. Unfortunately, at the time of writing this documentation its traffic is about 0. Hopefully the development will continue using these mailing lists. Namely,

xaos-announce@lists.sourceforge.net,
xaos-devel@lists.sourceforge.net,
xaos-discuss@lists.sourceforge.net.

Please feel free to join any or all of these mailing lists and share your ideas with the developers.

Index of functions

Table of Contents